A Survey on Technical Methods for Analyzing and Predicting the Reliability of Large-Scale Distributed Systems

1. Seyed Ali Sharifi*: School of Electrical and Computer Engineering, University of Kashan, Kashan, Iran. Email: miralisharifi@yahoo.com
2. Seyed Morteza Babamir: School of Electrical and Computer Engineering, University of Kashan, Kashan, Iran. Email: babamir@kashanu.ac.ir
*Corresponding Author: School of Electrical and Computer Engineering, University of Kashan, Kashan, Iran. Email: miralisharifi@yahoo.com

Abstract:
Reliability of distributed systems revolves around the way probability of free of failure software operating in a specified environment for a specified period with implications for having a huge effect on SDLC (Software Development Life Cycle). The reliability in distributed systems also is one of crucial concerns for all software vendors and end users. Software reliability models in providing reliable software are important since it makes important decisions. These models are powerful victors in the forecast, oversight and assessment of software reliability. Software reliability is categorized into three groups of user Oriented, architecture, state-based that emerge in the design stage of the software development. Big distributed systems are systems with distributed, dynamic, complex and modern computing environment. These environments incorporate several independent or grouped objects which can communicate and cooperate with one another in order to carry out various operations. In this paper we explained various issues of reliability in different computing environments such as Cloud, Grid and SOA. It is highly important in defining robust/reliable and fault resilient architectures before way actual commencing of performing its development. The purpose of this paper is to study technical approaches in analyzing and predicting the dependency of large-scale distributed systems. How these models can handle the different determinant aspects of reliability and introduce their own experimental solutions/models? The common goal of all these models is to provide a robust environment to end users either by in advance reliability forecast at design phase or the provision of a comprehensive failure resistant system which in turn permits the applications to carry on their operation fairly under occurrence of faults. We studied these models by testing their reliability as well as other various influencing factors over way reliability of whole system and pointed out that way present models possess brilliant areas and finally outlined diversified models for betterment of system reliability. Regarding the reliability computations, Methods and models of various factors can roughly be classified into User Oriented, Architecture, State based models. These models suggested different solutions on the basis of their experiments based on either precedent data of similar applications or on some simulation results or on building some testing environment similar to real environment to test their solutions. Because of Internet indefinite status, companies require that the reliability of business-scale distributed systems must be assessed and forecast continuously. At the early stage of any operating distributed system, certain factors such as User Load, CPU Load, and Network Traffic have significantly less effect on the overall reliability of system; however it poses increasingly needs for comprehensive failure resistant technique.

Keywords: Live Circumstances, Technical Methods, Distributed Systems, Network Traffic, Large-Scale, Reliability.

1. Introduction

Distribution is the key for the manipulation of very large data sets. Distribution is necessary (but not enough) to bring scalability that is; way preserving way stable performance of steadily growing data collections by adding new resources to the system. However, distribution brings a number of technical challenges which make way design and application of distributed storage, index marking and computation delicate. A determinant consideration is way failure risk. In an environment which includes hundreds or thousands of computers (a usual setting seen in large Web companies), it is very common to face with the failure of components network,
A distributed system is a set of software applications which are engaged to coordinate the operations of several computers. This coordination is obtained by message exchange, i.e., pieces of data accompanying information. The system relies on a network which connects way computers and handles the routing of messages. Resource sharing in large scale distributed systems may fuel various failures and the coordination among a variety of resources in the distributed environment have become intricate due to the heterogeneous nature of computing, network and data resources [45,44] As failure detection methods developed for current distributed systems have generally not been considered as suitable for large-scale, heterogeneous, dynamic grid systems, the analysis of these models showed that reliability can be guaranteed either by in advance reliability production at architectural level to analyze the system better before the actual implementation of development and by providing a failure resistant system to ensure fault-free environment for end users in case if any unanticipated or unpredicted event which may occur during the execution in real-world environments.

We have explored that, above failure resistant techniques are based on various replication strategies such load balancer and data replications for SOA based distributed systems, most of these techniques simply redirect the client requests to backup servers in case of any potential failure, however for real time distributed systems dealing with financial transactions over the Internet such as online banking, e-business, and so on, we still have some open issues such as maintaining service states which are still open to be solved where sometimes ongoing processing requests are not recover after failure occurrence. Still a large amount of research is needed to deal with such updates where updates process is triggered when failure had occurred or when various updates are issued simultaneously and the situations in which are network problems may fuel system failure even in modern networks. Incorporating versatility of technologies and configurations cannot guarantee that messages will be handled through a timely manner or they are not subject to occasional failure [44]. It is noted that services are sometimes hosted by some other organization [3], hence it is necessary to provide a strong layer of fault detection and restoration in case of failure of these services and where it is impossible to keep a transaction log for cascading service calls. Several models have discussed keeping transaction logs on backup servers to recover systems from the same point of failure which provides seamless interaction to end users, afterwards these techniques have their own overheads of keeping logs, modifying Linux kernel programs or apache web server’s modules and utilizing various backup servers logging every single transaction either among services or between client and server. However, in case of a remote web service, existing failure resistant methods have tackled various failures by replicating the same set of services on different geographically segregated servers. For guaranteeing reliability in grid systems various failure resistant techniques have been proposed focusing on various functional areas such as software, hardware, workflow services, networks, etc., of grid systems. Nevertheless still few areas need more attention for making a grid system to be fault tolerant. Too much effort has been spent in metrics development for grid networks. The prediction of reliability of business-scale distributed systems at early stages not only reduces re-engineering cost, time, and effort but also it helps in making systems reliable. And therefore it is highly required to test the reliability of all factors influencing overall system reliability; however various available techniques did not discuss all determinant factors collectively. As reliability can be considered from the viewpoint of probability [40] i.e.; it can be termed as random phenomenon so and challenging to measure or forecast reliability of any scalable, heterogeneous distributed system in a precise and accurate manner is really a demanding task and because of unpredictable Internet, companies dictate that reliability of these systems must be assessed and forecast continuously.

However it is not intricate to provide adequate reliability measures for any highly reliable large scale business-scale system. Models produced in this context tried to focus on the certain factors which effect reliability such as CPU process time, network latency, interaction with external services in service coordination, etc., however at early stage of any operational distributed system, certain factors such as CPU Load, User Load and Network Traffic have significantly lower impact on the overall reliability of system, which continuously increases.
Cloud computing is the delivery of computing as a service rather than a product, whereby shared resources, software, and information are provided to computers and other devices as a utility (like the electricity grid) over a network (typically the Internet).

Cloud computing also leverages concepts from utility computing in order to provide metrics for the services used. Such metrics are at the core of the public Cloud pay-per-use models. In addition, measured services are an essential part of the feedback loops in autonomic computing, allowing services to scale on-demand and to perform automatic failure recovery.

Grid resources are accessible worldwide without geographical or boundary limitations so they have prospective chance to replace for their counterpart local peers due to their reliability and geographical dispersion. In grid structures, a coherent form of collaboration is possible through efficient high volumes of data processing using data clustering techniques for any potential purpose.

SOA services have self-describing interfaces in platform-independent Web page documents. SOA services communicate with messages formally defined via XML Schema (also called XSD). Communication among end-users and providers services typically happens in heterogeneous environments, with little or no knowledge about the provider. Messages among services can be viewed as key business documents processed in an enterprise.

SOA services are maintained in the enterprise by a registry that acts as a directory listing. Applications can look up the services in the registry and invoke the service. Universal Description, Definition, and Integration (UDDI) are the standard used for service registry. Each SOA service has a quality of service (QoS) associated with it. Some of the key QoS elements are security requirements, such as authentication and authorization, reliable messaging, and policies regarding who can invoke services.

2. Research Literature

Large-scale and distributed systems are extending due to their high flexibility, easy implementation, and high independence. However, such systems suffer from node failures because P2P nodes might leave the system arbitrarily, and sensor nodes may suddenly die from running out of energy or being attacked. Hence, providing reliable services is a very crucial and challenging problem for those systems. For a large-scale distributed system, often it is not cost effective to protect all the nodes. Moreover, the nodes of a distributed system are usually not equally important considering the reliability. In contrast, if 4% of carefully selected nodes be put aside from the network, the overlay is fragmented into hundreds of pieces.

The reliability of a system depends on many aspects, such as the resilience and stability of network topology, the availability of the data and services, and the efficiency of content distribution and transmission among others. The dynamic and self-organizing nature of large-scale distributed systems brings more challenges to this context. Over the past few years, a plenty of methods and techniques have been suggested for modeling, monitoring and detection of failures [48, 49, 50], and various plans have been developed to improve the availability and the system resiliency [51, 52, 53, 54, 55] to provide reliable services. D. Dumitriu et al [51] sorted the patterns of Denial-of-Service attacks and observed the node behavior in P2P file sharing systems. Rong Ding et al. presented a heuristic method on the basis of classifying large-scale measurement systems [57]. Waseem Ahmed and Yong Wei Wu have analyzed in detail the existing reliability methodologies from the...
viewpoint of examining reliability of individual component and explained why we still need a comprehensive reliability model for applications running in distributed systems. They have described detailed technical overviews of research done in recent years in analyzing and predicting the reliability of large scale distributed applications in the fore-mentioned four parts [58].

Yi Luo and D. Manivannann addressed this important and timely issue and suggested a selective Pessimistic message logging protocol and scalable group-based Hybrid Optimistic check pointing protocol. Similar to performance evaluation indices, their protocol takes a balanced approach to lower the overhead of check pointing and message logging and enhances scalability [59]. Misbehaving "super-nodes" and power-law network topology are regarded as the contributing factors in the resilience of such systems. Rather than identifying vulnerable peers, they propose counter-strategies to provide immunity against the attacks which unfortunately sacrifices performance in the absence of such attacks. In [48, 49], noble techniques have been proposed for network anomaly detection and traffic measurement. This pool of approaches mainly focuses on the network anomaly and dynamic failure and does not benefit normal cases. Some other recently proposed algorithms also addressed to the reliability problems of overlay networks [51, 52, 53, 54], but they treated all nodes equally without distinguishing the “critical” nodes from the “ordinary” nodes. Indeed, critical nodes do exist in all sorts of distributed systems, such as critical nodes in service, data, resources [51], and topology. They are extremely critical for the reliability of network systems. With significant progress in the literature of distributed environments in recent decades especially by not preparing case-specific software/hardware services to different kinds of users, the computing environment with reference to data sharing, cycle sharing and other modes of interaction which involve distributed resources has almost been changed. With the aggregated number of online services or automotive end-to-end solutions and the growing number of users using these services, requirements for highly reliable systems have become inevitable. Various surveys in the highlighting or description of reliability significance in business-scale systems have already been produced [55, 56] discussing several failure resilient techniques such as recovery, replication, etc. However achieving highly reliable business-scale distributed systems is still a challenge for vendors. Now it is the appropriate time to highlight missing parts in previous papers for making highly reliable business-scale systems. In this paper we surveyed significant areas particularly the role reliability of individual components which would need researchers’ attention in producing a generic model for the prediction or measurement reliability of such systems and also we tried to summarize various models produced in this regard and discussed their merits and demerits. As far as concerned with reliability, efficiency, flexibility and extensibility users want highly reliable systems, because these software/hardware services have been always required to be efficient enough to compete with the ongoing user demands in terms of QoS and in formation of data volume and unreliable systems not only do shorten the age of application software but also become a cause of massive loss regarding both time and money for applications venders, too. With the emergence of web based end-to-end applications, competition among organizations to introduce better and reliable solutions drastically have been increased. Customers with unreliable applications not only lose their users but also have to bear huge losses in their businesses. Worth noting that customers want their applications to be reliable all the time even with growing number of concurrent requests or massively increase in data. These evolutionary pressures have generated new requirements for companies to devise intelligent mechanism in the analysis and prediction of the reliability of distributed systems at certain level before the applications are actually deployed or before the application start behaving abnormal. This will certainly reduce re-engineering cost and will save unexpected time by producing more reliable systems. Except for cost and time, customer’s experience in terms of system’s reliability have also become one of the major concerns for companies, and they are investing lots of efforts to keep reliability of their applications up to the mark. We have analyzed several models for prediction or measurement reliability of distributed systems that can roughly be classified into User Oriented, Architectural or 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, nonetheless, they have certain shortcomings in their approaches. They have considered the reliability of some important factors like the reliability of hardware fixed. Furthermore failure data which they have collected through in-house testing for their experiments cannot be compared with failures that can occur under actual operational environment. We have later discussed these models in terms of certain reliability factors and this paper postulates that fixing any of these factors at a constant value will affect reliability of the whole system, so we need to consider all factors for determining the system reliability in a
3. Reliability and Factors affecting reliability

We have defined some benchmark concepts and various important issues and noticeable challenges over reliability in distributed, grid and Cloud computing and Service Oriented Architectural environments. In this section, Inspired by the Bath Tub Curve we have highlighted certain factors affecting the reliability and assessed the criticality of hardware reliability. Reliability in distributed systems also is one of major concerns for all software providers and end users. Among other factors, some factors such as software extensibility, maintainability, and applicability, 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:

- Perform well in the pre-determined time without undergoing any halt state.
- Perform exactly in a way that is designed i.e. as required.
- Resist against various failures and recovered in case of any failure which may occur during system execution without producing any negative consequences.
- Successfully run software operations or its intended functions for a postulated period of time in way specified environment.
- Too have a functional unit for performing its required function for a specified time interval under postulated conditions.
- Have the ability of correct execution even after scaling up or down has been completed considering some aspects.

So it is crucial to consider all above-mentioned cases in predicting reliability or providing a strong layer of tolerance when failure is seen during live execution of any software application. Reliability in the software engineering can be defined as the possibility of successful software running or its intended functions for a specified period of time in a specified environment. For measuring and forecast reliability of large scale distributed systems it is required to fully scrutinize fundamentals of reliability. According to the above definition of reliability the following four distinct dimensions have been proposed: [39].

- Probability: System will meet the pre-specified probability of success, at a predefined statistical confidence level.
- Intended function: If a system is not providing the proposed functionality level without having any single failure, it will be termed as a non-reliable system too.
- Time dependent: System will perform well without having any potential failure during the specified time or before another specified time t.
- Specific conditions: System will work fine under pre-specified conditions or in certain environments considering the hardware or communication devices.

Existing articles [1–3, 25, 27] have produced these elements as a central theme of their models. Forecast reliability of any software application varies from one environment to another depending upon its kind or status. Since the last few decades, the scope of business-scale applications in distributed environment have been drastically increased allowing venders to develop cross-platform individual components using different technologies which can interact transparently through a call and return mechanism. Measurement and forecast reliability of such applications is nontrivial as these systems serve various stakeholders in their businesses or social life across the globe and also they improve the QoS of various services of complex systems. Various challenges have emerged in building large scale business-scale distributed systems in different computing environments. Emergence of various computing environments has changed the applications and problems from single PC to network of computers in size and complexity. These rapid changes have not only improved the architecture of processors and network technologies but also have brought various technical challenges for application venders when considering the development of highly reliable systems in distributed environment including a synchronization, heterogeneity, scalability, fault tolerance and failure management, security, etc.

One of the key challenges in ensuring reliability of any business-scale level distributed applications is to understand the variety of underlying technologies which in turn can impact the reliability of such applications.
Various technologies exist for building business-scale distributed systems among which we can for instance point to IP technologies, Web technologies, DB or Web servers, file servers, authorization, authentication servers, clock synchronization, locking services, etc. Various factors and their characteristic importance of reliability, it is apparent that failure of any item can result in application-level errors such as inconsistency, denial of services, etc. Therefore it is necessary to consider every aspect in order to measure or forecast the reliability of these applications and they must not ignore or consider a constant assumption as a full standard or reliably criterion. Performance, scalability, and reliability are important attributes of any business-scale distributed system [35], and to achieve better performance, scalability, availability and reliability the vendors have engaged various techniques in different stages of Software Development Life Cycle (SDLC) covering fault or failure management. In SDLC following four fault life-cycle techniques have been observed [38].

- Fault prevention: efforts in eliminating errors during the development phase.
- Fault removal: eliminating bugs or faults after completing multiple testing phases.
- Fault tolerance: To provide service consonance in the definition despite the faults, by making redundancy checks.
- Fault/failure prediction: Forecast or estimation faults at architectural level during design stage or before actual implementation.

The aim of all types of fault blocking life-cycle techniques is to reinforce system reliability where the former two types are typical fault life-cycle methods taken in nearly all sorts of software applications during SDLC; However fault withstand and fault/failure prediction have gained significant consideration about the reliability attributes both in industry and academia. To provide the end users with a seamless interaction, application vendors have introduced diversified fault tolerance and recovery methods to address to the certain hidden problems which can affect the system’s operations actually, such as [5,6,8–14]. These techniques want to prevent the activation of dormant software faults, and to recover software operations from erroneous conditions by methodically resisting against system-level failures [38]. They force systems to be designed and developed as failure resistant systems i.e. they can detect faults, block them, and continue working without any information loss while in the repairing stage of faults in off line mode. However, some critical and real time systems dealing with sensitive data cannot be overhauled even for a single moment, so it is nontrivial to forecast or assess reliability of such systems well on time by testing the reliability of individual elements of any business-scale application before the prediction or measurement reliability of whole system. Potential factors are influential over reliability of any individual component or application as mentioned in [37, 42].

The details of various factors such as improper testing, Operational errors, lack of a coherent processes in quality assurance, issues of managerial changes and low-quality source codes can be analyzed in different phases of Software Development Life Cycle (SDLC) as this infrastructure provides a formal basis for control, so that risks whether in requirements, performance, or implementation phases could be determined and resolved timely. While some of factors for instance testing of environment, hardware (i.e. Processors, output/storage devices, telecommunication/input, program workload, interactions with external services have significant effects in the prediction or measurement reliability of distributed systems already at design phase or before the actual deployment of applications in real environment. And they can be used to measure or analyze the software reliability by considering them along with other essential parameters such as operational profile [2], recent data failure [3] in detail. Missing any of these factors or considering their effect as fixed can culminate at vague results. Software reliability models usually make several common hypotheses [38], such as.

- The testing environment where reliability is to be measured or foreseen is the same as the operational environment in which the reliability model will be parameterized.
- Failure data accumulated in the in-house testing supposedly present failures of actual operational environment.
- Hardware Reliability is considered to be constant although the failure of hardware devices may cause certain contingent drivers to hang or crash in a distributed environment [36] which ultimately reduces the reliability of running applications.

Proper functioning of hardware beside the software is an important consideration for millions of users of these systems. Characteristics of hardware failure represented by the famous curve i.e. Bath Tub Curve is shown in Fig. 1. As in this curve, the chance of applying hardware failure approaches to as many as possible in the initial and end life of any specific hardware. At initial stage, failures can occur because of any manufacturing fault where in the final stage useful life degradation of component characteristics will cause hardware modules to fail. Based on such assumption models produced to predict reliability of distributed systems may not give
precise and cute results showed that reliability of software applications is highly contingent to these factors if we suppose them as constant, analogous or congruent to testing environment without any need to consider the hardware life can direct us towards unrealistic results [37].

Fig. 1. An example from Bath Tub Curve showing the characteristics of hardware failure. [37]

Table 1 shows most important of top 10 researches done so far as well as the benefits, features, objections, weak points and improvements to be done in these researches.

<table>
<thead>
<tr>
<th>Authors</th>
<th>Objections/Weaknesses</th>
<th>Research done/Features/Benefits</th>
<th>Improvements to be done</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z. Zibin, R.L. Michael 2010</td>
<td>Sometime services are hosted by some other organizations</td>
<td>In this study the reliability prediction of service-oriented systems is done Collaboratively</td>
<td>Therefore it is necessary to provide a strong layer of fault detection and recovery in case of failure occurrence for these services and where it is impossible to maintain transaction log for cascading service calls.</td>
</tr>
<tr>
<td>Vittorio Cortellessa, Vincenzo Grassi 2007</td>
<td>Insufficient testing, Operational errors, Lack of coherent process for quality assurance, managerial changes and poor source code</td>
<td>Reliability modeling and analysis of service-oriented architectures. Various factors such as Insufficient testing, Operation error, Lack of coherent process for quality assurance, Issues of changing management and managerial changes and poor source code can be analyzed in detail diversified stages of Software Development Life Cycle (SDLC)</td>
<td>Some factors of instance environment, hardware, program workload, interaction testing with external services have significant effects on the prediction or measurement reliability of distributed systems started formerly at design stage or just before the actual deployment of applications in the real environment</td>
</tr>
<tr>
<td>C. Leslie, et al. 2008</td>
<td>There are various models in this context, despite their given deficiencies in approach.</td>
<td>Early prediction of software component reliability. They have fixed the reliability of some important factors like reliability of hardware at a constant value.</td>
<td>Failure data which they have collected for their experiments failures of in-house testing cannot be compared with those may occur under actual operational environment</td>
</tr>
<tr>
<td>Jim Lau, Lau Cheuk Lung, J. da Fraga, G.S.</td>
<td>In-execution Requests were effectively dropped out and no</td>
<td>Designing failure resistant web services by the engagement of BPEL. Fault/failure prediction have become popular as reliability attributes in</td>
<td>A seamless interaction to the end users, package vendors have introduced various fault tolerance and recovery techniques to get along with</td>
</tr>
<tr>
<td>Authors</td>
<td>Objections/ Weaknesses</td>
<td>Research done/ Features/Benefits</td>
<td>Improvements to be done</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------</td>
<td>----------------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>Veronese 2008</td>
<td>mechanism was defined to recover those transactions into seamless ones</td>
<td>industry and academia</td>
<td>certain hidden issues which may affect system’s operations in reality</td>
</tr>
<tr>
<td>A. Immonen, E. Niemel 2008</td>
<td>---</td>
<td>The study Survey of reliability and availability prediction methods from the viewpoint of software architecture</td>
<td>---</td>
</tr>
<tr>
<td>S.O. Olabiyisi, et al 2010</td>
<td>To be more costly and less successful</td>
<td>A performance survey of evaluation models for distributed software system architecture</td>
<td>S.O. Olabiyisi, et al 2010</td>
</tr>
<tr>
<td>M.R. Lyu 2007</td>
<td>In SDLC there following four fault life-cycle techniques exist -To be more costly and less successful</td>
<td>Software reliability engineering: A roadmap to achieve performance, scalability, availability and reliability. Software providers have used various techniques in different phases of Software Development Life Cycle (SDLC) such as fault or failure management.</td>
<td>---</td>
</tr>
<tr>
<td>Yi Luo and D. Manivanna 2012</td>
<td>---</td>
<td>Propose a scalable group-based Hybrid Optimistic check pointing and selective Pessimistic message logging (HOPE) protocol.</td>
<td>Case - dependent check pointing and recovery algorithms are scalable</td>
</tr>
<tr>
<td>Y. Lin, B. Liang, and B. Li 2007</td>
<td>To be more costly and less successful</td>
<td>Data Persistence in Large-scale Sensor Networks with Decentralized Fountain Codes. Methods and techniques are proposed for failure monitoring, modeling, detection, and various plans have been developed to improve the overall system availability and resiliency to provide reliable services.</td>
<td>---</td>
</tr>
<tr>
<td>T. Q. Qiu, E. Chan, and G. Chen 2007</td>
<td>Highly reliable business-scale distributed systems are still a challenge to be achieved by venders</td>
<td>Iterative Detection and Proactive Recovery</td>
<td>---</td>
</tr>
<tr>
<td>R. Ding, X. Li, T. Zhu, Z. Zhong, J. Zhang 2011</td>
<td>---</td>
<td>presents an intelligent method on the basis of classifying large-scale measurement systems</td>
<td>---</td>
</tr>
</tbody>
</table>
Table 1

<table>
<thead>
<tr>
<th>Authors</th>
<th>Objections/ Weaknesses</th>
<th>Research done/ Features/Benefits</th>
<th>Improvements to be done</th>
</tr>
</thead>
<tbody>
<tr>
<td>W. Ahmed, Y. Wu. 2013</td>
<td>--</td>
<td>Analyzed and highlighted the significance of underlying factors determinant in the reliability of distributed computing systems.</td>
<td>Discussed numerous strategies which addressed to the reliability prediction models in different phases of SDLC and Discussed numerous strategies which deal with reliability prediction models in different stages of SDLC.</td>
</tr>
</tbody>
</table>

4. Techniques

The aim of all types of fault life cycle techniques [38] is to make systems reliable, so in this section we have evaluated fault tolerance and fault prediction techniques further in the background of various models and pointed out how these models have used various methodologies to make distributed system more reliable. User Oriented, Architectural and State based approaches are used in accordance with the above mentioned reliability definitions.

4.1. Failure Resistant Techniques

For distributed systems a fault tolerance mechanism operates as a backbone and is one of the important types of fault life-cycle techniques [38] which have important roles in the reliability of business-scale distributed systems. It deals with all those situations where failures or faults occurs during live execution of business-scale applications and provides the system with the ability to keep its function correctly. Based on various unanticipated situations in a distributed environment for instance Cloud, Grid, and Service Oriented Architecture 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 failure resistant system can be used either as spatial or temporal redundancy, including replication of hardware (with additional components), software (with special programs), and time (with diversified operations) [6]. In terms of data, Replication 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, where data consists of multimedia objects. With significant advancement in the field of distributed computing, the development of dynamic business-scale distributed systems have been drastically increased because of flexible architectures. Various failure resistant techniques have been introduced to provide a reliable and seamless view to the end user. At the below we have given an analysis of various models/strategies providing fault tolerance in different computing environments. We have also summarized possible shortcomings or open issues in these techniques. Most of these techniques in case of any failure simply redirect the client’s requests to backup servers regardless of the current executing process. Running requests are effectively dropped out and no mechanism has been defined to recover those transactions for the sake of robust 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.

4.2. Fault or Failure Prediction Techniques

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

![Graph showing the usage of various factors in different reliability prediction models.](image)

Fig.2 The Graph showing the usage of various factors in different reliability prediction models.

5. Concluding Remarks:

In this paper, we introduced different technical approaches in analyzing and forecasting the reliability of large-scale distributed systems. At first we explained some pragmatic requirements for highly reliable systems and highlighted the importance and various issues of reliability in different computing contexts. Then we clarified certain possible determinant agents and various challenges that are noticeable in highly reliable distributed systems such fault detection, recovery and removal through testing or various replication techniques. Later we exactly studied various research models which combine significant solutions in order to address the potential factors and various challenges in prediction and measurement of reliability of software systems in distributed systems. Finally at the end we discussed the constraints of existing models and suggested that future works for the forecast and analysis of reliability in distributed systems in real environment be founded on the basis of our study. Various models have been produced to assess or forecast the reliability of large scale distributed systems, but reliability issues with these systems are not resolved yet. To guarantee reliability of distributed systems per se requires the reliability test of each individual element or factor contributed in business-scale distributed systems before any forecast or assessment be completed over the whole reliability of system, and the implementation of transparent fault detection and fault recovery plan to provide the end users with robust interactions. We have precisely analyzed existing reliability methodologies regarding the reliability examination of individual components and cleared out why we should still have an extensive reliability model for applications running in distributed system.

Reference