Fault Tolerance in Distributed Systems: A Survey

Conference Paper · October 2018

DOI: 10.1109/PAIS.2018.8598484

3 authors:

Abdeldjalil Ledmi
Abbes Laghrour - Khenchela University
9 PUBLICATIONS **21** CITATIONS
SEE PROFILE

Hakim Bendjena
Université de Tébessa
59 PUBLICATIONS **231** CITATIONS
SEE PROFILE

Hemam Sofiane Mounine
Abbes Laghrour - Khenchela University
22 PUBLICATIONS **36** CITATIONS
SEE PROFILE

2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS)

Fault Tolerance in Distributed Systems: A Survey

Abdeldjalil Ledmi
Laboratory of Mathematics,
Informatics and Systems (LAMIS)
University of Larbi Tebessi,
Tebessa, 12002, ALGERIA
abdeldjalil.ledmi@yahoo.com

Hakim Bendjenna
Laboratory of Mathematics,
Informatics and Systems (LAMIS)
University of Larbi Tebessi,
Tebessa, 12002, ALGERIA
hbendjenna@gmail.com

Sofiane Mounine Hemam
Khenchela University, ICOSI Laaboratory,
Khenchela, Algeria
sofiane.hemam@gmail.com

Abstract— **Distributed systems can be homogeneous (cluster), or heterogeneous such as Grid, Cloud and P2P. Several problems can occur in these types of systems, such as quality of service (QoS), resource selection, load balancing and fault tolerance.**

Fault tolerance is a main subject regarding the design of distributed systems. When a hardware or software failure occurs in the system, it causes a failure and we call it, in this case, a fault. Moreover, in order to allow the system to continue its functionalities, even in the presence of these faults, they must find techniques, which tolerate failure; the goal of these techniques is to detect and to correct these errors.

In this paper, we introduce at first an overview of the basic concepts of distributed systems and their failures types, then we present, in a detailed manner, the different techniques that tolerate fault, used to identify and to correct faults in different kinds of systems such as: cluster, grid computing, Cloud and P2P systems.

Keywords— Fault-Tolerance Technique, distributed Systems, Cluster, Grid, Cloud, P2P.

I. INTRODUCTION

Various definitions of distributed systems have been given in the literature, according to [1] a distributed system is a collection of independent computers that appears to its users as a single coherent system.

Moreover, distributed systems are groups of interconnected nodes, which have the same goal for their work. In order to give the answer to a specific problem, the task has been divided into smaller tasks allocated in those nodes. Every node executes its portion of the task and resubmits the results to submission node. Furthermore as distributed systems can be homogeneous (cluster), heterogeneous (Grid, Cloud and P2P), they are subjected to different kinds of problems such as Quality of Service (QoS), Resource Selection, Load balancing and *Fault Tolerance*.

Fault tolerance perceives how a system responds to a surprising hardware or software failure, in comparison with uniprocessors; it is hard to detect fails in the distributed system. Primarily, fault tolerance is composed of two basic elements; failure detection and recovery.

To keep the system runs when a failure occurs or if any of its components disconnected or faulty, is a big challenge in distributed systems. There are many fault tolerance techniques in distributed paradigm such as retry, replication, check-pointing and message logging etc.

The remainder of the paper is structured as follows: In Section 2, some basic concepts of fault tolerance in distributed system are explained. In Section 3, we present the different kinds of reactive and proactive fault tolerance. A related work is presented, in detail, in Section 4. A brief conclusion is given in Section 5.

II. BASIC CONCEPTS

In order to understand the main role of fault tolerance in distribute systems, we need at first to check out what it actually means tolerate faults or being fault tolerate for a distributed system, this later is closely narrowly to ”Dependable systems”. As shown in Fig.1, Dependability is a key word, which includes a lot of useful requirements for distributed systems. In [1] , authors present the main characteristics of dependability, which are:

1) Availability: meaning that the system can be used instantly.
2) Reliability: means that the system is able to be run all the time without failure.
3) Safety: it means even if the system fails, nothing catastrophic will happen.
4) Maintainability: whenever the system fails it can be repaired in an easy and quick way and sometimes even without that the users notice that failure.

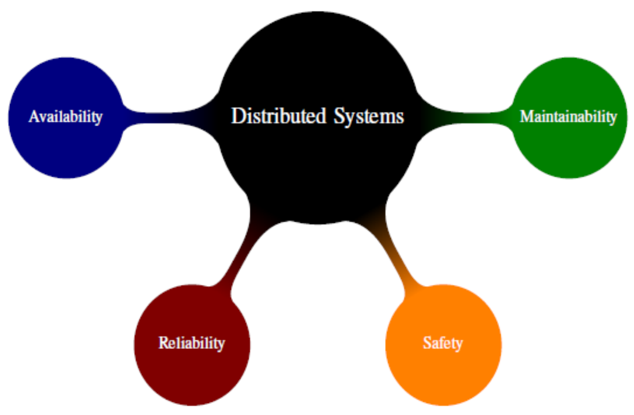

Fig. 1. Dependable distributed systems

III. FAULT TOLERANCE TECHNIQUES

As can be seen from Fig. 2 that in traditional distributed paradigms, many fault tolerance techniques are available and can be used at either task level or workflow level [2] .

A. Reactive fault tolerance

Reactive fault tolerance techniques are used to reduce the impact of failures on a system when the failures have occurred. Techniques based on this policy such as Retry, Replication, Check-pointing and Message Logging.

1) Retry: This is the most popular failure recovery technique to use, and we hope that no matter what the cause of the failures will not be encountered in the later retries [3] .

2) Replication: The basic idea of this technique is to create multiple replicas of tasks for each running task and submit them on the different hosts so that all replicated tasks do not hang (due to a host crash, the host disconnected from the network). client, etc.), the execution of the task would be succeed [3] .

3) Check-pointing: Check-pointing is the most simplest fault tolerant approach used in distributed systems; the basic idea requires that the system regularly save its state onto reliable and stable storage. After a crash, the system restarted from the last checkpoint instead of from the beginning [1] .

4) Message Logging: As check-pointing is an expensive operation, many techniques have been sought to reduce the number of checkpoints, but still enable recovery. The basic idea of this technique is that, if the transmission of messages can be replayed in a well-defined order, we can always achieve a consistent state of the system, without making a restore of that coherent state from stable storage. Instead, a check-pointed state is taken as a starting point, and all messages that have been sent since are simply retransmitted and handled accordingly [1] . In the literature, there are two approaches founded. The first approach is represented by the pessimistic logging protocols. . In contrast, in an optimistic logging protocol.

B. Proactive Fault Tolerance

Proactive Fault Tolerance Proactive fault tolerance expects the faults proactively and places healthy (working) components in place of the faulty components, to prevent recovery from faults and errors. Many techniques follow this policy such as Preemptive migration, software rejuvenation Load Balancing etc.

1) Software Rejuvenation: with each reboot of the system a planned change on the state have been taken.

2) Self-healing: the basic idea is to make an auto-control of the failure of an application instance running on multiple virtual machines.

3) Preemptive Migration: is a technique which allows to observe and analyze an application constantly. Fault tolerance taxonomy

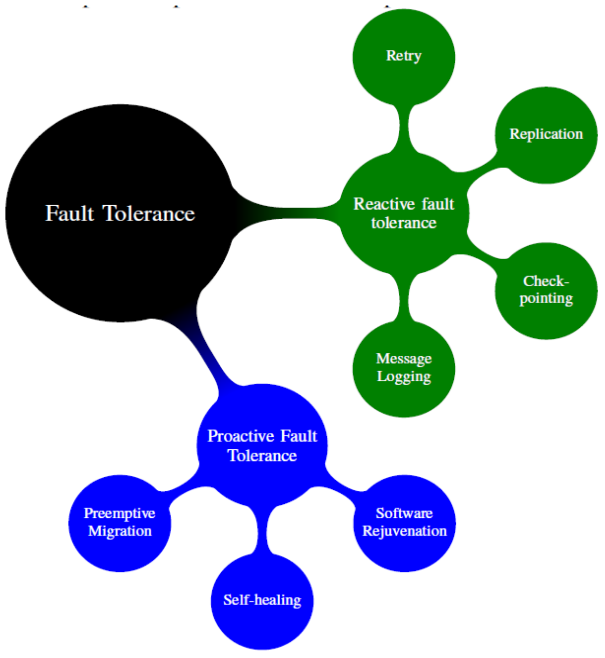

Fig. 2. Fault tolerance taxonomy

IV. LITERATURE REVIEW

we extend our survey from the works of [3] and we add some columns such as large-scale, load-balancing, availability, and reliability.

LA-MPI [4] the Los Alamos Message Passing Interface, is an implementation performed on the Message Passing Interface (MPI) for Tera scale clusters. Motivated by the need of additional software to ensure end-to-end reliability in a high performance message-passing system. LA-MPI uses the retry mechanism as the fault tolerant technique.

Co-Check-MPI [5] was the first MPI implementation based on the Condor library for implementation of fault tolerance using the technique of check pointing in MPI, focus on the applications failure. The disadvantage of this technique is that a bigger application would take more time in check-pointing.

In [6] , the authors propose new technique in proactive policy for fault tolerance in MPI applications based on task migration and load balancing capabilities of Charm++ and AMPI,

TABLE I. COMPARED FAULT DETECTION AND TOLERANCE TECHNIQUES USED IN DISTRIBUTED SYSTEMS

Source	**Platform**	**Techniques used**	**Faults detected**	**Proactive or Reactive Mode**	**Adapt Large Scale**	**Support Load Balancing**	**being available**	**being reliable**	**Recovery technique used**	**General comment**
LA-MPI [4]	Cluster	Checks Unacknowledged List	Network related failure	Reactive	✓	✗	✗	✓	Sender side retransmission	Function only on low error rate environments
Co-Check MPI [5]	Cluster	Application failure	Application Failure	Reactive	✗	✗	✗	✗	Check-pointing	Problem with the biggest applications, more time in saving its state
Proactive in MPI [6]	Cluster	Process failure	Process Stop Crash	Proactive	✗	✓	✗	✗	Replication	That deals only with predictable fault
Globus [7]	Grid	Heartbeat Monitor (HBM)	Host, Network failure	Reactive	✓	✗	✗	✗	Checkpoint, Resubmit the failed job	Cant handle user, defined exceptions
Globus [8]	Grid	New modular for Heartbeat Monitor (HBM)	Host	Reactive	✗	✗	✗	✗	Checkpoint	Only detecting the fault failure recovery mechanism not supported
Grid-WFS and FDS [9]	Grid	Heartbeat Monitor (HBM), message Notification	Host, Network failure	Reactive	✓	✗	✗	✗	Checkpoint, Redundancy, alternative Task	handle user, defined exceptions but very expensive
Assure [10]	Cloud	replica (shadow) of the application	Host, Network Failure	Reactive/ Proactive	✓	✗	✗	✗	Check pointing, Retry, Self-Healing	/
SHelp [6]	Cloud	Not mentioned	Application Failure	Reactive	✓	✗	✗	✗	Check pointing	/
FTMC [11]	Cloud	Not mentioned	VMs Machines	Reactive	✓	✗	✗	✓	Check pointing	No load balancing and Availability
FTM [9]	Cloud	heartbeat message exchange protocol	VMs Machines	Reactive	✓	✗	✗	✓	replicating users applications	No load balancing and Availability
[12] Behera and Tripathy, 2014	Mobile Grid Computing Systems	Heartbeat Monitor (HBM)	Host Failure	Reactive	✓	✓	✓	✓	Replication	No load balancing
[13] Ledmi, Hakim and Hemam , 2018	Volunteer Computing System	Heartbeat Monitor (HBM)	Host Failure	Reactive/ Proactive	✓	✓	✓	✗	Resubmit job	No reliability treating

the basic idea in this technique is, when a fault is imminent and before a crash happens the system proactively attempts to migrate execution off that processor using processor Virtualization.

Globus is a community of developers and users who collaborate with each other in the use and development of open source software, and documentations for distributed computing and resource federation [14] . The software itself called Globus Toolkit. [7] have proposed the fault tolerant system according to one grid middleware (Globus toolkit). For that, they propose the elimination of the single point failure, and make concept's ability to be integrated with a variety of grid middleware.

[8] propose a fault detection service designed to be incorporated, in a modular fashion, into distributed computing systems, tools, or applications.

[9] use a generic failure detection service named (FDS), this service is able to detect both the task crashes and the user-defined exceptions through a heartbeat monitoring, and based

on the Globus mechanisms, as a flexible failure handling framework (Grid-WFS). A Custom user-defined Exception Handling technique (This technique allows users to specifically handle the failure of a particular task), and an alternative Task as opposed to the retry technique.

ASSURE, Automatic Software Self-healing Using Rescue points introduces rescue points, a new software self healing technique for detecting, tolerating and recovering from software faults in server applications. In [15] , they use the existing quality assurance testing techniques to generate known bad inputs to an application, in order to identify candidate rescue points.

[16] state an insignificant runtime system Shelp which is able to make software faults survive for server applications running in a virtual machine environment. Shelp can be regarded as an extension of Assure to a virtualized computing environment. In order to make the system bypass the faulty path Yet two new techniques, weighted rescue points and two level storage hierarchy for rescue were introduced.

[11] offered a system that deals with the fault tolerance mechanism. In this system, a model called fault tolerance for cloud (FTMC) which depends on the reliability evaluation of each computing nodes going by the name of a virtual machine (VM) in cloud environment and fault tolerance of real time applications running on those VMs. A virtual machine is chosen for computation due to its reliability and can be taken away, if it lacks performance in real time applications.

[17] Suggested the FTM to beat the restriction of existing methodologies of the on-demand service. In order to release the reliability and resilience, an alternative perspective on creating and manipulating fault tolerance was suggested. The user is able to specify and apply the needed level of fault tolerance by this specific methodology without asking for any knowledge about its implementation. FTM architecture this can primarily be viewed as an assemblage of several web services components, each with a specific functionality.

[18] a step was taken going on the achievement of a large scale for peer to peer ,harvesting redundancy covered the networks. It was shown how to make routing tables. Though the presence of faulty nodes they can localize the network. Those approaches were made to the specific instances of Tapestry and Pastry; they seem to be about applicable and could upgrade other systems.

[10] have suggested a discovery system's resource in order to be used in peer to peer networks. The resource is planted on a setup of multiple chord rings. By the exposure of the systems performance and fault tolerance, we show that there is a tradeoff between the two elements. For instance, a better performance can be leaded by a smaller ring while the fault tolerance is increased by a larger ring.

In [12] The authors attracts the problems of load balancing, fault tolerance and the reliability for mobile grid environment, and propose two algorithms: one for de-centralized load balancing and the second for fault tolerance. The proposed algorithms are found better in time complexity and efficiency compared to the existing works. For finding the availability and mean time to failure, the authors also proposed a method for modeling the reliability of mobile grid. But the method is still needs to be improved.

In [13] , to achieve a good performance in the volunteer computing system (load balancing between volunteer resources) and improving the user requirements (time response and availability) in the context of volunteer computing platform, for that the authors propose four algorithms. Two algorithms are used to balance the load in two deferent way. Decentralized way between volunteer clouds and the centralized way between the volunteer resources of each volunteer cloud. Another algorithm is used to satisfy the user requirement (time response and availability). Moreover, they applied an algorithm to estimate the failure probability of the volunteer resources by using the Markov chain model.

In order of the refinement of the analysis of these different research works regarding fault tolerance in different platform, we propose a summarizing comparative table (see Table 1). Moreover, to clarify more the study we divided the table into 4 different platforms: we began with the oldest one the Cluster, the grid, the cloud and the volunteer computing system. As shown, we note that most of these works focus more on the reactive mode techniques then the proactive mode to detect the failure in a system, each work use methods to detect the failure, and how to tolerate the fault (ex: Replication, Resubmit jobs, reallocation …).

V. CONCLUSION

One of the important topics in distributed systems: Fault Tolerance, as well as it is considered to be a significant theme within distributed systems design. Basically. Fault tolerance is known as the capacity where a system is able to function in the presence of failure.

As presented in this paper, the majority of distributed systems (Cluster, Grids, Cloud and P2P) do not manage the load balancing, the availability and reliability, and focus only on Check-pointing technique, which is expensive, and consume more hardware for its implementation, most of this compared works are reactive to detect the fault and to repair the system. Finally, we aim to suggest an architecture in P2P system to deal with the load balancing problem by using proactive and reactive techniques to identify faults as our future work.

References

[1] Tanenbaum, A.S. and M. Van Steen, *Distributed systems: principles and paradigms*. 2007: Prentice-Hall.

[2] Amin, Z., H. Singh, and N. Sethi, *Review on fault tolerance techniques in cloud computing.* International Journal of Computer Applications, 2015. **116**(18).

[3] Haider, S., et al. *Fault tolerance in distributed paradigms*. in *In2011 International Conference on Computer Communication and Management, Proc. of CSIT*. 2011.

[4] Graham, R.L., et al., *A network-failure-tolerant message-passing system for terascale clusters.* International Journal of Parallel Programming, 2003. **31**(4): p. 285-303.

[5] Stellner, G. *CoCheck: Checkpointing and process migration for MPI.* in *Parallel Processing Symposium, 1996., Proceedings of IPPS'96, The 10th International*. 1996. IEEE.

[6] Chakravorty, S., C.L. Mendes, and L.V. Kalé. *Proactive fault tolerance in MPI applications via task migration*. in *International Conference on High-Performance Computing*. 2006. Springer.

[7] Affaan, M. and M. Ansari. *Distributed fault management for computational grids*. in *Grid and Cooperative Computing, 2006. GCC 2006. Fifth International Conference*. 2006. IEEE.

[8] Stelling, P., et al., *A fault detection service for wide area distributed computations.* Cluster Computing, 1999. **2**(2): p. 117-128.

[9] Hwang, S. and C. Kesselman, *A flexible framework for fault tolerance in the grid.* Journal of Grid Computing, 2003. **1**(3): p. 251-272.

[10] Salter, J. and N. Antonopoulos, *An efficient fault tolerant approach to resource discovery in p2p networks.* 2004.

[11] Meshram, A.D., A. Sambare, and S. Zade, *Fault tolerance model for reliable cloud computing.* International Journal on Recent and Innovation Trends in Computing and Communication, 2013. **1**(7): p. 600-603.

[12] Behera, I. and C.R. Tripathy, *Performance modelling and analysis of mobile grid computing systems.* International Journal of Grid and Utility Computing, 2014. **5**(1): p. 11-20.

[13] Ledmi, A., H. Bendjenna, and H.S. Mounine, *Optimizing Both the User Requirements and the Load Balancing in the Volunteer Computing System by using Markov Chain Model.* International Journal of Enterprise Information Systems (IJEIS), 2018. **14**(1): p. 35-62.

[14] Foster, I. *Globus toolkit version 4: Software for service-oriented systems*. in *IFIP international conference on network and parallel computing*. 2005. Springer.

[15] Sidiroglou, S., et al., *Assure: automatic software self-healing using rescue points.* ACM SIGARCH Computer Architecture News, 2009. **37**(1): p. 37-48.

[16] Chen, G., et al. *Shelp: Automatic self-healing for multiple application instances in a virtual machine environment*. in *Cluster Computing (CLUSTER), 2010 IEEE International Conference on*. 2010. IEEE.

[17] Jhawar, R., V. Piuri, and M. Santambrogio. *A comprehensive conceptual system-level approach to fault tolerance in cloud computing*. in *Systems Conference (SysCon), 2012 IEEE International*. 2012. IEEE.

[18] Hildrum, K. and J. Kubiatowicz. *Asymptotically efficient approaches to fault-tolerance in peer-to-peer networks*. in *International Symposium on Distributed Computing*. 2003. Springer.