2011 International Conference on Computer Communication and Management
Proc .of CSIT vol.5 (2011) © (2011) IACSIT Press, Singapore

Fault Tolerance in Distributed Paradigms

Sajjad Haider [1], Naveed Riaz Ansari [2], Muhammad Akbar [1], Mohammad Raza Perwez [1], Khawaja MoyeezUllah Ghori[1]

[1] National University of Modern Languages, Islamabad,

[2] Shaheed Zulfiqar Ali Bhutto Institute of Science & Technology, Islamabad

Abstract. Distributed systems are responsible for providing the main execution platform for High Performance Computing (HPC). As distributed systems can be homogeneous (cluster) as well as heterogeneous (grid and cloud etc), they are prone to different kinds of problems. The issues in distributed systems can be Security, Quality of Service, Resource Selection and Fault Tolerance etc. Fault tolerance is responsible for handling the reliability and availability of distributed systems. It is not feasible to ignore job failures in distributed environments where long and persistent commitments of resources are required.

In this paper we have presented a comprehensive classification of errors, failures and faults that can be encountered in a Distributed environment. Furthermore, we have examined different fault identification and tolerance techniques available in different Clustered and Grid Computing environments. Fault detection and tolerance techniques used in homogeneous and heterogeneous environments are different from each other and are not interoperable. We have proposed in this paper that a standard fault tolerant framework should be there capable of handling all the identified errors, failures and faults.

Keywords: Fault Tolerance; Distributed Systems; Cluster, Grid

1. Introduction

Distributed systems are well known for achieving high performance in computing. We distribute compute intensive jobs into portions and send them to the machines for computations that are part of the distributed system. Nodes that are part of the distributed system execute their portion of the job and submit the results to job submission node. Distributed systems are further classified into Clusters and Grids. If we want to establish a reliable and available distributed environment then a fault-tolerant mechanism should be there. The incorporation of faults handling mechanisms in Clusters and Grids play an important role for that environment to be reliable and available. Fault tolerance is a capability developed in the system so that it could perform its function correctly even in the presence of faults. Taking fault tolerance into consideration would result in increasing the dependability of a system [1].

According to [1] failure is encountered when a system moves away from its particular behaviour. The reason behind that failure is called error that also ultimately depicts some sort of fault or defect in that system. This means that fault is the actual cause of a failure, and error is just an indication or sign of a fault. Multiple errors could be due to a fault, and even a single error could be the cause of multiple failures. In fault tolerance we try to preserve the delivery of expected services in the presence of faults that can cause errors. Errors are detected and corrected, and permanent faults are located and removed while the system continues to deliver acceptable service [2].

2. Existing Fault Tolerance Techniques

Many fault tolerance techniques such as retry, replication, message logging and check pointing [3] are available in traditional distributed paradigms.

2.1. Retry

Retry is the simplest failure recovery technique in which we hope that whatever is the cause of failures, the effect will not be encountered in subsequent retries [7].

2.2. Replication

In replication based technique we have replicas of a task running on different machines and as long as not all replicated tasks crash (i.e. host crash etc), chances are that the task execution would succeed [7].

2.3. Message Logging

In message logging all participating nodes log incoming messages to stable storage and when a failure is encountered than these message logs are used to compute a consistent global state. Algorithms that take this approach can be further classified into those that use pessimistic and those that use optimistic message logging [8]. Different types of errors, failures and faults that are likely to occur in distributed paradigms are presented in Figure 1.

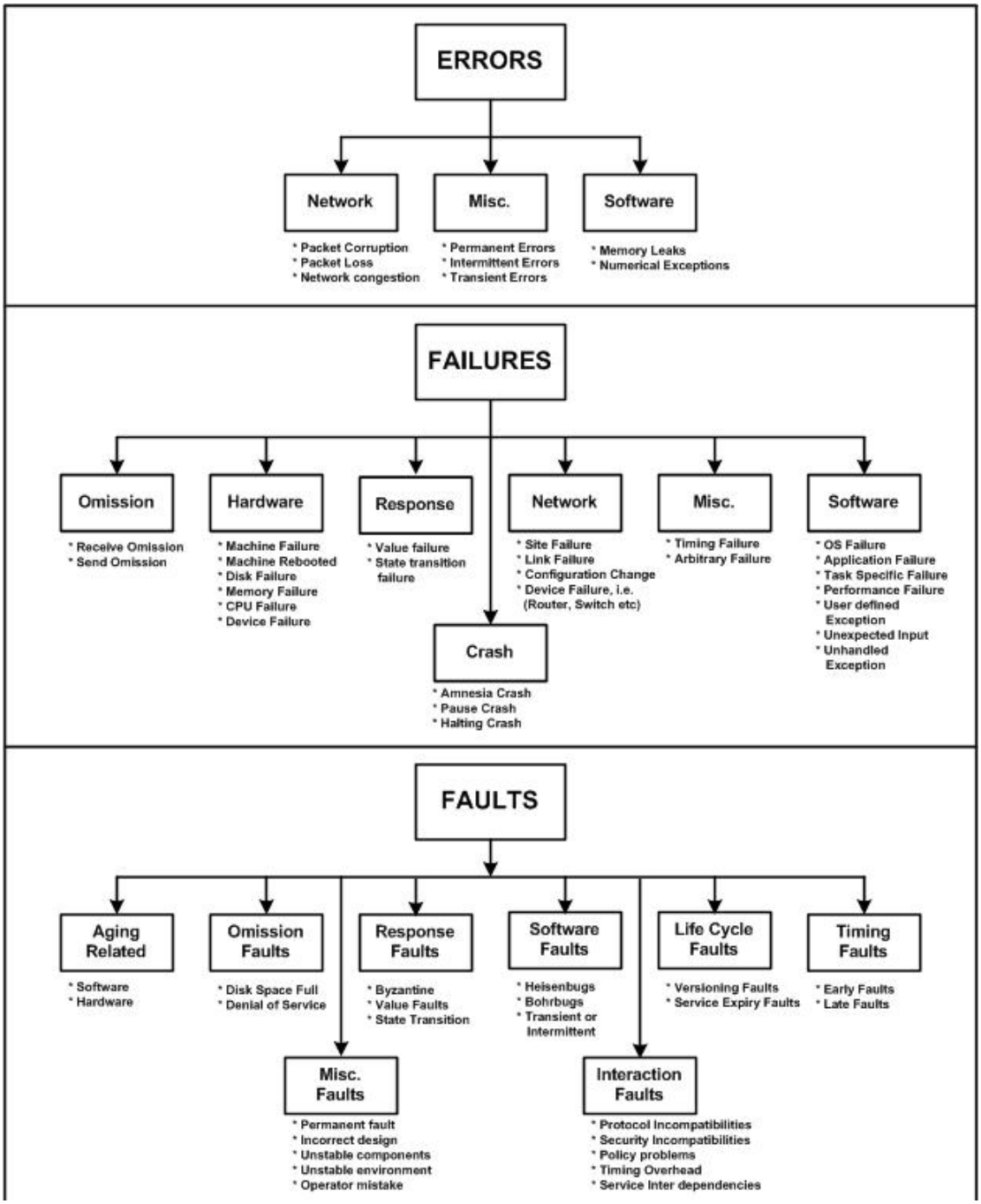

Fig. 1: Extended classification of errors, failures and faults [2, 3, and 4]

2.4. Check-pointing

Check-pointing is relatively more popular fault tolerant approach used in distributed systems, where the state of the application is stored periodically on reliable and stable storage, normally a hard disk etc. In case

of problem during execution, i.e. after crash etc, the application is restarted from the last checkpoint rather than from the beginning [9].

3. Literature Review

Globus offers a software infrastructure that enables applications to handle distributed heterogeneous computing resources as a single virtual machine [4]. The Globus Heart Beat Monitor [7, 10] uses a generic failure detection service that enables applications to detect both host or network failure through a process of heartbeat missing, e.g. 'the task died without un-registering' notification message.

LA-MPI [11, 12] is a type of implementation performed on the Message Passing Interface (MPI) [13, 14] motivated by the increasing need for fault tolerance at the software level in bigger High Performance Computing (HPC) scenarios. LA-MPI uses the retry/retransmission mechanism as the fault tolerant technique [15].

FT-MPI is another try of handling problems in MPI. Current fault tolerance and recovery strategies in FT-MPI [16] can regularly take checkpoints during a workflow step, that is normally a scientific application, and when a failure is encountered, the application is restarted from the last checkpoint [17].

Legion offers software based infrastructure for a system of heterogeneous and geographically distributed machines to interact with each other. Legion uses 'pinging and timeout' mechanism to detect task failures [5]. There is no way of detecting problems in Legion like the task crash failures, user-defined exceptions. Similarly Legion can't distinguish the pure task crash failure from host/network failures [23].

Condor-G [18] uses an 'ad hoc' failure detection mechanism and uses 'periodic polling to generic Grid server' to detect some specific types of failures e.g. host and or network failures. Condor-G can't detect task crash failures or user-defined exceptions, as is the case in Legion. Condor-G uses Retry on the same machine for fault tolerance in Grid environment [7].

NetSolve [3] is a client/server application designed to solve computational science problems in a distributed environment. The Netsolve system is composed of loosely coupled distributed systems, connected via LAN or WAN. Netsolve uses a generic hearbeat mechanism for failure detection and uses Retry on another available machine for fault tolerance [7].

Mentat uses polling for its failure detection and uses Replication as a fault tolerance mechanism [7].

CoG Kits [7] does not have failure detection mechanism and missing the advanced features of fault tolerance such as replication and check-pointing etc.

4. Critical Evaluation

For performing critical evaluation, we have compared different Grid Fault Tolerance implementations with each other and found some advantages and disadvantages of those techniques. Same technique was performed for Clustered based Fault Tolerant implementation scenarios.

4.1. Grid based Fault Tolerant implementations

Globus [5] uses the generic failure detection service that can be incorporated into a distributed system, tool or application. Faults detected by Globus are Host and Network failure. Globus can't handle user defined exceptions. Similarly only application level fault tolerance is provided in this environment.

The features offered by Legion are transparent scheduling, data management, fault tolerance, site autonomy and security [11, 14]. Legion uses pinging and timeout technique for detecting faults and doesn't have capability of detecting and handling task crash failures or host/network failures. Legion uses checkpoint recovery as fault tolerance technique [6].

NetSolve [3] is a programming and runtime system for accessing high-performance libraries and resources transparently. Netsolve uses a generic hearbeat mechanism for failure detection and uses Retry on another available machine for fault tolerance [9]. NetSolve uses generic heartbeat mechanism as its fault detection technique but doesn't support diverse failure recovery mechanisms [7].

Mentat uses polling for its failure detection and uses Replication as a fault tolerance mechanism [7].

CoG Kits [7] does not have failure detection mechanism and missing the advanced features of fault tolerance such as replication and check-pointing etc.

TABLE I summarizes fault detection and tolerance techniques used in parallel and distributed systems.

System	Type	Fault Detection Technique	Faults detected	Proactive / Reactive	Fault Tolerance Technique	Comments
Globus [5]	Grid	Heart beat monitor	Host, Network failure	Reactive	Resubmit the failed job.	Can't handle user defined exceptions
LA-MPI [11]	Cluster	Checks Unacknowledged list at specific intervals	Network related failure	Reactive	Sender side Retransmission	Appropriate only for low error rate environments
MDS-2 [7]	Grid	GRRP	Task Crash failure	Reactive	Retry	Can't handle user defined exceptions
LAM/MPI [20]	Cluster	Node / Application stops responding	Node Failure	Reactive	Replication of checkpoints	Communication increases by replication checkpoints on several machines
Co-Check-MPI [21]	Cluster	Application failure	Application Failure	Reactive	Check-pointing	Bigger application will take more time in check-pointing
Legion [23]	Grid	Pinging and Timeout	Task Failures	Reactive	Check-point recovery	Can't distinguish between task crash failure and host/network failure.
Condor-G [18]	Grid	Polling	Host Crash, Network Crash	Reactive	Retry on same machine	Use of Condor client interfaces on top of GLOBUS
NetSolve [7]	Grid	Generic Heart beat mechanism	Host crash, Network failure, k crash	Reactive	Retry on another available machine	Doesn't support diverse failure recovery mechanism
Mentat [7]	Grid	Polling	Host crash, Network failure	Reactive	Replication	Exploiting task's stateless and idempotent nature
Proactive FT in MPI [22]	Cluster	Process/node failure	Process/node stops responding	Proactive	Replication	Predicable failures

Table 1: Fault Detection and Tolerance Techniques used in Distributed Systems

4.2. Cluster based Fault Tolerant implementations

J. Hursey et al. in [19] has presented the design and implementation of an infrastructure to support checkpoint/restart fault tolerance in the Open MPI. The design uses an abstract interface for providing and accessing fault tolerance services without sacrificing performance, robustness, or flexibility. The drawback of implementation is that the design supports only some initial checkpoint/restart mechanisms.

C. Wang et al. in [20] have developed a transparent mechanism for job pause within LAM/MPI+BLCR. Fault detection technique discussed in the paper consists of a single timeout mechanism. Excessive delay in response from any process to a message request is assumed to indicate process (node) failure. So, it becomes a drawback that different types of failures are treated as node failure or simply failure.

LA-MPI [11, 12] is a network-fault-tolerant implementation of MPI designed for terascale clusters. The two important goals in LA-MPI are fault tolerance and performance. In LA-MPI process fault tolerance has completely been ignored and it only focuses on network relevant fault tolerance.

Co-Check MPI [21] developed by the Technical University of Munich was the first MPI implementation that used Condor library for implementation of fault tolerance using the technique of check pointing in MPI based applications. The drawback of this technique is that whole application needs check-pointing synchronously, and a bigger application would take more time resulting issues regarding performance.

S. Chakravorty et al. in [22] presents a fault tolerance solution for parallel applications that proactively migrates execution from processors where failure is imminent. The drawback in this idea is that it works on the assumption that failures are predictable.

All the fault tolerant techniques discussed with respect to Clustered computing environment focuses on reactive / post-active fault tolerance techniques, except the one discussed in [22].

5. Conclusion

According to our knowledge and survey, most of the distributed systems (Cluster and Grids) are not handling errors, failures and faults identified in Figure 1 and thus are not reliable. In order to achieve high availability a standard framework for fault tolerance should be the core of each distributed middleware, application or tool and should have the capability to implement that standard framework independently.

6. References

[1] Bran Selic, "Fault tolerance techniques for distributed systems", Staff, IBM, Software Group, 27th July 2004.

[2] A. Avizienis, “The N-version Approach to Fault-Tolerant Software”, IEEE Transactions on Software Engineering.

[3] Pankaj Jalote, “Fault Tolerance in Distributed Systems”, ISBN: 0-13-301367-7, 1994.

[4] Muhammad Affan, M. A. Ansari “Distributed Fault Management for Computational Grids”, Grid and Cooperative Computing, 2006. GCC 2006. Fifth International Conference, pages: 363-368.

[5] Mark Baker, Rajkumar Buyya and Domenico Laforenza, “Grids and Grid Technologies for Wide-Area Distributed Computing”, A technical report August, 2002.

[6] Mohammad Tanvir Huda, Heinz W. Schmidt, Ian D. Peake, “An Agent Oriented Fault-tolerant Framework for Grid Computing”, Proc. of 1st International Conference on e-Science and Grid Computing (e-Science’05), 2005.

[7] Soon Hwang and Carl Kesselman “A Flexible Framework for Fault Tolerance in the Grid”, Journal of Grid Computing 1: 251–272, 2003.

[8] A. P. Sistla , J. L. Welch, Efficient distributed recovery using message logging, Proceedings of the eighth annual ACM Symposium on Principles of distributed computing, p.223-238, June 1989, Edmonton, Alberta, Canada.

[9] N Hussain, M. A. Ansari, M. M. Yasin, A Rauf, S Haider, Fault Tolerance using “Parallel Shadow Image Servers (PSIS)” in Grid Based Computing Environment, IEEE—ICET, 13-14 November 2006.

[10] P. Stelling, I. Foster, C. Kesselman, C. Lee and G. von Laszewski, “A Fault Detection Service for Wide Area Distributed Computations”, Proc. of 7th IEEE Symp. on High Performance Distributed Computing, 1998

[11] R. L. Graham, S.-E. Choi, D. J. Daniel, N. N. Desai, R. G. Minnich, C. E. Rasmussen, L. D. Risinger, and M. W. Sukalski. “A network-failure-tolerant message passing system for terascale clusters”. In Proceedings of the 16th international conference on Supercomputing, pages 77–83. ACM Press, 2002.

[12] R. T. Aulwes, D. J. Daniel, N. N. Desai, R. L. Graham, L. D. Risinger, and M. W. Sukalski. “LA-MPI: The design and implementation of a network-fault-tolerant MPI for terascale clusters”. Technical Report LA-UR- 03-0939, Los Alamos National Laboratory, 2003.

[13] Message Passing Interface Forum. MPI: A Message Passing Interface Standard. Technical report, 1994.

[14] Message Passing Interface Forum. MPI-2.0: Extensions to the Message-Passing Interface. Technical report,

[15] Aulwes, R.T, Daniel, D.J, Desai, N.N, Graham, R.L, Risinger, L.D, Taylor, M.A, Woodall, T.S, Sukalski, M.W, “Parallel and Distributed Processing Symposium”, 2004.

[16] R. Wolski, N. Pjesivac-Grbovic, K. London, and J. Dongarra, “Extending the MPI Specification for Process Fault Tolerance on High Performance Computing Systems,”, ICS, Heidelberg Germany, Jun. 2004

[17] Kandaswamy, G, Mandal, A, Reed D.A, "Fault Tolerance and Recovery of Scientific Workflows on Computational Grids", 8th IEEE International Symp. on Cluster Computing and the Grid, 2008. CCGRID '08.

[18] J. Frey, T. Tannenbaum, I. Foster, M. Livny and S. Tuecke, “Condor-G: A Computation Management Agent for Multi-Institutional Grids”, Cluster Computing, Vol. 5, No. 3, 2002.

[19] J. Hursey, J. M. Squyres, T. I. Mattox, and A. Lumsdaine, “The Design and Implementation of Checkpoint/Restart Process Fault Tolerance for Open MPI,” in IPDPS ’07: Proceedings of the 21st annual International Parallel and Distributed Processing Symposium. IEEE Computer Society, 2007.

[20] C. Wang, F. Mueller, C. Engelmann, and S. L. Scott, “A Job Pause Service under LAM/MPI+BLCR for Transparent Fault Tolerance,” in IPDPS ’07: Proceedings of the 21st International Parallel and Distributed Processing Symposium. IEEE Computer, 2007, pp. 116–125.

[21] G. Stellner, “CoCheck: Checkpointing and Process Migration for MPI”, In Proceedings of the International

Parallel Processing Symposium, pp 526-531, Honolulu, April 1996.

[22] S. Chakravorty and C. Mendes and L. V. Kal´e, “Proactive Fault Tolerance in MPI Applications via Task Migration,” in HiPC ’06: Proceedings of the 13th International Conference on High Performance Computing, LNCS 4297, 2006, pp. 485–496.

[23] A. Grimshaw, W. Wulf and T.L. Team, “The Legion Vision of a Worldwide Virtual Computer”, Communications of the ACM, 1997.