Journal of Engineering and Applied Sciences 14 (3): 785-788, 2019
ISSN: 1816-949X

Validation and Availability Techniques for Computer Faults

Zena Hussain Fahad, Ansam Ahmed Alwan and Zena Tariq Nayyef
Department of Computer Science, Dijlah University, Baghdad, Iraq

Abstract: Users are not concerned with the computer faults but they need the computer working in a correct mode. Specially, in critical circumstances. Fault Computing (FC) plays a important role, especially, since, early 50's. This area opens a wide aspect to be researched area, this study involves varieties of categorizations of techniques towards the effort to make system fault tolerant, modeling and testing helping with system development and bench-marking to evaluate and compare systems. The concept, modeling and methodology of fault tolerant computing are very much controversial to software while they are considered fairly mature for hardware. Often refer to real-time critical system embedded systems. Faculty tolerant computing far from strictly application the system development and operational process in the above area.

Key words: Fault and their manifestation, system fault response stages, validation and availability techniques, FC, techniques, application

INTRODUCTION

A number of modern trends such as harsh climate, novice users, larger and more complicated systems and down costs have running interest in making general-target computer systems fault tolerant and the primary aim of fault tolerance are avert down time and the ensure correct operation even in the existence of faults or more applicable, high availability long life, put back maintenance, high-performance computing and critical computations. System performanc, minimally declare to be the number of results per unit time times the uninterrupted length of time of correct processing, should not be risk. In real systems, however, price-influence trade-offs must be make, fault tolerance features will incur some costs in hardware in performance and both (Mittal and Agarwal, 2015).

Fault features basic allow the computer keep precisely with the presence of imperfections. These systems are usually, classified as either highly assurances or highly available validation and availability techniques for computer fault as a function of time is the modality probability that the system has survived the interval [0, t], given that it was operational at time t = 0. Highly reliable systems are used in positions in which repair cannot take place, (e.g., spacecraft) or in which the computer performing a critical function for which even the small amount of time lost due to patches cannot be tolerated, (e.g., flight control computers). Availability is the intuitive sense of reliability. A system is available if it is able to perform its intended function at the moment the function is required. Formally, the availability of a system as a function of time is the probability that the system is operational at the instant of time, t. If the limit of this function exists as t goes to infinity, it expresses the expected fraction of time that the system is available to perform useful computation (Penry *et al*., 2005).

Availability is a frequently used as a figure of merit in systems for which service can be delayed or denied for short periods without serious consequence. For a system in which downtime costs tens of thousands of dollars/min, (e.g., airline reservation system) an increase of only 1% vailability makes a substantial difference. In general, highly available system are easier to build than highly reliable systems because of the more stringent requirements imposed by the reliability definition (Yin *et al*., 2003).

Fault techniques and architecture found their way in mainstream computer design when computers began to be used in situation which failure could endanger life or property or could foment significant economic loss. Examples of fault systems can be found many now a days for instance, August, Parallel, Tandem, AT and T3B20D, stratus and intel 432 are some well know fault tolerant systems (Castro and Liskov, 2002).

Faults and their manifestation: To realize how a system fails is absolutely necessary before design a fault-tolerant system. Basically, failures start from physical failure and then logical faults appears and then system errors are results. Usually, the definitions include in this propagation process are as follow (Eliaz, 2002):

Corresponding Author: Zena Hussain Fahad, Department of Computer Science, Dijlah University, Baghdad, Iraq

Failure: physical change in hardware. Fault; incorrect state hardware or software resulting from failure of components, physical intervention from the environment, operator error or improper design. Error; appearance of a fault within a program or data structure. The error may occur some distance from the fault site.

Permanent; describes a failure, fault or error that is continuous and constant. In hardware, permanent failure reflects in irrevocable physical change. The word "hard" is used interactively with the word permanent. Intermittent explain a fault or error that is only from time to time present due to unsteady hardware or varying hardware or software states, (e.g., as a function of load or activity). Transient. describes a fault or error resulting from temporary environmental conditions. The word "soft" is used interchangeably with transient.

Transient faults and intermittent faults are the major source of system error. The distinguish between these two types of faults are ability of repair. Consider transient faults considered not repairable and intermittent ones as repairable. The manifestations of transient and intermittent faults and of incorrect hardware or software design are much more difficult of determine than permanent (Castro and Liskov, 1999).

System fault response stages: Table 1 shows the detail of the ten system fault response stages and give each stage a detailed explanation and some more points that need to pay attention.

Validation and availability techniques: Two approaches to increasing system reliability are fault avoidance and fault. Fault avoidance results from conservative design practices such as the use of high-reliability parts. Through the goal of fault avoidances is to reduce the likelihood of failure, even after the most careful application of fault-avoidance techniques, failures will occur eventually owing to defects in the system. In comparison to this approach, fault tolerance appears

Table 1: System fault response stages

Name of fault response stages	Explanation	Extra mention
Fault confinement	Limit the scope of fault affection into local area or protect other areas of the system from getting contaminated by this fault	This technique may be applied in both hardware and software. For instance, it can be achieved by liberal use of fault detection circuits, consistency checks before performing a function ("mutual suspicion") and multiple requests/confirmations before performing a function
Fault detection	Locate the fault. Multiple techniques have been developed and applied for fault detection. They can be basically classified into off-line fault detection. With the off- line detection, the device is unable to perform any function during test, while for the on-line detection, the operation can keep going on while tests and the consequent work are being applied	Thus, off-line detection assures integrity before and possibly at intervals during operation but not during the entire time of operation while on-line techniques have to guarantee system integrity all through the arbitrary period that passes by before detection occurs is called fault latency
Fault masking	Also called static redundancy, fault making techniques hide the effects of failures through the means that redundant information outweighs the incorrect information. Majority voting is an example of fault masking	In its pure from, masking provides no detection. However, many fault-masking techniques can be extended to provide on-line detection as well. Otherwise, off-line detection techniques are needed to discover failures
Retry	In some cases, a second attempt to a operation is effective enough especially for those transient faults which cause no physical damage. Retry can be applied more than once and then when certain number is arrived, system need to go through diagnosis, detection	It may appear that "retry" should be attempted after recovery is affected. But many times an operation that failed will execute correctly if it is tried again immediately. For instance, a transient error may prevent a successful operation but an immediate retry will succeed, since, the transient will have died away a few moments later
diagnosis	Diagnosis stage becomes necessary when detection could not provide fault location and other fault information	Refer to the article of "Diagnosis" listed also in this course web page for more detail
Reconfiguration	If a gut is detected and a permanent failure located, the system may be able to reconfigure its components to replace the failed component or to isolate it from the rest of the system. The component may be replaced by backup spares. Alternatively, it may be switched off and the system capability degraded ad called graceful degradation	Graceful degradation is one of the dynamic redundancy techniques
Recovery	After detection and maybe reconfiguration, the effects of errors must be eliminated. Normally, the system operation is back up to some point in its processing that preceded the fault detection and operation recommences from this point. This form of recovery, often called rollback, usually, entails strategies using backup files, check pointing and journaling	In recovery, error latency becomes an important issue because the rollback must go far enough to avoid effects of undetected errors that occurred before the detected one

Table 1: Continue

Name of fault response stages	Explanation	Extra mention
Restart	This might be possible in the case too much information is damaged by an error or if the system is not designed for recovery. A "hot" restart of fault t, a resumption of all operations from the point of fault detection is possible only if the occurred damage is not recoverable. A "warm" restart implies that only some of the processes can be resumed without lose .A "cold" restart corresponded to a complete reload of the system with no processes surviving	
Repair	Replace the damaged component. It can be either off-line or on-line	
Reintegration	After all, the repaired the device or module is reintegrated into the system. And specially, for on-line repair, this has to be done without delay system operation	

Table 2: Taxonomy of reliability techniques

Region	Technique
Fault avoidance	Environment modification qualitychanges component integration level
Fault detection	Duplication error detection codes(M-of-N codes, parity, checksums, arithmetic codes, cyclic codes self-checking and fail-safe logic watchdog timers and time-outs consistency and capability checks
Static redundancy/masking redundancy	NMR/voting error correcting codes (Hamming SEC/DED, Other codes) Masking logic (interwoven logic, coded-state machines)
Dynamic redundancy	Reconfigurable duplication reconfigurable backup sparing graceful degradation reconfiguration recovery

much better as fault tolerance approaches the system design with the assumptions that defects very much likely surface any way during system operational stage, so that, the design is oriented towards making the system keep operating correctly with the presence of defects. Redundancy is very classic technique used in both fault avoidance and fault tolerance approaches. With the redundancy technique a system could highly likely pass the ten fault response stages listed above (Sari and Akkaya, 2015). Table 2 will give a very clear graphical description of the reliability techniques. And there are some other useful fault techniques such as hardware redundancy, n-version programming, graceful degradation, etc.

Available tools and metrics: Fault Injection is one of the well know techniques/metrics to help measure system fault tolerant capability (Yin *et al*., 2003).

Relationship to other topics

Fault injection: As mentioned above, fault injection is a very useful technique used for measuring system fault capability. It works together with tests generation tools which generate faults to be injected into the system and by measuring the coverage of the faults system able to tolerate, we could get the idea of this particular system capability, fault tolerance.

Software: Validation and availability techniques for computer Fault. software reliability is getting more and more attention to the researchers working in the FC area as it appears to be the vast majority of the cause of system defects. Although, fault tolerant techniques existent so far seem working reasonably well to insure hardware, they are validation not of same effect when applied to world of software. Hardware has the experience of ware-out as a function of time while software never done so. Software reliability has been shown to face the big obstacle of complexity issue.

Software testing: This is the necessary approach for software validation as testing is always an important tool towards system fault capability. As no testing method can explore the population space thoroughly, especially, for software for software testing case due to the prevalent complexity issue, software testing is often considered as an 'art' in this fault tolerance research field (Nandi *et al*., 2013; Kim and Somani, 2002; Oh and McCluskey, 2001).

RESULTS AND DISCUSSION

Experiment: There are two examples to explants the computer fault tolerance: computer system running a program to control the temperature of a boiler by calculating the firing rate of the burner for the boiler. The Fault: If a bit in memory becomes struck at one, that is a fault. If the memory fault effects the operation of the program in such a way that the computer system outputs cause the boiler temperature to rise out of the normal zone, that is a computer system failure and a fault in the overall boiler system. If there is a gauge showing the

temperature of the boiler and its needle moves into the "yellow" zone (abnormal but acceptable), that is a symptom of the system fault. If the boiler explodes because of the faulty firing calculation, thatis a (catastrophic) system failure.

Fault in the memory could be: Chip used might not have been manufactured to specification (a manufacturing fault). The hardware design may have caused too much power to be applied to the chip (a system design fault). The chip design may be prone to such faults (a chip design fault). A field engineer may have inadvertently shorted two lines while performing preventive maintenance (a maintenance fault).

CONCLUSION

In this study, we have introduced basic fault concepts, techniques and tools to achieve this special system feature and also give a description of the core issue, fault, its manifestation and behavior. In general fault tolerance computing is considered as a study of faults/failures as mastering of faults/failures behavior is the reasonable starting point of stopping their effects as any system defect and all those technique and tools are developed toward how to probe this behavior and further how to stop the propagation. As most of the techniques and tools are generated initially for coping with hardware defects or more effective when applied to hardware world, software fault still has not been that relatively mature in comparison with hardware. And software fault research has drawn more and more focus now a days as the majority of system defects are shown to be software defects.

REFERENCES

Castro, M. and B. Liskov, 1999. A correctness proof for a practical byzantine-fault-tolerant replication algorithm. MSc Thesis, MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, Massachusetts, USA.

Castro, M. and B. Liskov, 2002. Practical byzantine fault tolerance and proactive recovery. ACM Trans. Comput. Syst., 20: 398-461.

Eliaz, K., 2002. Fault tolerant implementation. Rev. Econ. Stud., 69: 589-610.

Kim, S. and A.K. Somani, 2002. Soft error sensitivity characterization for microprocessor dependability enhancement strategy. Proceedings of the 2002 International Conference on Dependable Systems and Networks (DSN'02), June 23-26, 2002, IEEE, Washington, DC, USA., ISBN:0-7695-1101-5, pp: 416-425.

Mittal, D. and N. Agarwal, 2015. A review paper on fault tolerance in cloud computing. Proceedings of the 2nd International Conference on Computing for Sustainable Global Development (INDIACom'15), March 11-13, 2015, IEEE, New Delhi, India, ISBN:978-9-3805-4415-1, pp: 31-34.

Nandi, B.B., H.S. Paul, A. Banerjee and S.C. Ghosh, 2013. Fault tolerance as a service. Proceedings of the 2013 IEEE 6th International Conference on Cloud Computing (CLOUD'13), June 28-July 3, 2013, IEEE, Santa Clara, California, USA., ISBN:978-0-7695-5028-2, pp: 446-453.

Oh, N. and E.J. McCluskey, 2001. Low energy error detection technique using procedure call duplication. Proceedings of the 2001 International Symposium on Dependable Systems and Networks (SDN'01), July 1-4, 2001, DSN Publisher, Goteborg, Sweden, pp: B56-B57.

Penry, D.A., M. Vachharajani and D.I. August, 2005. Rapid development of a flexible validated processor model. Proceedings of the 2005 Workshop on Modeling, Benchmarking and Simulation (MoBS'05), June 5, 2005, Madison, Wisconsin, pp: 1-21.

Sari, A. and M. Akkaya, 2015. Fault tolerance mechanisms in distributed systems. Intl. J. Commun. Netw. Syst. Sci., 8: 471-482.

Yin, J., J.P. Martin, A. Venkataramani, L. Alvisi and M. Dahlin, 2003. Separating agreement from execution for byzantine fault tolerant services. Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP'03), October 19-22, 2003, ACM, Bolton Landing, New York, USA., ISBN:1-58113-757-5, pp: 253-267.