[Deep, 2(8): August, 2013]

ISSN: 2277-9655
Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

A Survey on Software Fault tolerance in Parallel Computing

Jashan Deep[*1], **Dr. Rajiv Mahajan**[2]
[*1,2] Department of (CSE), GIMET, Amritsar, India
jashan2009@gmail.com

Abstract

Software almost inevitably contains defects. Do everything possible to reduce the fault rate; Use fault-tolerance techniques to deal with software faults. Fault tolerance is the ability of a system to perform its function correctly even in the presence of internal faults. Most of the ordinary systems lack fault tolerant software fix. This paper surveys various software Fault Tolerance techniques and methodologies. The conventional fault tolerant approaches viz., Recovery Block (RB), N Version Programming (NVP) etc., are too costly to fix in an ordinary low-cost application system because, both the RB and NVP rely on multiple (at least three) versions of both software and computing machines.

Keywords: Software Fault, Fault Tolerance, Recovery Block, N-Version Programming.

Introduction

industries depend upon the computer-based systems for their daily work. The daily works like saving data, updating data, maintaining data and generating reports.

The ability of the software to detect and recover from the fault that is happening or has already happened in the software or hardware in which the software is running to provide the service in accordance with the specifications. In order to implement the fault tolerance it is important to understand the nature of the problem that fault tolerance is supposed to solve.

Software Faults

A fault is the identified or hypothesis cause of an error. Some time it is called a “bug”. A fault is simply a mistake made by a programmer intentionally or un-intentionally or misunderstanding of the requirement specifications. An error is that part of the system state that is liable to lead to a failure. An error may propagate that is an error in one component may cause error in other components. If error detected faults are known to be present. A failure is said to be occurred if the service delivered by the system deviates from the specified service, otherwise termed as incorrect result. Failures so with fault tolerance, prevent failures by tolerating faults whose occurrences are known when errors are detected. The cycle shows relationship between these three as.

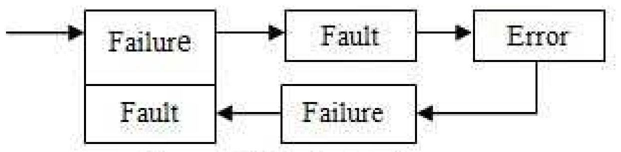


Figure 1.1 Fault Cycle

Types of Software Bugs

Software should work correctly i.e. it should meet the intended specifications in all conditions in the operational phase. On the other hand bugs or errors should be minimize, so that to make a clean and reliable software. These are mainly two types as given below [1]:

a) Bohrbugs: They are essentially permanent design faults and hence almost deterministic in nature. They can be identified easily and weeded out during the testing and debugging phase (or early deployment phase) of the software life cycle.

b) Heisenbugs: They belong to the class of temporary internal faults and are intermittent. They are essentially permanent faults whose conditions of activation occur rarely or are not easily reproducible. Hence these faults result in transient failures, i.e. failures which may not recur if the software is restarted [2].

Conventional Software Fault Tolerance Techniques

From past two decades many researchers focus their efforts on this aspect of the software engineering that is software fault tolerance or how to make dependable software. To apply fault tolerance

ISSN: 2277-9655
Impact Factor: 1.852

researchers developed fault tolerance techniques. In which two basic and original techniques are Recovery Blocks and N-version Programming techniques.

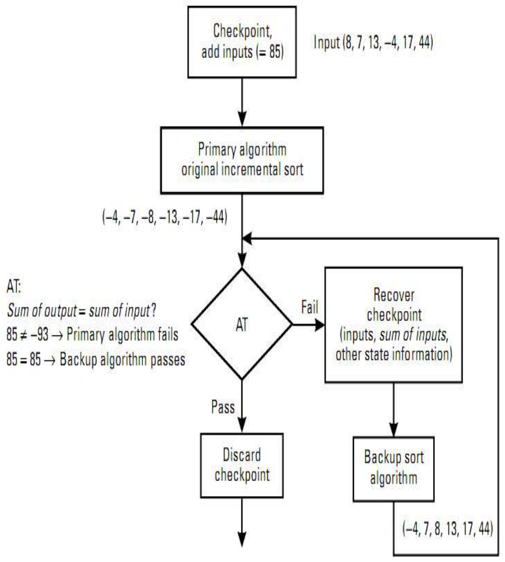
a) Recovery Block: This is one of the basic and original design diverse fault tolerance techniques. Recovery block uses an acceptance test and backward recovery to implement fault tolerance. It has been said that a program function can be performed more than one way by using different algorithms and designs. These differently implemented variants have different degree of efficiency, complexity, memory usage, execution speed and other criteria [3].

The general syntax of the recovery blocks is as follows:

```
Ensure       acceptance test
by           Primary alternate
else by      Alternate  2
else by      Alternate  3
…………………
else by      Alternate  n
else         Failure exception
```

The syntax states that at first the executive will try to satisfy acceptance test by using primary alternate if the result satisfies the acceptance test then the result is provided to the other module or component and the recovery block is exited. If the primary fails to satisfy acceptance test then the executive will try the remaining secondary alternates, if all alternates are exhausted without satisfying the acceptance test then an exception is raised [11].

Recovery Block Example


Checkpoint, add inputs (= 85)
Input (8, 7, 13, −4, 17, 44)
Primary algorithm original incremental sort
(−4, −7, −8, −13, −17, −44)
AT:
Sum of output = sum of input?
85 ≠ −93 → Primary algorithm fails
85 = 85 → Backup algorithm passes
AT
Fail
Recover checkpoint (inputs, sum of inputs, other state information)
Pass
Discard checkpoint
Backup sort algorithm
(−4, 7, 8, 13, 17, 44)

The stepwise working of the above example is as follows:

1. Upon entry to the recovery block, the executive will establish a checkpoint, formats the calls to primary and backup alternates, inputs (8, 7, 13, -4, 17, 44) are summed, sum is 85.
2. Primary sort algorithm is executed the result is (-4, -7, -8, -13, -17, -44).
3. The result is submitted to the acceptance test. Acceptance test sums the elements in the resulted output. The sum of outputs is= -93. Then it compares the sum of inputs=85 with it but 85 != -93 so primary fails acceptance test.
4. Control returns back to the executive, executive sets a flag indicating that primary has failed acceptance test.
5. Executive recovers the checkpoint calls the backup algorithm and passes the same input vector to it.
6. The result of the backup algorithm is submitted to the acceptance test. It compares the output of the result that is, sum of outputs=85 and sum of inputs is=85, 85=85, the alternate passes the test.
7. Control returns to the executive, executive discards the checkpoint and passes the output outside the recovery block and recovery block is exited [4] [5].

b) N-Version Programming: This is the second of the two basic and original design diverse techniques. NVP is further categorized under static techniques. In static software fault tolerance techniques, a task is executed by several processes or programs and a result is accepted only if it is adjudicated as an acceptable result, usually via a majority voting [5]. The NVP technique consists of an executive, n variants and a decision mechanism (DM). The general syntax of the N-version programming technique is as follows [5]:

```
run Version 1, Version 2,……., Version n
if (Decision Mechanism (Result 1, Result 2, ………, Result n))
return Result
else failure exception
```

The syntax states that n-versions run on different processors concurrently. The results of these versions are gathered and submitted to the decision mechanism (DM). Decision mechanism usually uses majority voting to find the correct result. If two or more results are similar then one of them is randomly selected for the output, if it is not able to find a correct result then a failure exception is raised.

N-Version Programming Example

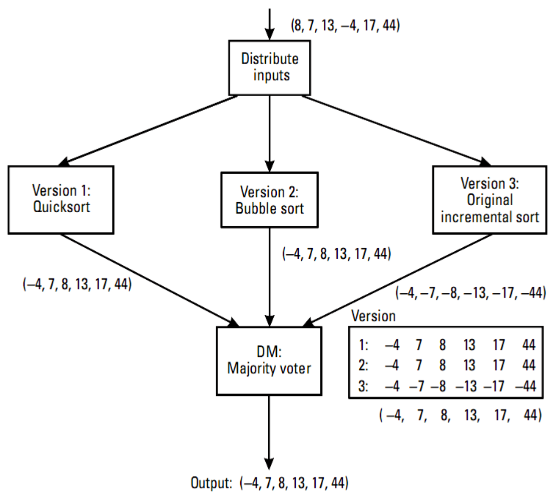


Figure shows the implementation of the NVP technique. The components that are required are an executive which formats the calls to the n-versions, supplies the inputs and synchronizing the variants, one or more versions and a decision mechanism. Three versions are used as quick sort, bubble sort and incremental sort [13][14].

Conclusion

Software fault tolerance is an ability of software to work properly even in the presence of the faults. Two software fault tolerance techniques are discussed here i.e. Recovery Block and N-Version Programming. Recovery block uses the concept of retrying the same operation in expectation that the problem is resolved after the second try. N-Version Programming closely resembles hardware N modular redundancy. Study and analysis the effects of complexity over fault tolerance by different sorting approaches like Bubble Sort Merge Sort, Incremental Sort, Selection Sort, Quick Sort, Insertion Sort.

References

[1] A. Benzekri and R. Puigjaner, "*Fault Tolerance Metrics and Evaluation Techniques*", 1992.
[2] Brian Randell and JieXu, "*The Evolution of the Recovery Block Concept*," University of Newcastle upon Tyne, England, 1971.
[3] Chris Inacio, "*Software Fault Tolerance*" Carnegie Mellon University18-849b, Dependable Embedded Systems, Spring 1998.
[4] D.F. McAllister and M. A. Vouk, "*Software Fault-tolerance Engineering*," Chapter 14 in Handbook of Software Reliability Engineering, Mcgraw Hill, editor M. Lyu pp. 567-614, January 1996.
[5] *K.C. Joshi* & *Durgesh Pant*, "*Software Fault Tolerant Computing: Needs and Prospects*" , ACM Ubiquity, Vol 8 issues 16 , April 2012.
[6] George A. Reis, Jonathan Chang, Neil Vachharajani, Ram Rangan, David I. August, "*SWIFT: Software Implemented Fault Tolerance*," Departments of Electrical Engineering and Computer Science, Princeton University, NJ, June 2002.
[7] Goutam Kumar Saha, "*Software Fault Avoidance Issues*," Member ACM, Ubiquity -- Volume 7, Issue 46 (November 28, 2006 - December 4 2006).
[8] Kim, K. H., and H. O. Welch, "*Distributed Execution of Recovery Blocks: An Approach for Uniform Treatment of Hardware and Software Faults in Real-Time Applications*," IEEE Transactions on Computers, Vol. 38, No. 5, 1989, pp. 626-636.
[9] Laprie, J., and C., "*Definition and Analysis of Hardware- and Software-Fault-Tolerant Architectures*," IEEE Computer, Vol. 23, No. 7, 1990, pp. 39-51.
[10] C.V. Ramamoorthy et al., "*Software Engineering: Problems and Perspectives*," Computer, Vol. 17, No. 10, pp. 191-209, October 1984.
[11] ZaipengXie, Hongyu Sun and KewalSaluja*"A Survey of Software Fault Tolerance Techniques,"* University of Wisconsin-Madison/Department of Electrical and Computer Engineering, 2011.
[12] M. Sghairi, A. de Bonneval, Y. Crouzet, J.-J.Aubert and P. Brot*"Challenges in Building Fault-Tolerant Flight Control System for a Civil Aircraft,"* International Journal of Computer Science, Vol. 35 No.4, Nov 2008.
[13] Goutam Kumar Saha*" A Single-Version Algorithmic Approach to Fault Tolerant Computing Using Static Redundancy"*,Clei Electronic Journal, Vol 9, No. 2, Paper 9, December 2006.
[14] Bharathi V,*" N-Version programming method of software fault tolerance: A Critical Review"*, National Conference on Nonlinear Systems & Dynamics (NCNSD)-2003.
[15] Pham Ba Quang, Nguyen Tien Dat, Huynh Quyet Thang "*Investigate the relation between the correctness and the number of versions of Fault Tolerant Software System"* World Academy of Science, Engineering and technology3 2007.
[16] Goutam Kumar Saha *"A Software Tool for Fault Tolerance"* Centre for Development of Advanced Computing, Journal of Information science and engineering 22,953-964, 2006.

ISSN: 2277-9655
Impact Factor: 1.852

[17] Goutam Kumar Saha *"A Single-Version Scheme of Fault Tolerant Computing"* Centre of Development of Advanced Computing, Kolkata, India, JCS&T, Vol. 6, No. 1, April 2006.