Journal of Engineering and Applied Sciences 2 (10): 1563-1564, 2007
ISSN: 1816-949X
© Medwell Journals, 2007

Providing Fault-Tolerance of Distributed Systems by Process-to-Processor Reassignment

Oleg Viktorov
P.O.Box 1764, Amman 11821, Jordan

Abstract: The study describes the method which provides fault tolerance of distributed systems with not less than four processing modules. The process-to-processor reassignment method is based on process relocation, intermediate results comparison and analysis of special tables. Implementation of this method reduces three times incorrect result probability compared with voting method.

Key words: Fault-tolerance, distributed systems, process-to-processor reassignment algorithm, redundancy, voting, check point, scheduling

INTRODUCTION

Essentially all fault-tolerant systems achieve fault-tolerance by using redundant components in one or another form. Some system use duplicate components and compare their outputs to check the validity of the computation (Jalote, 1994). Other systems use triple modular redundancy with voting, so special voting logic (majority element) compare the component outputs and accept the majority outputs values as correct (Neumann, 1956; Jalote, 1994; Xu and Bruck, 1998; Hardekopf *et al.*, 2001). The main disadvantage of such systems with voting is the failure of majority element causes the failure of the whole system. Another drawback of conventional system with duplication or voting is that they can not locate the faulty components. To reconfigure distributed system, diagnostic functions have to be implemented in the system. An attempt to provide diagnostics using unreliable failure detectors was presented by Chandra and Toueg (1996). In distributed systems with number of processing modules not less than four and one-to-two process-to-processor assignment process shuffling method can be used to detect faulty components and to identify the hard or soft processor failure.

PROCESS SHUFFLING METHOD

Let us consider that suggested method can be implemented in the distributed system which consists of 8 processing modules A, B, C, D, E, F, G, H and run 4 processes P_1, P_2, P_3, P_4. Suppose process-to processor assignment at the i-1 check point step is as follows:

$$P_1^{i-1} \rightarrow (A, B);$$
$$P_2^{i-1} \rightarrow (C, D);$$
$$P_3^{i-1} \rightarrow (E, F);$$
$$P_4^{i-1} \rightarrow (G, H).$$

We are going to use for diagnostics of faulty processing modules the mismatch tables which columns are marked with processing models and rows with i -1 and i check points. The table contains the results of comparison. If the pair of processing modules (A,B) produces the same result at the step i, the squares (i,A) and (i,B) in mismatch table are marks with zeros. If there is a mismatch in results of processing modules (C,D) at the step i, the squares (i,C) and (i,D) in mismatch table are marks with ones (Table 1).

If there is a mismatch in the results of processing modules in the pair (C, D), then at the next step i+1 the distributed system reassigns processes to the processing modules in such way that the processing modules with mismatched results will be separated.

$$P_1^{i+1} \rightarrow (A, B);$$
$$P_2^{i+1} \rightarrow (C, F);$$
$$P_3^{i+1} \rightarrow (D, E);$$
$$P_4^{i+1} \rightarrow (G, H).$$

With the assumption of single failure the four variants of mismatch table are presented in (Table 2-5).

Table 1: Mismatch table

	A	B	C	D	E	F	G	H
i-1	0	0	0	0	0	0	0	0
i	0	0	1	1	0	0	0	0

Table 2: Processing module D is faulty

	A	B	C	D	E	F	G	H
i	0	0	1	1	0	0	0	0
i+1	0	0	0	1	1	0	0	0

Table 3: Processing module C is faulty

	A	B	C	D	E	F	G	H
i	0	0	1	1	0	0	0	0
i+1	0	0	1	0	0	1	0	0

Table 4: Soft failure of C (or D)

	A	B	C	D	E	F	G	H
i	0	0	1	1	0	0	0	0
i+1	0	0	0	0	0	0	0	0

Table 5: Double failure C and D

	A	B	C	D	E	F	G	H
i	0	0	1	1	0	0	0	0
i+1	0	0	1	1	1	1	0	0

Given certain probability to get correct result we can choose a proper value of b (the number of check points). If b = t a system needs t + 1 check points (steps) to finish processes without repeating some steps. Acceptance of multiple failure models brings more variants for mismatch tables and necessity of steps repetition. So, number of process shuffling algorithms with different number of steps and repetitions can be suggested. An algorithm for single failure model without repetition is presented below:

Process shuffling algorithm

```
1.  Set j = b;
2.  Run the process on two processing modules during one step;
3.  If the results are different then fill in the proper squares with ones else go to step 9;
4.  Process execution is postponed for next step;
5.  If j = b go to step 8;
6.  If it is hard failure then eliminate the faulty processor from consideration;
7.  Correct the state pointer of processing module;
8.  Assign processes to new pairs of processing modules and go to step 10;
9.  Fill in mismatch table with zeros;
10. If j = 0 then go to step 12;
11. Set j = j -1 go to step 2;
12. End.
```

Process shuffling algorithm can be implemented as a process (task) scheduling procedure which is run by operating system. Process scheduling in distributed systems have been discussed in the following references (Ishfaq and Kafil, 1997; Hong and Goo, 2005).

RESULTS AND DISCUSSION

New process shuffling algorithm for single failure model has been presented. OS assigns each process to the pair of processors. In case of result mismatch it reassigns the processes to a new pair of processing modules and creates a mismatch table for further analysis. Algorithm can define the type of failure and detect a faulty processing module to provide system fault-tolerance.

Table 6: Probability of incorrect result for the process shuffling system

	Process shuffling system	Voting system
Probability of incorrect result	$q^2(i)/(N^*-1)$	$3q^2(i)/(N^* - 1)$
System performance	$\sum_{i=A}^{a}[(N - i)/2]\sum_{j=0}^{b} Pi, j$	$\sum_{i=A}^{a}[(N - i)/3]\sum_{j=0}^{b+1} Pi(t + jT)$

Where,

q (I)	=	Probability of getting incorrect result at the moment t;
N*	=	Tthe number of possible result variants;
N	=	No. of processing modules;
A	=	System degradation level at the moment *t*;
a	=	Acceptable level of degradation (a = N – 3);
T	=	Processing interval lasting;
P	=	Probability of processing module correct execution;
q	=	1 – P;
b	=	No. of check points
$P_{i,j}$	=	min ($P_i(t + jT)$, $P_i(t + (j + 1)T)$)

Expression, presented in the Table 6, shows that probability of incorrect result for the process shuffling system is 3 times less than for the voting system. System performance of the process shuffling system is 1.5 times higher than the performance of the system with triple modular redundancy (voting system). The main advantage of process shuffling systems is their ability to detect processor failures and reassign processes among unfaulty processing elements again.

REFERENCES

Chandra, T.D. and S. Toueg, 1996. Unreliable Failure Detectors for Reliable Distributed Systems. J. ACM. (JACM)., 43: 225-267.

Hardekopf, B. *et al*., 2001. Secure and fault-tolerant voting in distributed systems, aerospace conference. IEEE. Proc., 3: 1117-1126.

Hong, Y.S. and H.W. Goo, 2005. A fault-tolerant scheduling scheme for hybrid tasks in distributed real-time systems. 3rd IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, 1: 3-6.

Ishfaq, A. and M. Kafil, 1997. A parallel algorithm for optimal task assignment in distributed systems. Advances in Parallel and Distributed Computing Conference (APDC)., 1: 284.

Jalote, P., 1994. Fault-tolerance in distributed systems. PTR Prentice-Hall, NJ., pp: 448.

Von, Neumann, J., 1956. Probabilistic logic and synthesis of reliable organisms from unreliable components. Automata Studies, Princeton University Press, 34: 43-98.

Xu, L. and J. Bruck, 1998. Deterministic Voting in Distributed Systems Using Error-correcting Codes. IEEE. Trans. Parallel Distributed Sys., 9: 823-824.