PAV: Parallel Average Voting Algorithm for Fault-Tolerant Systems

Abbas Karimi[1, 2,*], Faraneh Zarafshan [1, 2], Adznan b. Jantan[2]
[1] Department of Computer Engineering, Faculty of Engineering, Islamic Azad University, Arak Branch, Iran
[2] Departments of Computer and Communication Systems Engineering, Faculty of Engineering, UPM, Malaysia
*Akarimi@ieee.org

***Abstract*—Fault-tolerant systems are such systems that can continue their operation, even in presence of faults. Redundancy as one of the main techniques in implementation of fault-tolerant control systems uses voting algorithms to choose the most appropriate value among multiple redundant and probably faulty results. Average (mean) voter is one of the commonest voting methods which is suitable for decision making in highly-available and long-missions applications in which the availability and speed of the system is critical. In this paper we introduce a new generation of average voter based on parallel algorithms which is called as parallel average voter. The analysis shows that this algorithm has a better time complexity (log n) in comparison with its sequential algorithm and is especially appropriate for applications where the size of input space is large.**

Keywords- Fault-tolerant; Voting Algorithm; Parallel- Algorithm; Divide and Conquer.

I. INTRODUCTION

Fault-tolerance is the knowledge of manufacturing the computing systems which are able to function properly even in the presence of faults. These systems compromise wide range of applications such as embedded real-time systems, commercial interaction systems and e-commerce systems, Ad-hoc networks, transportation (including rail-way, aircrafts and automobiles), nuclear power plants, aerospace and military systems, and industrial environments in all of which a precise inspection or correctness validation of the operations must occur (e.g. where poisonous or flammable materials are kept)[1]. In these systems, the aim is to decrease the probability of system hazardous behavior and keep the systems functioning even in occurrence of one or more faults.

One of the mechanisms to achieve fault tolerance is fault masking which is used in many fault-tolerant systems [2]. In fault masking, hardware modules or software versions are replicated and then voting is used to arbitrate among their results to mask the effect of one or more run time errors.

Replication of hardware modules is the most applicable form of hardware redundancy in control systems which can be in forms of passive (static), active (dynamic) and hybrid.

The aim in static redundancy is masking the effect of fault in the output of system. N-Modular Redundancy (NMR) and N-Version Programming (NVP) are two principal methods of static redundancy in hardware and software respectively. Three modular redundancies (TMR) is the simplest form of NMR which is formed from N=3 redundant modules and a voter unit which arbitrates among modules' outputs (figure 1).

Voter performs a voting algorithm in order to arbitrate among different outputs of redundant modules or versions and mask the effect of fault(s) from the system output. Based on the application, we can use different types of voting algorithms.

Average voter is one of several voting algorithms which are applied in fault-tolerant control systems. Main advantages of this voter are its high availability and its potentiality to extend to large scale systems. Furthermore, in contradict with many voters like majority, smoothing and predictive; it does not need any threshold. The main problem of this voter is that whatever the number of inputs increases, the complexity of its formula increases. Hence, more calculations overhead imposes and the processing speed will decrease. In this paper, we use parallel algorithms on EREW shared-memory systems to present a new generation of average voter – we call as parallel average voter- which provides the average voter extension without enlarging the calculations, suitable for large scale systems and with optimal processing time. Basically there are two architectures for multi-processor systems. One is shared-memory multi processor system and the other is message passing[3]. In a shared-memory parallel system it is assumed n processor has either shared their public working space or has a common public memory.

The current paper is organized as follows: in section 2, background and related works are described. In Section 3, the sequential average voting algorithm and the parallel average voting algorithms are presented. Section 4, deals with performance analysis of new parallel algorithms and its comparison with sequential algorithm. Finally, the conclusions and future works are explained in section 5.

II. RELATED WORKS

Voting algorithms have been extensively applied in situations where choosing an accurate result out of the outputs of several redundant modules is required. Generalized voters including majority, plurality, median and weighted average have been first introduced in [4].

Majority voter is perhaps the most applicable voter that chooses a module output as the output of voting if majority of voter inputs has been produced that value but if less than majority of modules are in agreement, plurality voter can make an agreement. Plurality and majority are actually extended forms of m-out-of-n voting in which at least m modules out of n modules should be in agreement; otherwise, voter cannot produce the output. This voting method is a suitable choice for the systems where the number of voter inputs is large. The other generalized voter is median voter that always chooses the mid-value of voter inputs as the system output. The most significant limitation of this algorithm is that the number of the voter inputs is assumed to be odd [4]. In weighted average algorithm, the weighted mean of the input values is calculated as the voting result. The weight value is assigned to each voter input in various methods [2, 4-6], then, calculated weights, w_i, are used to provide voter output, $y=\sum w_i.x_i/\sum w_i$, where x_is are the voter inputs and y is the voter output. Average voter is a special case of weighted average voter in which all weights are assumed to be equal to $\frac{1}{n}$. In two latest methods, the voting results may be clearly different from input values, while some voters like majority, plurality and median always choose a value among their input values as the voter output.

One difficulty with majority voter and alike is their need to threshold, while so far not any general approaches have been achieved to calculate fair value of them; however, average voter is free of this issue. Furthermore, average voter can always produce output. So the availability of this voter and voter's alike including median and weighted average is 100 percent which makes them the choicest voters for highly available missions.

One critical issue about the voters is their performance in large scale systems. In [7, 8], the above mentioned algorithms along with their operation and time complexity for small and large number of inputs are analyzed and it has demonstrated that the complexity of them depends on the structure of the input space. The main problem with all the weighted methods and consequently average voter is the increasing in the complexity of voter output calculations while the number of voter inputs increases. It also has harmful effects on speed of processing in control system.

To address this problem for average voter, by using parallel algorithms, we have proposed an effective parallel average algorithm based on shared memory EREW. So far, parallel voters have not been taken into account and only two references [2, 9] have covered this issue. In [3], an efficient parallel algorithm has been proposed to find the majority element in shared-memory and message passing parallel systems and its time complexity was determined, while an approach for parallelized m-out-of-n voting through divide-and-conquer strategy has been presented and analyzed in [9].

III. Shared Memory System Parallel Algorithm

In this section, we propose an optimal parallel average voting algorithm on EREW shared-memory systems for large object space applications such as public health systems, geographical information systems, data fusion, mobile robots, sensor networks, etc.

First, we introduce sequential average voting. Then we proceed with introducing and describing the parallel average algorithm with inspirations from the functions of this algorithm and using Divide-and-conquer method and Brent's theorem [10-12].

A. Sequential Average Voting

As mentioned in the previous section, in sequential average voting, the mean of the modules output will be chosen as the output. This will be simply gained through the Lorczak relation mentioned in [4] considering $w_1 = w_2 = \ldots = w_n = \frac{1}{n}$, provided in (1) in which x_i is output of the i^{th} redundant module; w_i, weight of i^{th} module; and X is the output of voter.

$$X = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i} \quad (1)$$

B. Parallel Average Voting

In this section, an effective parallel algorithm is presented for calculating average voting in PRAM machines with EREW shared- memory technology. To do so, the following assumptions are taken into account:

- Array A [1...n] with n elements, comprises $a_1, a_2, \ldots, a_n$,where each a_i is the output of i^{th} module.
- Number of redundant modules, n, is considered as the power of 2.
- Array A is divided to $p = \frac{n}{2}$ sub-arrays each of which contains at most log n element.
- We assume $\sum w_i = 1$, $\sum w_i * x_i = \sum x_i$.
- For enhancing the algorithm, the number of required processors is assumed equal to the number of sub-arrays i.e. p.

The Pseudo-code of our optimal parallel average voter is presented in fig. 2.

Procedure PAV (PRAM-EREW)

Input: A is an array of n elements $a_1,a_2,\ldots,a_n$ where n is a power of 2.

Output: Return X as the output of parallel average voting.

1. A is subdivided into *p=n/2* subsequences A_i of length *log n* where $1 \le i \le n$;
2. ***∀i∈[1 .. n/2]***
   Create array A with corresponding elements a_i in Parallel.
3. ***End Par.***
4. *j ← p;*
5. ***While*** *j>=1* ***do***
6. ***For*** *i=1* ***to*** *j* ***Do in Parallel***
7. *A[i]← A[2i-1]+A[2i];*
8. ***End Par.***
9. *j ← j/2;*
10. ***End While.***
11. X←A[1]/n;
12. **End.**

Figure 1: Pseudo- code of parallel average voter

IV. Analysis and Comparison

In this section, step by step, we try to analyze both parallel and sequential average voting algorithms introduced in sections 3.Aand 3.B through using the rule of complexity of the computation of the algorithms in order to highlight the efficiency of the new parallel algorithm.

To describe the time complexity of the two algorithms we define T_s (n), the function of executing time of the average sequential voting algorithm and T_p(n), the function of the executing time of the parallel voting algorithm in which p is the number of the processors.

Definitely as a result of using $\sum$ operator, sequential average voter needs time complexity equal to T_s (n)=O(n), while parallel algorithm needs constant time of O(1) to divide array A into sub-arrays having maximal length of (log n). Line 2 of PAV uses O(log n) time in order to copy and transfer the information. Since in lines 4-10, we do calculation (adding odd and even nodes) in each sub-array by using tree structure, the overall time complexity of these lines will be equal to O(log n). Finally in line 11, we need an O(1) time to calculate the average voting output.

Hence, the total time complexity of our parallel average voting algorithm is:

$$T_p(n)=O(\log n). \qquad (2)$$

By comparing the time complexities of sequential and parallel algorithms we can conclude that since the execution time of parallel average voter is logarithmic, it is able to run faster than sequential average voter. Also, it can be seen obviously that the total number of required processors in parallel algorithm does not exceed $\frac{n}{2}$. So taking into account the execution time and number of processors needed, the cost and time complexity of the proposed algorithm is better than sequential algorithm. We also have good Speedup (S_p) and Efficiency (E_p) which are indicated in equations (3) and (4).

$$S_P = \frac{T_S}{T_p} = \frac{n}{\log n} \qquad (2)$$

$$E_P = \frac{S_P}{P} = \frac{n/\log n}{n/2} = \frac{2}{\log n} \qquad (3)$$

For large scale system i.e. for big n we have good speed up and efficiency.

V. Conclusion and Future work

In this paper, we proposed an effective parallel algorithm for finding average voting among the results of n redundant modules in parallel shared-memory systems in EREW model. As seen in section 3 the execution time of the sequential algorithm is linear whereas it is logarithmic in our proposed parallel algorithm. Since the parallel average voter can always make result, it has more availability than other parallel voters including parallel majority and parallel m-out-of-n.

Furthermore, in contradict with many voters like majority, smoothing and predictive; it doesn't need any threshold. It also resolves the problem associated with sequential average voter in dealing with large number of inputs.

This algorithm can be implemented in future on parallel on Bus, Hyper Cube and Mesh typologies in message passing systems. Additionally, it can be developed for generating parallel Weighted Average Voting algorithm in which the weights are unequal.

References

[1] G. Latif-Shabgahi, *et al.*, "A Taxonomy for Software Voting Algorithms Used in Safety-Critical Systems," *IEEE Transactions on Reliability*, vol. 53, pp. 319- 328, 2004.

[2] G. Latif-Shabgahi, "A Novel Algorithm for Weighted Average Voting Used in Fault-Tolerant Computing Systems," *Microprocessors and Microsystems*, vol. 28, pp. 357-361, 2004.

[3] C.-L. Lei and H.-T. Liaw, "Efficient Parallel Algorithms for Finding the Majority Element," *Journal of Information Sceince and Engineering*, vol. 9, pp. 319-334, 1993.

[4] P. R. Lorczak , *et al.*, "A Theoretical Investigation of Generalized Voters for Redundant Systems," in *FTCS-19. Digest of Papers., Nineteenth International Symposium on Fault-Tolerant Computing,*, chicago, USA, 1989, pp. 444-451.

[5] Z. Tong and R. Y. Kain, "Vote Assignments in Weighted Voting Mechanisms," *IEEE Transactions on Computers*, vol. 40, pp. 664-667, 1991.

[6] G. Latif-Shabgahi, *et al.*, "A Novel Family of Weighted Average Voters for Fault Tolerant Computer Systems," in *Proceedings of ECC03: European Control Conference*, Cambridge, UK, 2003.

[7] B. Parhami, "Voting Algorithms," *IEEE Transactions on Reliability*, vol. 43, pp. 617-629, 1994.

[8] B. Parhami, "Optimal Algorithms for Exact, Inexact and Approval Voting " presented at the 22nd International Symposium on Fault-Tolearnt Computing(FTCS-22), Boston,N.A,USA, 1992.

[9] B. Parhami, "Parallel Threshold Voting," *The Computer Journal*, vol. 39, pp. 692-700, 1996.

[10] R. P. Brent, *Algorithms for Minimization without Derivatives*. Englewood Cliffs, NJ.: Prentice-Hall, 1973.

[11] P. B. Richard, "The Parallel Evaluation of Arithmetic Expressions in Logarithmic Time," ed: Academic Press, 1973.

[12] R. P. Brent, "The Parallel Evaluation of General Arithmetic Expressions," *J. ACM*, vol. 21, pp. 201-206, 1974.

AUTHORS PROFILE

Abbas Karimi was born in Ahwaz, Iran, in 1976. He received the B.S. degree and M.S. degree in computer hardware and software engineering from Iran. He is currently Ph.D. candidate of computer system engineering, UPM, Malaysia. He has been working as a lecturer and a faculty member in the department of computer engineering in I.A.U-Arak Branch. He was leader of multiple research projects, and author of three textbooks, multiple journals and conference papers. He is senior members of IACSIT, member of IEEE, IAENG, SDIWC, WASET and reviewer in multiple journals. His research interests include load balancing algorithms, and real time, distributed, parallel and fault-tolerant systems.

Faraneh Zarafshan was born in Ahwaz, Iran. She received the B.S. degree and M.S. degree in computer hardware engineering from Iran. She is currently Ph.D. candidate of computer system engineering, UPM, Malaysia. She was leader of multiple research projects, author of three textbooks, multiple journals and conference papers .She is senior members of IACSIT, and member of SDIWC. Her research interests include sensor network, real time systems, and fault-tolerant systems.

Adznan b. Jantan Obtained his PhD from University College of Swansea, Wales, UK. He is currently associate professor in Universiti Putra Malaysia (UPM) under the Faculty of Engineering. Before that, he had been collaborating with Universiti Sains Malaysia (USM), Multimedia University of Malaysia (MMU), Universiti Islam Malaysia (IIUM) and King Fahd University Petroleum Minerals (KFUPM), Saudi Arabia as a lecturer. He has published many papers in international conferences and journals and is the author of several books in the field on engineering. His research interests include speech recognition systems, data compression systems, human computer interaction systems, medical imaging, and smart school design systems.