ŽILINSKÁ UNIVERZITA V ŽILINE
FAKULTA ŠPECIÁLNEHO INŽINIERSTVA

KRÍZOVÝ MANAŽMENT - 1/2010

A REVIEW OF FAULT TOLERANCE WITH THE APPLICATION TO PARALLEL COMPUTER SYSTEMS

Milan ROŠKO[1] Jozef ŽIVČÁK[2]

SUMMARY:

In this paper we review the main definitions and concepts of fault tolerance of a system, and discuss various approaches of implementing the ability to withstand faulty subsystems in a complex system, with the application to parallel computer systems. We describe some approaches that support fault tolerance through a process of recovery from a failure. We also describe parallel fault tolerant system model on basis of Dataflow computer system designed at Technical University of Košice, Faculty of Electrical Engineering and Informatics, which is able to recover after error.

Key Words: Fault Tolerance, Parallel Systems, DataFlow Model

1 INTRODUCTION

All real systems can malfunction and fail due to faults in their components or parts. Logically, the chances for malfunctions increase with the system's complexity. Fault detection and fault diagnosis are main approaches in fault tolerance management of all real systems.

To derive a fault tolerance behavior of the system, we must recall the definitions that state what a system as itself is, while we focus on the system whose correctness and reliability is our interest.

A system is defined as an identifiable mechanism that maintains an observable behavior at its interface with its outside environment. Each system can be modeled as a set of components which interact each other different way(s). In order to emphasize the hierarchical structure of components that are part of the system, we need to denote a lower level system as a subsystem of the main upper level system.

A fault is an abnormal condition that may cause a reduction in, or loss of the capability of a functional unit to perform a required function [1].

Failure in the system is an event which corresponds to the first occurrence of the generated error. The action that has caused the failure originates in the source of the error. In a way of error source can be considered as a defect in the system, which defines a fault as a source of the error. Thus, a fault has not necessarily generated an error but it has the potential of generating one [2].

The detection of an error is the first step of the recovery mechanism of the fault tolerant system. This will start steps of the recovery process of whole system according to its findings. The system decomposition to subsystem level and its components within the subsystem in aspect of fault tolerance approach process of the system, is described in Section 2.

In Section 3 is described fault tolerant parallel system model which can use MIMD (Multiple Instruction stream Multiple Data stream) parallel computer system analogy with a distributed memory and a communication

[1]Ing.Milan Roško, Technical University of Košice, Faculty of Mechanical Engineering, Department of Biomedical Engineering, Automation and Measurement, Letná 9/A, 042 00 Košice, (Slovak Republic) e-mail: milan.rosko@gmail.com

[2] Dr.h.c. prof. Ing. Jozef Živčák, PhD., Technical University of Košice, Faculty of Mechanical Engineering, Department of Biomedical Engineering, Automation and Measurement, Košice, Letná 9/A, 042 00 Košice (Slovak Republic), e-mail: jozef.zivcak@tuke.sk

system with data interchange (communication) between the processors.

Parallel approach of the Dataflow system that supports fault tolerance through a process of recovery from a failure on basis of Dataflow computer system, designed at Technical University of Košice, Faculty of Electrical Engineering and Informatics, is described in Section 4.

2 HARDWARE AND SOFTWARE FAULTS

In Section 1 we have defined the system, and its subsystem(s). Having said that, this hierarchical structure, demonstrating a system and its interface with the outside environment, its subsystems, and subsystem level interfaces and all components within the subsystem, we can graphically show in the Fig.1 [2].

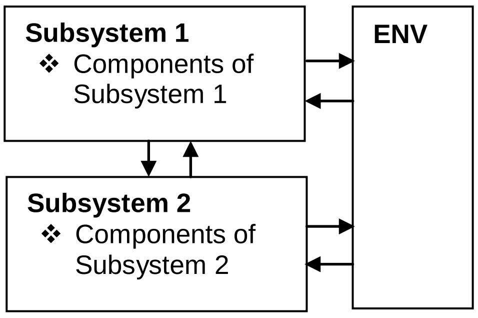

Fig. 1 System decomposition – hierarchical structure (ENV – Environment)

In each system and/or subsystem, a fault (error) has a state defined by the error's value. Also the fault is associated with an action in which the fault was made. Real systems have observability limitations that we may be able to detect the existence of an error not as the failure occurs, but only when it is reflected in the system's behavior.

Furthermore, the fault (error) detected can be a propagated error and not necessarily the generated error. In addition, a transient error is likely to be observable only within a limited time interval.

In many fault tolerance approaches, a fault detection is the first step of the recovery mechanism. This will start steps of recovery of whole system according to its findings. Fig. 2 shows fault tolerance approach process of the system, and its main three steps: (1) Fault detection, (2) Recovery, and (3) Resume execution of the system after a fault was detected and system recovered [2]. This is the subject of fault tolerance control of which detection, isolation and identification of all faults are its important functions [3], [4].

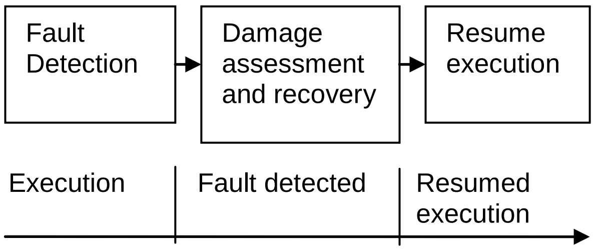

Fig. 2 Fault tolerance approach: (1) Fault detection, (2) Recovery, and (3) Resumed execution

The first step of fault tolerant control concerns the detection and identification of existing fault(s).

A dynamical system with input ***u*** and output ***y*** is subjected to some fault ***f***. The system behaviour depends on the fault $\boldsymbol{f} \in \boldsymbol{F}$, where the element $\boldsymbol{f_0}$ of the set of ***F*** symbolizes the faultless case. The diagnostic system obtains the Input /Output pair ***(U, Y)***, which consists of the sequences:

$$U = (u(0), u(1), u(2), \dots, u(k_h)) \qquad (1)$$
$$Y = (y(0), y(1), y(2), \dots, y(k_h)) \qquad (2)$$

The input and output values are measured at discrete time points ***k*** within a given time horizon $\boldsymbol{k_h}$ [5].

Fig. 3 illustrates the fault diagnosis of a dynamical system.

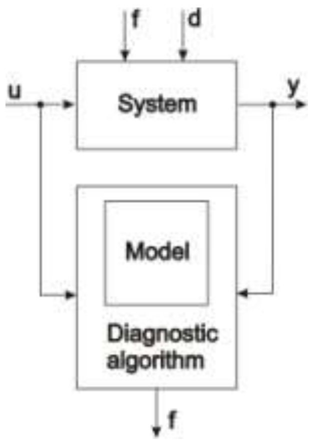

Fig. 3 Fault diagnosis

The role of fault recovery procedure is to bring the system to fault free consistent state. A consistent state of a system conforms with the system correctly reachable states, and the events history as reflected in the system behavior (its interface).

There are two approaches of recovery after fault (error):

(1) *Backward recovery* – it restores the system to a prior state, which is assumed to be error free and is called a recovery point. The advantage of this approach is its simplicity, and the knowledge it requires is that the relevant prior state is fault free.

(2) *Forward recovery* – it assumes that no previous state is known to be fault free, and thus it restores the system state by some sequence of actions. To achieve a correct system state, there must be a good knowledge of what is wrong in the system's current state [4], [6].

Temporal redundancy is sufficient for recovery from transient faults, while physical redundancy is better suited for permanent faults. In a case of transient faults in hardware and software, temporal redundancy is achieved by repetition of allocation and the execution of object is adequate (e.g., transient fault in communication over the communication network). A permanent fault (in hardware) cannot be recovered by a repeated execution of the specific operation.

Real time constraints significantly narrow the ability of the system to recover in the time domain, since a repetition can lead to a deadline failure.

There are temporal effects of faults which contaminate the system behavior: (1) Operations are not completed in scheduled time, thus processor/process is not released on time for other tasks; (2) Latency effects; (3) Invocation of time-out mechanisms changes the system behavior [2].

3 FAULT TOLERANT PARALLEL SYSTEM

To obtain a parallel fault tolerant system, we can use MIMD (Multiple Instruction stream Multiple Data stream) parallel computer system analogy with a distributed memory and a communication system with data interchange between the processors. Such a system contains processor elements (PE), communication links and switches, where all generate a system topology or intercommunication network.

Each processor executes at least one application process. Application processes are executed by parallel or sequent order, to communicate through the communication (intercommunication) network, and they are part of the executed task. Parallel system is able to execute more than one task. All processes (running or idle) are mapped and/or associated to PE.

System is fault tolerant if executes its function(s) even the hardware fault and/or software (program) errors. It means: (1) program wasn't terminated or changed as a result of error; or (2) the results are still valid even the error occurred; or (3) the execution finished at real time; or (4) the result is valid and correct. Fault tolerant system could be achieved by using a redundancy software, a redundancy hardware, or a redundancy data [7], [8].

High-integrity systems must have the ability of fault tolerance, thus the faults are compensated in such a way that they do not lead to system failures. After the application of principles to improve the perfection of the components of the system, the addition to the considered module - one or more modules - can be added as backup modules in parallel configuration [1] (Fig. 4).

The function modules (identical, diverse, modified) can be hardware or software components or modules. In general, the function modules are controlled with fault detection capability of the system followed by a reconfiguration mechanism to switch off failed modules and to switch on spare modules (dynamic redundancy) [4].

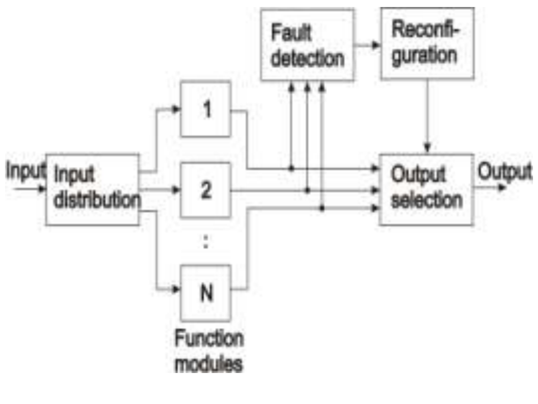

Fig. 4 Scheme of a fault tolerant system with parallel function modules as redundance [1]

System is able to recover from its last (actual) process failure only if has a record of its last (actual) process, and an information about last (actual) process status. Let us assume that there is one copy of process (e.g., on the second processor), where the actual process is running on first processor. In this case, the assumption that a fault (error) appears on one processor only, we have a fault tolerant system, thus the process is able to run on second processor (as a back-up process).

The part of fault tolerant operating system is its core for fault tolerance and/or resistance. It is set of high priority processes distributed to each processor, executing tasks and functions of recovery from the errors. Operating system must have full record of status, parameters, data (values), processes, and all resources of the system in order to fully recover from the error(s) with the assistance of functions and data records of other parts of the operating system.

The roles of operating system during the recovery process are: (1) Terminate any of running process; (2) Record process status; (3) Reallocate processes in consequence of fault (error) occurred; (4) Transfer any process from one PE to another one; (5) Start, and reschedule a process communication; (6) Control and/or actualization of processor and processes parameters and data during the recovery system stage.

The detailed description of the fault tolerant operating system and its core for fault tolerance and/or resistance is in [5] and [8].

4 DATAFLOW MODEL

In this section we describe some parallel approach of the dataflow system that supports fault tolerance through a process of recovery from a failure. Many fault tolerance mechanisms employ such a process sequentially, as a reaction to a failure. However, these actions are not sequential.

At Technical University of Košice, Faculty of Electrical Engineering and Informatics was designed the parallel fault tolerant system model on a basis of Dataflow (DF) computer system, which was able to recover after error.

Dataflow (DF) computer (Fig. 5) is computer based on the DF computing model, where the instructions/programs are executed whenever competent operands are available. DF computer system have the potential for exploiting all the parallelism available in the program represented by the direct, acyclic dataflow graph DFG.

The nodes in the DFG represent the instruction (operator), which is a pure function whose output is determined solely by its inputs. The arcs indicate the flow of the data token (DT) from producer to consumer instructions (nodes) [9], [10].
During the instruction execution the Coordinating processor (CP) can traverse across *LOAD, FETCH, OPERATE, MATCHING and COPY* States.

The operation result is available for the next operator execution, if the *CP* is free *(CP_free=0)*. If *CP* is busy, then the instruction execution is provided by another *CP* available through the interconnection network *IN* and its control of interconnection network switch *INS*.

Detailed description of the DF computing model organization, its main components and functions is given in [8].

The Dataflow model consists of the processors as a part of various systems. The main role of system diagnostics is to detect and localize of fault (error) occurring, and/or to inform about the fault (error) on the level of PE, communication link, and switches. System recovery after process fault occurred is realized by recovering faulty process(es), and/or process(es) affected by faulty processor.

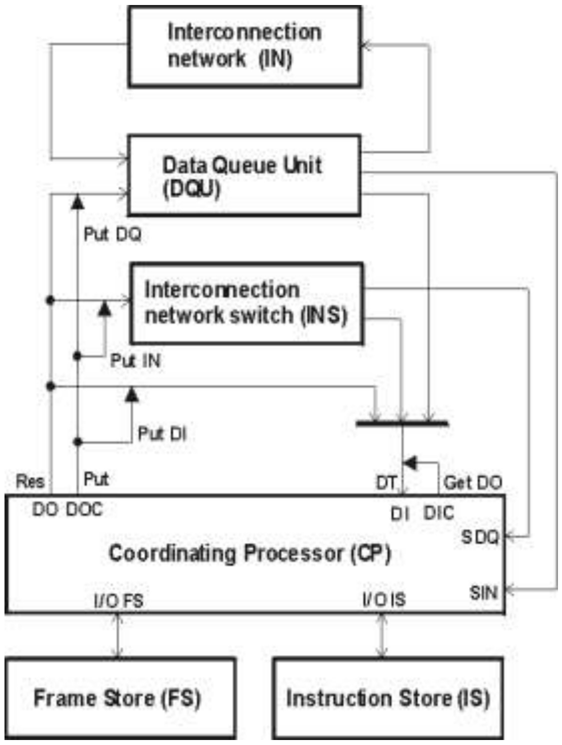

Fig. 5 Dataflow computer system - architecture structure

Fault tolerance provides an application method of system stability (resistance) against errors. If error occurs, the system becomes instable in consequence of processor, network and/or switch fault. Instability of the system bounds to reduction of the system fault tolerance, and/or declining in processor performance.

From the fault seriousness point of view, the communication network has the less impact, which results in to an active process deadlock. In this case, the core for fault tolerance and/or resistance must stop initializing next communication over the faulty network and redirect it (if possible). A fault tolerant communication system between control units and nodes can be realized by a multiple bus system, which has to cover real-time requirements (at least a dual bus system with two independent buses). Both buses are then connected to all nodes (redundancy) [1].

Faulty switch has more serious impact on fault tolerance system because it influences all network channels connected to and from the faulty switch.

Very serious impact on fault tolerance system has a faulty processor. All allocated processes on affected processor or processor element (PE) must be redirected to another processor or PE and recovered and/or initialized. However, the memory data contents and process status are lost (if there is not a second processor), and the system performance will decline at all.

In the next we describe parallel system model on basis of dataflow computer system (Fig. 5) as a parallel fault tolerant system which is able to recover after error. Each process of a parallel system has a new attribute assigned to and valid since the fault detection to the system recovery state.

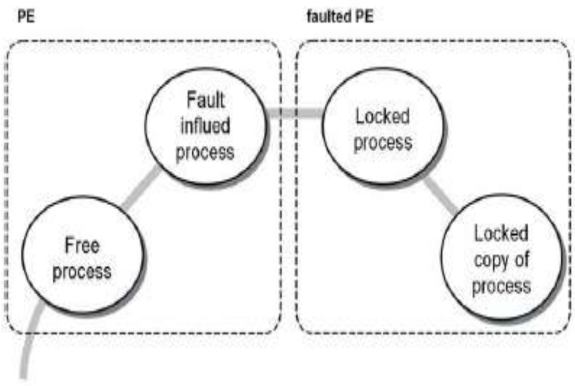

Fig. 6 Interaction of Process Elements and running processes

Process allocated on the faulty PE gets locked (last actual process and his copy as well). Each process interacted with the locked process is ‘process affected‘ by a faulty process. Fig. 6 shows PE, processes (locked, affected/influed, free) and their interaction [3], [8].

Each processor (processor element PE) has allocated processes that generate a task. Processes are planned by CPU (Central Processor Unit) on FIFO (First In First Out) principles.

DF model consists of 5 main pages. Fig. 7 shows its hierarchical relationship sets of 5 main pages, where:

GDS is Global Data Structure (contains Object-oriented fronts);

GDN is Global Declaration Node (text page with definitions, tokens, variables and functions);

FTPS is Fault Tolerance of Parallel System page (top level of DF model; there are 5 identical PE, network initialization, error generator, and message routing system);

PE is Processor Element (consists of 2 pages – Diagnostics and Recovery);

DIAGNOSTICS is Diagnostics (System diagnostic page - generates fault in processor, and/or initializes recovery of the system after detection of fault);

*REC*OVERY is Recovery (Recovery system page).

Detailed description of parallel fault tolerant system on basis of Dataflow system approach, communication processes, message types, can be found in [3], [7], [8], [11].

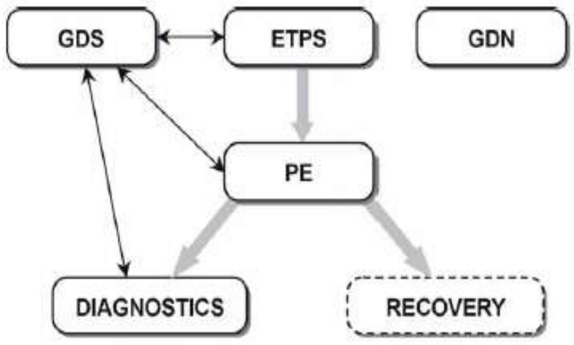

Fig. 7 Hierarchical page relationship of DF model

5 CONCLUSIONS

We have discussed the detection of faults as a first step of the recovery mechanism in fault tolerant systems. This step will start the recovery process of the whole system according to its findings.

The system decomposition to its subsystem levels and its components within the subsystem in aspect of fault tolerance approach process of the system was described (Section 2). If an additional mechanism is added, the system can be altered by either its own faults or this mechanism's faults. Therefore, mechanisms which detect errors must be implemented such that the reliability of these mechanisms does not affect the system (in the fault tolerant aspect).

Parallel fault tolerant system model on a basis of Dataflow computer system which was able to recover from the error was discussed in Section 4, and it was a subject of research at Technical University in Košice (Slovakia), Faculty of Electrical Engineering and Informatics, Department of Computers and Informatics - (VEGA, No. 1/2174/05): "Research of Parallel computing environment used for execution of computing processes in specialized application areas: Theory, Models, Simulation, Assessment, Application Determination". The author was one of the members of the research team [8].

REFERENCES

[1] ISERMANN, R.: Fault-diagnosis systems: an introduction from fault detection to fault tolerance, Springer, 2006, ISBN 3-540-24112-4.

[2] LEVI, S.T. – AGRAWALA, A. K.: Fault Tolerant System Design. Mc Graw-Hill Inc.,1994, ISBN 0-07-037515-1.

[3] VOKOROKOS, L.: Diagnosis of fault tolerant compound system using the observer. Journal of Electrical Engineering, Vol.51, No.11-12, pp. 333-336, 2000, ISSN 1335-3632.

[4] PATTON, R. J. – FRANK, P. M. – CLARK, R. N.: Issues of Fault Diagnosis for Dynamic Systems. Springer, 2000, ISBN 3-540-19968-3.

[5] BLANKE, M. – KINNEART, M. – LUNZE, J. – STAROSWIECKI, M.: Diagnosis and Fault-Tolerant Control. 2nd Ed. Springer, 2006, ISBN 3-540-35652-5.

[6] KORBICZ, J. – KOSCIELNY, J. – KOWALCZUK, Z. – CHOLEWA, W.: Fault Diagnosis. Models, Artificial Intelligence, Applications. Springer, 2004, ISBN 3-540-40767-7.

[7] VOKOROKOS, L.: Data Flow Computing Model: Application for Parallel Computer Systems Diagnosis. Computing and informatics, formerly: Computers and artificial intelligence, Vol. 20, No. 4/2001, SAP, pp. 411-428, ISSN 1335-9150.

[8] VOKOROKOS, L, et al: Research of Parallel Computing Environment Used for Execution of Computing Processes in Specialized Application Areas: Theory, Models, Simulation, Assessment, Application Determination. VEGA No.: 1/2174/05. Technical University, KPI FEI, Košice. Technical Report, 2008.

[9] JELŠINA, M. – ROŠKO, M. – ADAM, N.: Multipipelined and Multithreaded Architectures Approach - Overview of Dataflow Architecture. Proc. of 6th Int. Scientific Conf. on Electronic Comp. and Informatics 2004, ECI'2004, Košice-Herľany, 2004, pp.241-246.

[10] JELŠINA, M: Computer Architectures: Principles, Structure Organization, Functions. Technical University Košice, EF KPI. 467 P., 2002, ISBN 8089066402.

[11] VOKOROKOS, L.: Faults diagnosis of control system using the observer. 4th IEEE International Conference on Intelligent Engineering Systems. Portorož, Slovenia, 2000, pp. 189-192.