Parallel Computing Recovery for Fault Tolerant Systems

Liberios VOKOROKOS

Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Letná 9, 042 00 Košice, Slovakia, Liberios.Vokorokos@tuke.sk

Abstract: This paper deals with the principle of the structure organisation of data flow computer architecture in which direct operands matching is used. The main objective of the proposed architecture is, by the use of data flow principles, to design a computer system for the efficient implementation of functional languages at the programming parallel problems. Data Flow model comprises of functional blocks and activating signs in accordance with message flow in parallel system. A fault in a system occurs according to the continuity of message routing providing the communication between processes and enables the fault diagnosis. These computer configurations take advantageous application for controlling the systems with high severity on security and operation reliability like e.g. processes of continual production (high furnaces), systems of vertical mine transportation etc.

Keywords: diagnosis, recovery, parallel system, Data flow, process, tolerant, fault.

1 Introduction

Nowadays computing art is in vast penetrating into both the producing and non-producing sphere stressing the creation of information and controlling systems. One of possibilities how to increase the efficiency of computing systems is the architecture concept of heavy-duty parallel computing systems. In mono-processor systems based upon the **Von Neumann** computer type, this requirement is achieved by acceleration of individual computer parts.

Conventional **Von Neumann** computers comprise the Control Flow computing model. Computing process is governed by interpretation of sequential program instruction flow [1, 10, 12]. In terms off different tendency in development of computers of new generation marked by extremely high performance, there is at present, attention paid to a special type of parallel computers based on computing model Data Flow (DF) [2, 8, 20].

Data flow computer architectures are based on the DF computing model where program instructions are executed when corresponding operands (data) are enabled. From several DF computing models dynamic models belong to the most significant ones.

The dynamic DF models enable parallel execution of some operators in dependence on the number of operands, which are available for the execution. The nodes of the corresponding DF graph (DFG), by means of which the computing process is represented, can be created dynamically during the execution.

The operands matching enables to execute the instruction which represents the corresponding node in DFG [3, 7, 11]. This is one of important problems of DF computer architecture design.

This work presents a method for diagnosing parallel computer systems using computer model Data Flow. We come out from parallel computer system MIMD (Multiple Instruction steam Multiple Data steam) with distributed memory and communication based upon exchange of messages. This system consists of processor elements (PE), communication lines and switches. At least one application process is running on each of the processor of parallel system. Processes are executed parallely and sequently, communicating with each other through the communication lines executing one task. Several tasks can be run on the parallel system. Processes are mapped to the processor elements [17, 18, 20].

2 The Parallel Implementation of the Computing Process

The parallelism level of the programs is depended from the granuality of the computing process organisation. The granuality is the scale of computing amount inclosed in the code block which by the one processor element of the parallel system is processed.

Any source program in the functional language [14] is represented by a set of definitions of functions and a main expression to be evaluated. The corresponding data flow graph is issued from the definition of the supercombinator-based intermediate code of the functional language by means of selected DF operators.

$$\rangle\, f_1 x_{1_2} \ldots x_{1_{n_1}} = E_1 \Big[\ldots, c_j, \ldots x_k, \ldots, \Theta_1, \ldots, f'_p E_{p_1} E_{p_2} \ldots E_{p_{n_p}}, \ldots,$$

$$x_r E_{r_1} E_{r_2} \ldots E_{r_{n_r}}, \ldots, C_s E_{s_1} E_{s_2} \ldots E_{s_{n_s}}, \ldots \Big]$$

$$\rangle\, f_2 x_{2_1} x_{2_2} \ldots x_{2_{n_2}} = E_2 [\ldots]$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$\rangle\, f'_1 x_{1_1} x_{1_2} \ldots x_{1_{n'_1}} = E'_1 [\ldots]$$

$$\rangle\, f'_2 x_{2_1} x_{2_2} \ldots x_{2_{n'_2}} = E'_2 [\ldots]$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$?P = E_p \Big[\ldots, c_j, \ldots, \Theta_l, \ldots, f_p E_{p_1} E_{p_2} \ldots E_{p_{n_p}}, \ldots, C_s E_{s_1} E_{s_2} \ldots E_{s_{n_s}}, \ldots \Big]$$

The set of function definitions and the main expression (program) holds the form as follows [15]:

The function f_i of the set $f = \{ f_1, f_2, \ldots \}$ is being computed as expression E_i, which includes constants c_j, variables x_k, operators Θ_l, functions f'_p, variables x_r and constructors C_s. The f'_p, x_r and C_s are defined by the expressions $E_{p_1}, \ldots, E_{p_{n_p}}, E_{r_1}, \ldots, E_{r_{n_r}}, E_{s_1}, \ldots, E_{s_{n_s}}$ where $i, j, k, l, p, r, s \in \{1, 2, \ldots, max\}$ and max is the number of all components of DF program. The program represented by the function which is calculated as an expression E_p with components $c_j, \ldots, \Theta_1, \ldots, f_p, \ldots C_s$. The expression E_p contains no variable at all.

The indices "i"s are omitted for the shortness, i.e. $f_i = f$, $x_i = x$ and so on. The block code (supercombinator) of the translated function f is fired by the synchronising token, which is indicated like the trigger.

3 Process of System Recovery

This part of the paper describes the function of the system after system fault. Faults in different parts of parallel system have different importance. Let's think about a fault processor, line or switch. The most important is fault on processor. In this case the processes allocated on this processor have to be moved to other processor, recovered and initialled one more time. Usually we can think about that processor memory content is lost after fault appearing, or unaccessing. It is necessary to remove and to redirect all communications lines going through this process [4,16].

Every process of parallel system from the moment when the fault appears till the end of the recovery is getting a new attribute (fig.1). When processor element PE failed, then:

- every process allocated on the processor element PE is called locked process main and copy too,
- every process except locked process, communicating with locked process is called fault influenced process,
- every process except locked process, not communicating with locked process is called free process.

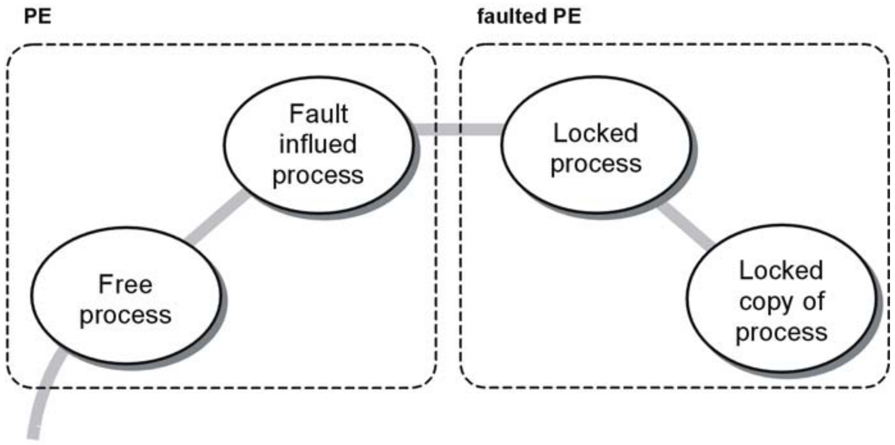

Fig. 1 Properties of processes after fault appears in processor element

The process of system recovery is known. But there is a question how and who controls recovery of kernel of processor. Control can be either centralised or decentralised. In case of decentralised control it is necessary to build on the fact that all kernels dispose the same data, according to which they determine final processors. Every kernel determines final processors for those locked processes which have on its processor allocated copies of processes. If the copy of process is located on more than one processor then the corresponding processor transmit message about system recovery to other processors where the other copies are located. Content of the message is about final processor for the exact copy of process and time mark of begin of recovery. The kernel of the system after receiving all messages about system recovery compares this time mark with its own time of recovery. Lately the kernel doesn’t realise any code reallocation of the relevant processes. In case of equality of time marks can be decisive by another criterium, like for example identification number of a processor.

There is a question how many copies of processes are enough for sufficient resistance against faults. In case of active and passive processes it depends on requested security. One passive copy of the process is sufficient if we assume, that fault doesn’t appear on two processors occupied by the same process at the same time or in time of recovery of the system [19].

4 Data Flow Model of Parallel System

A general model of a part of recoverable parallel system resistant to faults is designed through Data Flow model. It consists of the processes which can be placed into different parts of the system changing their mutual distance. There are allocated processes on the processors which build one task and communicate with each other. They are planned on CPU (Central Processor Unit) according to FIFO (First In First Out) Principe on constant time T (time of the simulation of the process). After some cycles on CPU process requests other process for communication (requested communication). Communication is permitted if requested process finishes task on CPU. When the communication is done both of the processes are inserted to queue for assignment CPU or they can request for another communication. Communication is blocking. This way every process requests and also is requested by another process for communication [6,9,13].

The copies of the processes are also allocated on the processors. Let’s assume, that fault appears on one of the processors. System diagnostics detects this condition and recovery of the system begins. Locked processes are recovered from the copies of the processes and the task can continue. Every activity of every process is recording in system through the whole simulation – cycle on CPU, communication. Based on this record it is possible to intepretate the change of the system efficiency [5].

DF model consists of six pages. Their hierarchic relation is shown on figure 2. The pages are:

- GDN – Global Declaration Node – text page with definitions of coloured group, marks, variables and functions.
- FTPS – Fault Tolerance of Parallel System – top level of DF model of part of parallel system resistant against faults. In this part there are 5 identical processor elements PE, initialisation of the network, generator of fault and the system of data flow.

- PE – processor element, consists of 2 pages – diagnostics and recovery.
- Diagnostics – page of system diagnostics generate fault in processor, which should be in fault and in other processors inicializates recovery of the system after detection of fault.
- Recovery – page of recovery of system kernel. Recovery is realised by decentralizated method in every recovery page of processor.
- GDS – Global Data Structure consisting of object – orientated queues.

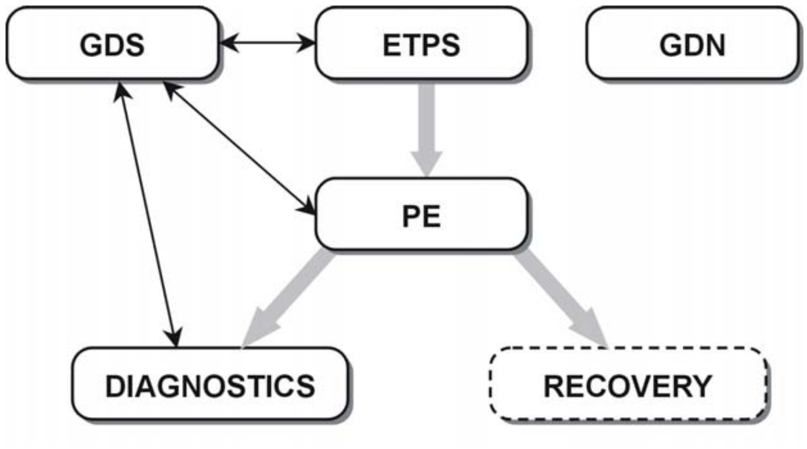

Fig. 2 Hierarchic page of DF model - part of parallel system resistant against faults

5 Page of DF Model - FTPS

Page FTPS (Fault Tolerance of Parallel System) is on top of hierarchy model of system. It consists of two parts:

- inicialization of DF model, generator of fault,
- movement of messages.

Model of inicialization of DF is on fig. 3. These operations are performed on page FTPS:

- opening and reading input file Parameters, which contains input information about simulations,

- opening output file Results for logging simulation,
- generating mark <AF_I, GenerProc,30>, (AF is Activation Frame, I is Instruction)
- generating mark <AF_I, Wait, Tfault> used for generating faults,
- generating mark <AF_I, Wait, Tsimul>, used for closing output file Results.
- creating copy of every process,
- mark of main process <G_PM, index> is assigned for allocation process on page PE,
- mark of copy of process <G_MCP, index+30> is assigned for recovery of system on page Recovery,
- generating mark <G_PoutS, message12>. Length of mark is equal to length of code of process and is assigned to processor, where process will be allocated.

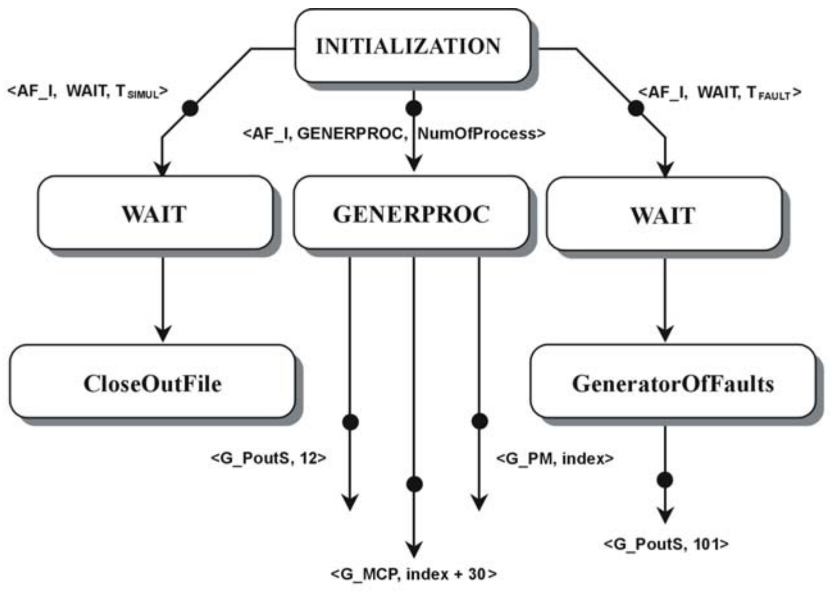

Fig. 3 Model of initialization of DF

Main processes in Processors Memory (PM) are used for allocation on each processor [3]. Copies of processes have identification numbers increased by 30 more than their main processes. All processes are located in Memory of Copies of

Processes (MCP) during all simulation, where all processors can access. This way is model more simple.

In fact the code is not transferring through junction net, but instead of it message type of 12 with the same length as code of process. Transition of the process after initialization of net is bypassing for easier synchronization of all processes. When time of simulation Tsimul expires, transition CloseOutSub is executed and output file Results after simulation will close.

When time Tfault expires transition GeneratoOfFaults will execute and generate message 101, which will force fault to exact processor planned to get into the fault according to input file Parameters.

Type 12 - message is generated by transition GenerProc on page FTPS when processes are generating during initialization of simulation

Type 101 - message is generated by transition Generator of faults on page FTPS after expiration of Tfault on page is only one and is assigned to transition EventOfMyFault on page Diagnostics of processor, which has get to the fault.

5.1 Programm Blocks of Initialization

Block Initialization. From this program block will start simulation of parallel system from level of reading input file and filling global data structures. Process is allocating array of pointers as many as processes when allocating global variables.

Block Wait. This block holds activation mark for exact time (total time of simulation and time of fault appearance in this parallel system) for purpose of planning of simulation DF model. The mark is hold in this block taken in seconds.

Block GenerProc. Indexes of processes and copies of processes are generating in this block to global data memory. Indexes of processes are between 1 to 30 and copies of processes form 31 to 60. Also message type of 12 is added to queue.

Block CloseOutFile. This block is simpler and is used for closing output file of simulation of this DF model.

Block GeneratorFaults. In this block is generated fault after access activation mark, which is delayed from previous block Wait. Under term of generation of fault we mean sending activation mark to memory of output messages with statement message type of 101.

Conclusion

In the proposed data flow computer architecture a basic outline of its structure organisation is emphasised. The direct operands matching and instruction processing by means of coordinating processors are presented, too. DF model is powerful computing engine for computing complex and time demanding mathematical functions, which are separable to independent functional blocks.

One of the main objective of this work was to applicate principles of Data Flow for designing the bulk data processing systems when fault can also appear. DF model is described through DF graph, which consist of functional blocks and activation marks. Based on these properties of activation marks of DF graph there was another problem of matching operands. To solve this issue data structures were designed and ways of controlling of selection.

It is expected that the new model of the DP architecture, leading to the powerful implementation of functional language, will support the programming on the basis of both the specification and the transformation in the environment of parallel computer architectures, being built mainly on the combination of the data driven and reduction computing models.

References

[1] Abram, G., Treinish, L.: An Extended Data-flow Architecture for data Analysis and Visualisation, Proc. of Conf. on Visualisation ´95 (Cat. No. 95CB35835), Atlanta, Ga, USA 1995, 263 - 270, 461.

[2] Bohm, A., Sargeant, J.: Code Optimization for Tagged – Token Dataflow Machine. IEEE Transaction on Computers, Vol. 38, No. 1, Jan. 1989, pp. 4-14.

[3] Depta, J.: Data Flow Architecture for Advanced Process Control, Proc. of Conf. on Computer Software Structures Integrating AI/ KBS Systems in Process Control, A Postprint Volume from the IFAC Workshop, Lund (Sweden) 1996, 21 - 26.

[4] Frank, P. M.: Advances in Observer Based Fault Diagnosis. International Conference on Fault Diagnosis. France, 1993, pp. 817-836.

[5] Hudec, L.- Lesko, J.: Parallel Computing Recovery by Rollback Point Insertion. In: Proc. Of Scientific Conference with Intern. Participation. Electronic Computers and Informatics, Košice-Herlany 26-27.9.96, pp. 2-12 (in Slovak).

[6] Hungwen Li, Stout, Q.F.: Reconfigurable SIMD Massively Computers. Proc. Of the IEEE, Vol.79, No.4, April 1991, pp. 429-443.

[7] Hwang, K.: Advanced Computer Architecture: Parallelism, Scalability, Programmability. McGraw-Hill, Inc., 1993, 768 p.

[8] Iannucci, R.: A data flow/von Neumann hybrid architecture. TR 418, Lab for Computer Science, MIT, Nov. 1988.

[9] Jamil, T., Deshmukh, R. G.: Design of a Tokenless Architecture for Parallel Computations Using Associative Dataflow Processor, Proc. Of Conf. on IEEE SOUTHEASTCON ´96, Briging Together Education, Science and Technology (Cat. No. 96CH35880), Tampa, FL, USA 1996, 649 - 656.

[10] Janík, P.: Optimalization of reconfiguration multiprocessor systems resistant against faults. Thesis, Bratislava 1996 (in Slovak).

[11] Jelšina, M., Krahulík, P., Legnavský, M.: Data Flow Architecture of the Functional Language, Proc. of International Conf. on Information, Communications and Signal Processing, IEEE Singapore Section, Singapore 1997, Vol. 3 of 3, 1452-1456.

[12] Jelšina, M., Legnavský, M.: Dynamic Pipeline Architecture of Data Flow System. Journal of Electrical Enginerinrg, Vol. 50, No. 7-8, 1999, pp. 206-210.

[13] Kollár, J.: Implementation of functional language at the dataflow computer system, FEI KPI TU, 1993.

[14] Kollár, J.: Functional programming, ELFA, Košice, 1995.

[15] Modrák, V., Paško, J., Pavlenko, S.: Alternative Solution for a Robotic Stereotactic System. Journal of Intelligent and Robotic Systems. 2002. Kluwer Academic Publishers, Holand, Nr. 35 (2), p. 193-202, ISSN 0921-0296.

[16] Plander, I.: Mapping Strategies for Reconfigurable Massively Parallel Computers. Third Workshop on Compilers for Parallel Computers, Vienna, Austria, July 6-9, 1992, pp. 359-375.

[17] Vokorokos, L.: Data flow computing model for fault tolerant systems. Fascicola Matematica – Informatica, Buletinul Stiintific al Universitatii din Baia Mare, Seria B, vol. XVI (2000), Nr. 2, 2000, pp. 319-326.

[18] Vokorokos, L.: Faults diagnosis of control system using the observer, 4th IEEE International Conference on Intelligent Engineering Systems 2000, Portorož Slovenia, September 17-19, 2000, pp. 189-192.

[19] Vokorokos, L.: Diagnosis of mechanical machineries using the parallel computer system. Monography. East-Slovak printers l.t.d. 2000. p. 152. ISBN 80-7099-619-6. (in Slovak)

[20] 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.

Supported by VEGA project No. 1/9027/02