International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 03 | Mar-2016 www.irjet.net p-ISSN: 2395-0072

AFT Scheduling Algorithm for Multicore Architecture: A Novel Approach

S.Priyadarsani[1], V.Rajalakshmi[2], Dr.G.Muneeswari[3]

[1]*Student, Department of Information Technology, SSN college of engineering, Chennai*

[2]*Student, Department of Information Technology, SSN college of engineering, Chennai*

[3]*Associate Professor, Department of Information Technology, SSN college of engineering, Chennai*

Abstract - *In this period of modern technology, the speed and efficiency of computing systems have become inevitable. This scenario has led us to adapt to multicore systems over multi-processor units. Multicore systems aids in achieving an enormous increase in throughput and faster execution. In the ever growing energy demands, hardware failure poses a threat to the continuous computing environment. Although various scheduling algorithms have been implemented to improve performance achievement over processor execution, fault tolerance during process execution, is the major challenges faced by the real time systems. In the paper, AFT (Agent Based Fault Tolerance) scheduling for multicore architecture, we propose a new agent based fault tolerant scheduling algorithm which enhances the performance of the overall system by 60% when compared to the traditional system.*

Key words: **Fault-tolerant, Multicore, Agent, Scheduling, Throughput**

1. INTRODUCTION

Multiple Independent processing units built on a single chip constitutes a Multi-core processing system. These multi-core systems provide greater efficiency over single processor running on a single chip. All the cores share a common memory in which the tasks, that are to be assigned, are stored. The scheduler is responsible for assigning the tasks to the individual processors. The tasks are allocated in such a way that none of the processing unit remains idle. This increases the processor utilization and hence the throughput. It is ensured that every core runs some process from the memory.

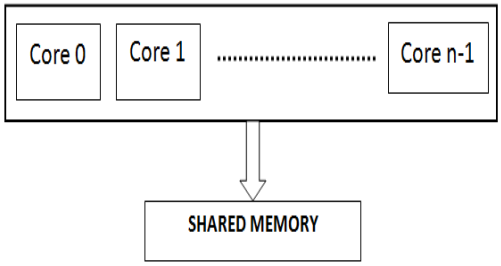


Fig-1: Multicore Architecture

Agent based models, often referred as ABMs, are actively evolving in various phases of technology. These agents are usually software programs or computer simulation, which are embedded in a system to perform a specialized task. A Multi-agent system consists of several agents which are used to test how a change or fault occurrence is likely to affect the overall behavior of the system. Various scheduling algorithm have been incorporated in real time multi-core systems. However, problem such as hardware failure of a processor leads to a drop in efficiency levels. In order to overcome this, fault tolerance mechanisms are being implemented.

In this paper, Section II discusses the Literature Review and Section III elaborates on the working of agent based fault tolerance algorithm .The results are discussed in Section IV and Section V concludes the paper.

2. LITRATURE REVIEW

The research on primary-backup scheduling in real-time systems [1] proposes techniques such as dynamic grouping and PB overloading to increase the scheduling efficiency and reliability. In dynamic grouping, the processors are dynamically grouped into logical groups in order to achieve efficient overloading of tasks on resources. In PB overloading, the primary task can overlap in time with the backup of another task on a processor. Grid Scheduling Algorithm with Load Balancing and Fault Tolerance [2] discusses the Multi constrained Load Balancing Fault Tolerant algorithm (MLFT) which reduces the schedule make span, schedule cost, and task failure rate and improves resource utilization.

The algorithm is evaluated using grid sim toolkit. In Fault-tolerant Scheduling Algorithm for Precedence Constrained Tasks, [3] failure in a heterogeneous system is discussed. For non-preemptive tasks, each task has two copies that are scheduled on different processors and mutually excluded in time. For tasks with precedence constraints, an overlapping scheme allows the backup copy of a task to overlap with its successors' primary copies. The paper on Fault Tolerant Scheduling in Multicore Systems, [4] presents a hardware based algorithm which uses triple and double modulo redundancy. Redundant multi-threaded processes are used which helps in soft errors detection and recovery. In Energy Minimization for Fault Tolerant Scheduling of Periodic Fixed-Priority Applications [5] the problem of energy minimization for scheduling periodic fixed-priority applications on multiprocessor platforms with fault tolerance requirements is discussed. Check points are introduced to allow scheduling of an application which tolerates up to k faults on a single processor. The Fault Tolerant Global Scheduling [6] is a backup based algorithm which uses resource reclaiming fault tolerant global scheduling (RRFGTS). This algorithm delays the execution of backup and rescues the resource distributed to backups after the execution. The dynamic fault tolerant scheduling (DFTS) algorithm in multicore systems [7] is designed to tolerate single or multiple transient faults. In case of multiple faults, the feasibility rate is considerably higher. It is used to dynamically select fault recovery method to overcome the faults occurring in the system. This method makes use of the task utilization for the critical and non-critical tasks. The paper on Tolerance to multiple transient faults [8], it is noted that all the methods proposed in this paper are used to sense only some special types of faults, and therefore there is no appropriate method to detect arbitrary faults that occur in real-time systems.

In this paper [9], sanity and consistency checks are performed at checkpoints and faults are detected at the end of each task. The paper on fault recovery in embedded systems [10] uses check pointing, rollback and re-execution are used to increase execution time and cope with transient faults. The EFRD algorithm is proposed [11] in which the efficiency is improved by prohibiting the overlapping of backup copies and also by assigning the tasks to highly reliable processors. The Fault-tolerant Reliability Cost Driven algorithm is extended by relaxing the requirement that backup copies of tasks are forbidden to overlap with each other. A feasibility check algorithm is introduced in the paper [12] for the fault tolerant scheduling. This algorithm is designed for homogenous systems where tasks are considered independent from one another. The paper on proposing a fault tolerant scheduling algorithm for real time periodic tasks [13] uses the rate monotonic scheduling in which the priority of the task is defined based on the time required. A real-time scheduling algorithm called fault-tolerant static scheduling for heterogeneous systems and for multipoint links [14] [15] considering fault tolerance and tasks with precedence is established here. In order to handle both the processor and communication link failures, the paper on Off-Line Real-Time Fault-Tolerant Scheduling [16], suggests an efficient offline scheduling algorithm which are of good use in real time systems.

3. AGENT BASED SCHEDULING WITH FAULT TOLERANCE

Before starting the execution of a process, the scheduler selects the process from the memory and assigns it to the middle agent(Fig.2). There is also a dedicated Fault-tolerance Agent for each processor which continuously monitors its corresponding processor. It also checks the status of the processor to avoid allocating a task to that core in case of any hardware malfunction. Each processor is checked periodically with a time period of t (ms), in which 't' is less than the burst time of the process with the smallest burst time. The value of 't' gets dynamically updated for each set of process' running on the cores at one particular interval.

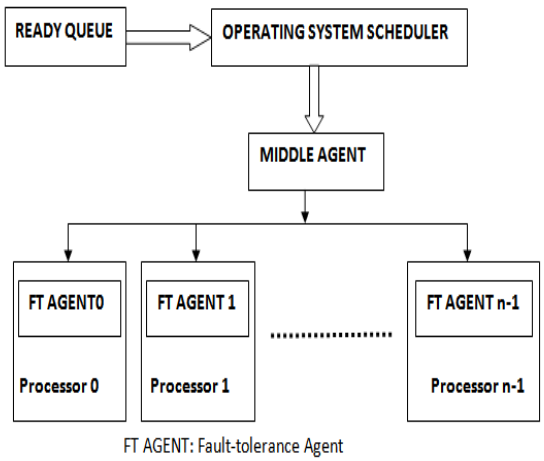


Fig-2: Process Allocation Method

The agent updates the status of the processor as failed, busy or idle to the middle agent. In case of failure, the middle agent informs the scheduler which in turn assigns the task to a different processor. Now, it is again the responsibility of the middle agent to allocate the process to the respective new processor.

3.1. AFT SCHEDULING ALGORITHM

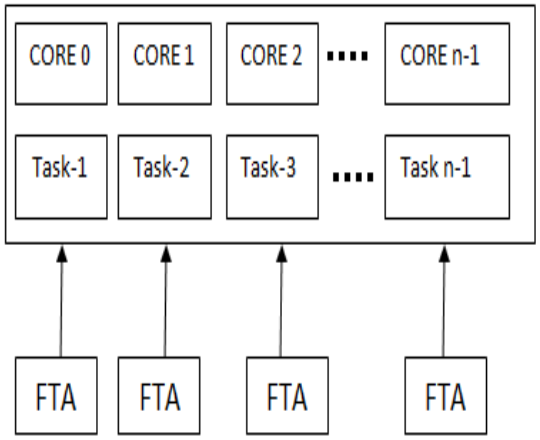
The tasks are selected from the main memory and stored in the ready-queue. The scheduler selects and assigns a process to the middle agent in a round-robin fashion. The middle agent allocates the processes to an independent processor. It checks the burst time of all the processes in the round-robin queue. The fault tolerant agent then monitors each processor periodically with a time period t (ms), where t is less than the burst time of the process with the shortest burst time. The status of the processor after every t ms is noted as busy, idle or failed by the fault tolerant agent. The middle agent is informed in case of a failure status. The middle agent in turn allocates the set of processes to a different processor.

For every new workload i to n perform the following operation:

Begin

Step 1:	Middle Agent (MA) sets the window size as **α** in the ready queue depends on the number of cores.
Step 2:	MA gets the number of process **Pi to Pn** from the ready queue
Step 3:	MA obtains the burst time **Bi** and arrival time **Ai** for all the processes **Pi to Pn.**
Step 4:	MA calculates the minimum burst time by calling the function **min(BT(Pi..Pn))**
Step 5:	MA assigns processes **Pi to Pn** to the processor whenever it becomes ready.
Step 6:	The Fault Tolerant Agent (FTA) monitors the processor periodically with a time period **t<min(BT(Pi..Pn).**
Step 7:	The status of the processor after every **t** ms is noted as busy, idle or failed and the FTA informs the MA in case of failure.
Step 8:	After the first level of execution, MA again calculates the minimum burst time by calling the function **min(BT(Pi..Pn**))with the remaining time tasks.
Step 9:	Repeat steps 6 through 8 until no tasks in ready queue.
Step 10:	Repeat steps 1 through 9 for all the tasks in the ready queue

End



FTA – Fault-Tolerance Agent

Fig-3: Agent based fault tolerance system implementation middle agent. The burst time of tasks P0-P(n-1) is calculated by the middle agent itself(Fig.3). Each processor executes an individual task and is constantly monitored by a fault tolerant agent (FTA). Each dedicated FTA checks its core periodically with a period t (ms).

4. EXPERIMENTAL RESULTS

The Proposed algorithm was implemented and tested in the GEMS simulator and also in the hardware With respect to operating system simulation for our algorithm, we used a gcc compiler and linux kernal. Agent scheduler is simulated and executed with the help of Flame tool. For our simulation we have taken 1000 processes and sample SPEC2000 benchmark programs and tested against 25, 50, 75, 100, 125, 150, 175, 200, 225, 250, 300 cores. Table 1. describes the sample average waiting time for 150 to 250 cores.

Number of Cores	AWT of Sample processes(in ps)
150	5.2
175	4.2
200	3
225	2.4
250	1.2

Table- 1: Sample Average Waiting Time for cores

In Fig.4, the average waiting time of the process along with the number of cores is shown.

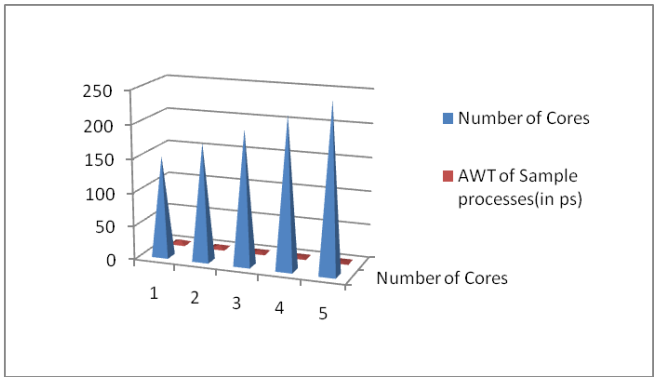


Fig-4: AWT vs No.of Cores

5. CONCLUSION

Fault detection and tolerance have always been a challenge in multi-core systems when compared to uniprocessor systems. In this paper, special fault tolerance agents are used to achieve a fault tolerant environment and to handle unexpected hardware or software failure of the processor. Each agent monitors its respective processor periodically to check for the proper functioning of the core. The status is updated to the middle agent in case of failure. Reallocation of tasks to a different processor is then done by the middle agent. This method helps in achieving an efficient fault tolerant environment and timely completion of tasks/processes, thereby providing increased throughput.

6. REFERENCES

[1] R. Al-Omari,a,1 Arun K. Somani,b and G. Manimaran, Efficient overloading techniques for primary-backup scheduling in real-time systems, J. Parallel Distributed Computing 64 (2004) 629–648

[2] P. Keerthika, P.Suresh, A Multiconstrained Grid Scheduling Algorithm with Load Balancing and Fault Tolerance, Hindawi Publishing Corporation, the Scientific World Journal, Volume 2015, Article ID 349576.

[3] Xiao Qin, Hong Jiang, A Novel Fault-tolerant Scheduling Algorithm for Precedence Constrained Tasks in Real-Time Heterogeneous Systems, Parallel Computing, vol. 32, no. 5-6, pp. 331-356, June 2006.

[4] ShefaliMalhotra, ParagNarkhede, Kush Shah, SamanthMakaraju, M. Shanmugasundaram, A Review Of Fault Tolerant Scheduling In Multicore Systems, International Journal Of Scientific & Technology Research Volume 4, Issue 05, May 2015.

[5] Qiushi Han, Ming Fan, LinweiNiu, Gang Quan,Energy Minimization for Fault Tolerant Scheduling of Periodic Fixed-Priority Applications on Multiprocessor Platforms, 2015 Design, Automation & Test in Europe Conference & Exhibition.

[6] HaoPeng, Fan Yang, Fault Tolerant Global Scheduling for Multiprocessor Hard Real Time Systems, International Conference on Information Sciences, Machinery, Materials and Energy (ICISMME 2015).

[7] .Mohammad H. Mottaghi, Hamid R. Zarandi, DFTS: A dynamic fault-tolerant scheduling for real-time tasks in multicore processors, Microprocessors and Microsystems 38 (2014) 88–97.

[8] F. Liberato, R. Melhem, D. Mosse, Tolerance to multiple transient faults foraperiodic tasks in hard real-time systems, IEEE Trans. Computing. 49 (9) (2000)906–914.

[9] H. Aydin, Exact fault-sensitive feasibility analysis of real-time tasks, IEEE Trans. Computing. 56 (10) (2007) 1372–1386.

[10] Y. Zhang, K. Chakrabarty, Fault recovery based on check pointing for hard real time embedded systems, in: 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT), 2003, pp. 320–327.

[11] X. Qin, H. Jiang, and D. R. Swanson, "A Fault-tolerant Real-time Scheduling Algorithm for Precedence-Constrained Tasks in Heterogeneous Systems," Technical Report TR-UNL-CSE 2001-1003, Department of Computer Science and Engineering, University of Nebraska-Lincoln, September 2001.

[12] F. Liberato, R. Melhem, and D. Mossé, "Tolerance to Multiple Transient Faults for Aperiodic Tasks in Hard Real-Time Systems," IEEE Transactions on Computers, Vol. 49, No. 9, September 2000.

[13] Ching-Chih Han, K.G. Shin and J. Wu, —A fault-tolerant scheduling algorithm for real-time periodic tasks with possible software faults,‖ IEEE Transactions on Computers, vol.52, no.3, pp.362,372, 2003

[14] A. Girault, C. Lavarenne, M. Sighireanu and Y. Sorel, "Fault-Tolerant Static Scheduling for Real-Time Embedded Systems," Proc. Int'l Conf. Computing Systems, April 2001.

[15] A. Girault, C. Lavarenne, M. Sighireanu, and Y. Sorel, "Generation of Fault-Tolerant Static Scheduling for Real-Time Embedded Systems with Multi-Point Links", IEEE Workshop on Fault-Tolerant Parallel and Systems, San Francisco, USA, April 2001.

[16] C. Dima, A. Girault, C. Lavarenne, and Y. Sorel, "Off-Line Real-Time Fault-Tolerant Scheduling," Proc.Euromicro Workshop on Parallel and Processing, pp. 410-417, Mantova, Italy, Feb. 2001.