Designing of Real Time Scheduling Algorithm For Multicore Architecture – A Review

Jyotsna Gaikwad[1], Radhakrishna Naik[2]
[1]Dept of Computer Science and IT
[2]Dept of Computer Science and Engineering
[1, 2] Dr. Babasaheb Ambedkar Marathwada University, Aurangabad
[2]Maharashtra Institute of Technology, Aurangabad

Abstract- *Various works in the literature commerce with the real time scheduling algorithm for multi core architecture. In maximum proposed multicore platforms, different cores share the common memory. Every multicore processor contains of a single computing module with several autonomous central processing units (cores) that perform the user written program instructions. The actual processor expansion has stimulated from single core chips to ones with hundreds or even thousands of cores. In addition to that most of the multi core chips have come up with Simultaneous Multi-Threading (SMT) idea which extremely improves the computing power.*

Keywords- Multicore, Multi-Threading, Processor.

I. INTRODUCTION

The researcher presents a task scheduling algorithm for multi-core processors, which is based on priority queue and task duplication. In this algorithm, the **D**irected **A** cyclic **G**raph (DAG) is used to construct a task model. Based on the model, task critical degree, task reminder, task execution time and the average communication time are all measured as the priority metrics. A priority built task posting list is set up by complete analysis and calculating the priority for every task. Then interval addition and task duplication approaches are employed to map tasks to processors, which can reduction the communication cost, progress the processor utilization rate and reduce the schedule length [1].

Author proposed a task scheduling algorithm on the source of task duplication, which is collected of three steps of operations so that threads are assigned to processing cores more appropriately. Designed algorithm not only increases the executive efficiency of task scheduling, but also can adjust scheduling sets according to the number of processing core. This algorithm reduces communication overhead and keeps load balancing between cores, and for the moment speedup ratio of parallel program is enhanced. The model experiment data shows that the algorithm can find close optimal solutions in rational time, and that it can find results in less time [2].

The faulty multicore (FTM) algorithm performs backups in order to recuperate from errors produced by non-permanent and permanent hardware faults. The worst-case schedule ability analysis of FTM algorithm is obtainable considering an application level error model, which is autonomous of the stochastic comportment of the underlying hardware-level fault model. Formerly, the stochastic behavior of hardware-level fault model is attached in to the analysis to originate the probability of meeting all the goals. Such probabilistic guarantee is the level of guarantee concerning the correct functional and timing performances of the system [3].

The central processing unit is the core of the computer system so it should be used resourcefully. For this purpose central processing unit scheduling is very essential. Central processing unit scheduling is basic fundamental concepts of operating system. Allocation of computer resources between multiple processes is called scheduling [4].

II. SINGLE CORE AND MULTICORE

Single Core: A single core tasks runs at any point in time, meaning that the central processing unit is dynamically executing information for that task. Multitasking explains this problem with the help of scheduling. That means, which task may run at any given time and when alternative waiting task gets a turn [4].

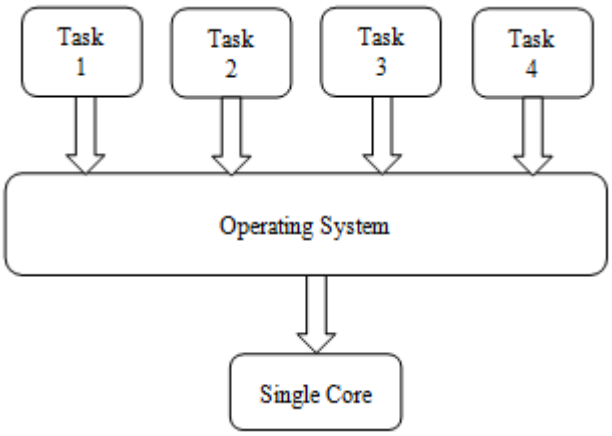

Figure 1: Single Core Architecture

Multicore: When running on a multicore structure, the multitasking operations can execute multiple tasks simultaneously. The multiple computing machines work autonomously on different tasks. For example, on a dual-core system, four tasks can access a separate processor core at the same time. That is nothing but multitask, which improving overall performance of the system [4].

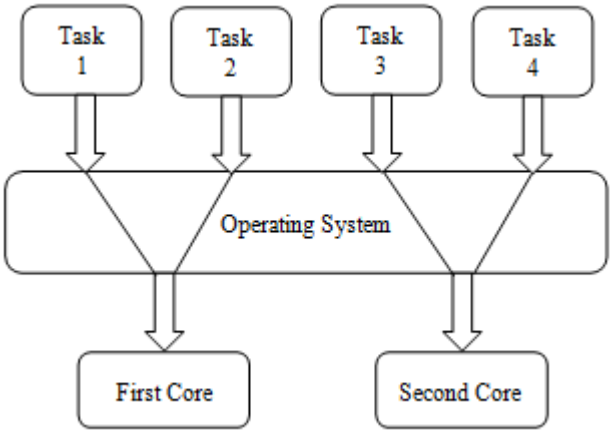

Figure 2: Dual Core Architecture

III. THREAD

A thread is anelementary unit of central processing unitoperation, containing of a program counter, a stack, and a set of registers. Customarymethods have a single thread. Multi-threaded applications have multiple threads within a single process, each thread having their own program counter, stack and set of registers. Multithreading spreads the idea of multitasking into applications [4]

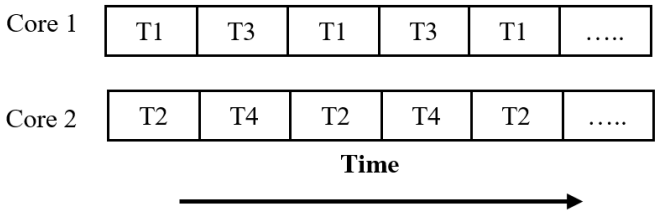

Figure 3: Parallel Execution on Multicore

.

IV. METHODOLOGY

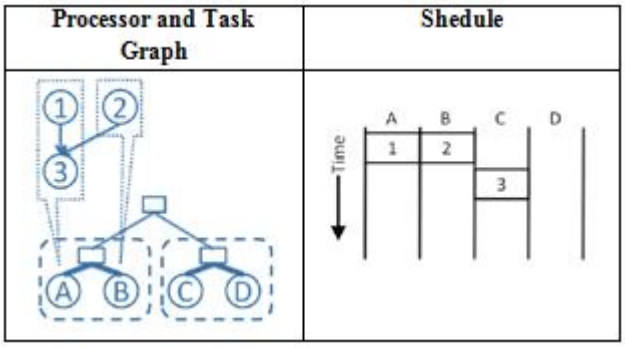

Figure 4: Processor Graph, Task Graph and Schedule

Above figure shows the processor and task graph and schedule.

The symbols used are listed in Table 1.

Task graph: A task graph is a DAG in which each node characterizes a task to be performed. The computation time to execute task v, which is a node in the graph, is denoted $w(v)$.

Processor graph: A processor graph is a graph that signifies the network topology between processors. A node with onlyone link is called a processor node [5].

Table 1: Symbol Table

Symbol	Meaning
G	Task graph
$\boldsymbol{V}$	Set of all task nodes
$\boldsymbol{E}$	Set of all task links
H	Processor graph
$\boldsymbol{P}$	Set of all processor nodes
$\boldsymbol{R}$	Set of all processor links
e_{ij}	Task link from i-th task node to j-th task node
$proc(n)$	Processor assigned to task node $n \in V$
$Proc(N)$	Set of processors assigned to task nodes in set $N \subseteq V$
$pred(n_i)$	Set of all parent nodes of n_i

Scheduling Algorithm:

Following algorithm is a hypothetical scheduling algorithm based on previous literature work.

INPUT: Task graph G = (**V**,**E**, w, c) and processor graph

$H = (\mathbf{P}, \mathbf{R})$.

1: Sort nodes $n \in V$ into list L, according to priority scheme and priority constraints.
2: **for** each $n \in L$ do **do**
3: Find processor $p \in \mathbf{P}$ that allows earliest finish time of$n$.
4: Schedule n on p.
5: **end for**
The algorithm proposed by Sinnen, et al. [5].

V. CONCLUSION

In real-time systems, a task desires to be performed appropriately and timely. The accuracy of each calculation depends on both the logical results of the computation and the time at which results are formed. So, the time is very important in real-time application structures. Multicore and multithreaded becomes the new method in real time system to accomplish system performance, power competence, etc. The presents work is a comparative study of various customized Multicore scheduling algorithms which make the most of system performance and resolves the real time tasks that can be managed without sacrilegious timing constraints. A main advantage of the model is that it delivers a fast and easy way to evaluate the system performance in real-time system and reflect tasks priorities which cause higher system utilization and lowers deadline miss time.

REFERENCES

[1] Xuanxia Yao, Peng Geng and Xiaojiang Du, "A Task Scheduling Algorithm for Multi-core Processors", International Conference on Parallel and Distributed Computing, Applications and Technologies, IEEE Xplore, ISSN: 2379-5352, DOI: 10.1109/PDCAT.2013.47, 2013.
[2] Xiaozhong Geng, Gaochao Xu, Dan Wang and Ying Shi, "A Task Scheduling Algorithm Based on Multi-Core Processors",International Conference on Mechatronic Science, Electric Engineering and Computer, IEEE Xplore,DOI: 10.1109/MEC.2011.6025620, 2011.
[3] Risat Mahmud Pathan, "Real-Time Scheduling Algorithm For Safety-Critical Systems On Faulty Multicore Environments", Springer, Real-Time System, DOI 10.1007/s11241-016-9258-z, 2017.
[4] Prerena Jaipurkar and Pranali D. Tembhurne, "A Comparitive Study of Different Customized Multiprocessor Scheduling Algorithms on Multicore Architecture", International Conference On Emanations in Modern Engineering Science and Management , ISSN: 2321-8169, Volume: 5 Issue: 3 25 – 29, 2017.
[5] Sinnen, O. and Sousa, L.A. : "Communication Contention in Task Scheduling," IEEE Trans. on Parallel and Distributed Systems, 16, 6, pp. 503-515, 2005.
[6] Sinnen O. and Kaur M., "Contention-aware scheduling with task duplication", Journal of Parallel and Distributed Computing 71(1), 77–86, 2011.
[7] Sinnen O., Sousa L., and Sandnes, F., "Toward a realistic task scheduling model", Parallel and Distributed Systems, IEEE Transactions on 17(3), 263–275, 2006.
[8] Gotoda S., and Shibata N., "Task scheduling algorithm for multicore processor system for minimizing recovery time in case of single node fault", In Proceedings of Cluster, Cloud and Grid Computing (CCGrid), 12th IEEE/ACM International Symposium on., 260–267, 2011.
[9] Park J and Dally W., "Buffer-space efficient and deadlock-free scheduling of stream applications on multi-core architectures", In Proceedings of the 22nd ACM symposium on Parallelism in algorithms and architectures, 2010.
[10] Julian Bui and Chenguang Xu, "Understanding Performance Issues on both Single Core and Multi-core Architecture" IEEE Transaction Parallel Distribution System, pp. 599–611, 2009.
[11] C. Cao Minh, J. Chung, C. Kozyrakis, and K. Olukotun, "Stamp: Stanford transactional applications for multiprocessing", in IEEE International Symposium on Workload Characterization, 2008.
[12] M. Diener, F. Madruga, E. Rodrigues, M. Alves, J. Schneider, P. Navaux, and H.-U. Heiss, "Evaluating thread placement based on memory access patterns for multi-core processors," in IEEE International Conference on High Performance Computing and Communications,sept., pp. 491–496, 2010.