RESEARCH PAPER | Engineering | Volume - 5 | Issue - 1 | Jan Special Issue - 2015 | ISSN - 2249-555X

cache performance enhancement through Hybrid LWFG partition algorithm in real-time operating system

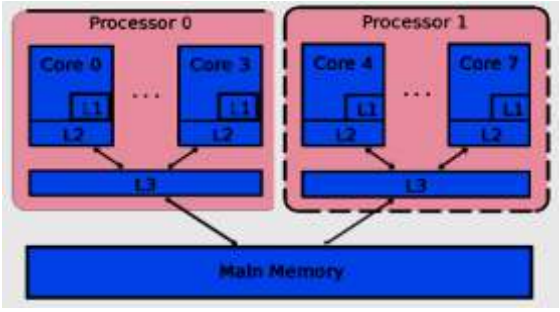
KEYWORDS: Multi-core; Hybrid LWFG algorithm; Real-Time scheduling;

Krupali A. Patel

ABSTRACT Today multi-core system having complex Hierarchical shared memory architecture which create Cache-awareness related problem like Cache misses, task tardiness, task migration. So the Problem of cache aware real-time scheduling on multiprocessor systems has been considered. Methods for improving real-time performance on multi-core platforms are task partitioning & global scheduling. Both scheduling scheme have problems .This research make hybrid scheduling combination of two schemes. The Existing LWFG cache-aware partitioning algorithm cause problem like to schedule tasks which share memory with one another in such a way as to minimize the total number of cache misses & bin packing problem. This research implement HLWFG(Hybrid Largest Working set size First, Grouping) algorithm which overcome problems like task migrations, deadline satisfaction ratio ,task execution efficiency, bin packing problem of LWFG algorithm. HLWFG implement in cluster environment which solve memory sharing problem & bin packing problem. The title evaluation shows that in some cases, the HLWFG partitioning algorithm increases, execution efficiency & minimize task migration..

I. INTRODUCTION

Nowadays, multi-core platforms are widely used in various high performance demanding applications, due to the limitations of the single core platforms. Multi-core platform offers various benefits. [1] present Due to optimization of real-time performance on multi-core systems poses new challenges, which are specifically related to the effects of complex interaction among cache memories Today Hierarchical caches are the most common type of memory architecture.



[2]Figure-1 memory architecture

In hierarchical cache architecture, there are several layers of cache between the processing cores and the shared physical memory. As you move from the cache nearest the processing core to that nearest main memory, the caches typically become larger, slower, and are shared between more cores.[2] Figure- 1 shows a typical example of hierarchical cache organization. In this paper multi-core real-time scheduling algorithms, implementation oriented issues and the impact of operating- system and cache overheads have been considered.

1.1. Real-Time Task Scheduling

Real-time systems depends not only on the results of their computation, but also on the time at which the results are produced. The main objective of a fast system is to minimize the average response time of a set of tasks, while the objective of a real- time system is to meet the timing requirements of each task. Scheduling is the process of telling which task should be executed on a particular processor during a specific time interval. [4] In extension this task is given access to all the resources that it requires such as memory, I/O devices and locked variables. The main reason for scheduling is to improve the performance of the system as whole. Real-time tasks have a set of parameters that are of interest for us in terms of scheduling, and they are:

- C, the worst case execution time of a job.
- T, the period of a job, which is only applicable to periodic tasks.
- D, the relative deadline of a job.
- r, release time of the first job of a task.

Hence the main objective of the scheduling algorithm should be allocated to the available resources to the arriving tasks in the best possible manner, which in turn improves the performance of the system. The main advantage of a multi-core system where we have more resources to provide for a task, depending on the type scheduling we are using. Another advantage is that the number of context switches is decreased. Context switch is the process where the processor stops the execution of a job, stores its present state and then loads the new pre-empting job into its present address space. The job that was pre-empted is later restored to the point until it was executed previously and continues execution when it is possible to schedule that particular job.

II.RELATED WORK

Multi-core Real-Time scheduling algorithm classify in to three terms(1) Global scheduling algorithm(2) Partition Scheduling algorithm(3) Hybrid Scheduling algorithm.

2.1 Global Scheduling

In this approach there will be a single global queue of ready tasks, from which tasks will be allocated to processor as per scheduling processing for the execution. The task will hold the allotted processor until it finishes its execution or until it is being pre-empted by the scheduler to execute a higher priority task. In the case of pre-emption the task after being pre-empted will be added to the global queue and then it will be scheduled. Author[3] presented mega task scheduling algorithm which have task migration problem

2.2 Partition Scheduling

Task migration and task preemption cost are very high in global scheduling algorithm. So, due scalability concern global scheduling algorithm is not sufficient. Partition scheduling means all the tasks are distributed to particular

Volume - 5 | Issue - 1 | Jan Special Issue - 2015 | ISSN - 2249-555X

processor and processor wise tasks ready queue will be generated. In partition algorithm task migration are not allowed. Hence obviously there is no consideration for overhead or delay due to communication between processors.

2.3 Hybrid Scheduling

This scheduling algorithm was developed for soft real time systems having hierarchical shared cache. This hierarchical shared cache may likely exist in many core platforms in order to avoid cache contention. This approach is a combination of both the partition and the global scheduling approach.

The Hybrid approach considered here is based upon the Earliest Deadline First scheduling policy. In this Scheduling approach, there are three steps;

- Dividing the system into clusters of cores,
- Allocation of tasks to these clusters by partition based allocation Algorithm
- Scheduling of tasks within a cluster using a GEDF algorithm.

HEDF is the combination of both partition and GEDF approaches In cluster architecture system is partitioned such that the cores that are sharing a common cache a grouped into one cluster. The number of cores in a cluster is the cluster size. Since a single cache is shared by cores in a cluster, the migration cost within a cluster is time taken to access the shared cache. As we are using cluster instead of a single core we are reducing the bin pack limitation. By adjusting the cluster size in accordance with the given task set, we can either lower migration cost or improve bin-packing. Ideal cluster size depends on both maximum task utilization and task set size. If the task utilization is high we can have a large cluster size to improve bin packing and if the task utilization is low then we can have lower cluster size in order to minimize the migration cost. There is an admission control mechanism in a approach which decides what tasks are going to be allowed into system in-order to get schedule and also provides mechanism to feasibly allocate tasks to a processor or group of processor such that each task

Figure-2 outline of HEDF algorithm

IIII. PROBLEM WITH THE EXISTING ALGORITHM

Author [11] propose LWFG-algorithm. Now problem can be identify using by the proper architecture of LWFG algorithm.

(a)In LWFG first tasks are sorted in desending order by task's WSS size.

(b)Then make group the tasks which shared memory with current task.

(c)Now group of tasks are allocated to particular core using next fit partition technique.

(d)If density of group of tasks is higher than core's density so it can be allocate to another core. At that condition task's migration are allowed

3.1 Problem with LWFG-algorithm

(a) If group of memory sharing task-set WSS size is large than core's size that can be allocated to another core. So, it cannot solve bin packing problem. Which is the main goal of partition algorithm?

(b) If sharing tasks on same core so all tasks have to wait for completion of executing task. So it will create problem like Core utilization are not properly utilized. This will decrease IPC rate. So cache awareness are not properly implemented.

(c) Tasks are scheduled using partition EDF. But PEDF having problem like at that time task scheduling only particular Local tasks of individual core are consider so if all task Completed their execution and if another core are executing tasks so at that time remaining tasks for execution Do not migrate to idle core. It means all the core's utilization are not properly utilized. So tasks response time will going to high

IV PROPOSE HYBRID LWFG ALGORITHM

This section describes how hybrid LWFG algorithm satisfies cache awareness property of existing LWFG algorithm. this chapter describe: How can we implement best partition multi-threaded real-time applications on platforms with complex memory/- cache hierarchies in order to decrease the execution time by minimizing cache misses and improving overall system efficiency? Restricting migrations makes more problem like bin-packing problem concern with task working set size because of there is no shred cache working structure is used. By limiting the maximum cache distance a task is allowed to migrate, these approaches decrease the potential for costly cache misses.

Goal of Hybrid LWFG-algorithm

(a)Develop strategy which solved the problem of bin packing because all the memory sharing tasks cannot be allocated to particular core. if we arrange cluster problem can be solved.

(b) Develop cluster method in which all memory sharing tasks can be allocated to cluster and task can be migrate within cluster. To solve problem related to PEDF algorithm for task scheduling.

(c)Develop Hybrid scheduling technique which is combination of global and partition algorithm and it overcome problem of global scheduling and partition scheduling.

(d) Maintain cache awareness property of LWFG algorithm while enhancing Hybrid LWFG algorithm

V. EXPERIMENTAL EVALUATION

In order to test the HLWFG partitioning strategy, we compared its performance to three of the the most widely used static partitioning schemes: LWFG partitioning scheme and HEDF using next-fit partitioning scheme. For this algorithm implementation Real-time tool has been generated in .net 2010 using internal SQL server. Result has been conducted in table form of internal SQL database. Author [17] developed Real-time tool which is defined through the specification of an execution environment, a workload and a scheduling strategy. In real-time tool working on multithreaded model in cluster environment. In Real-time tool task set generation can be specify by .XML file. In XML file different parameter like no. of task set, no. of tasks, LCM, Task utilization can be defined by user. After that in that tool task input files can be defined which .XML file is. In task input file total tasks defined per task set using period, execution time, deadline parameters. In this topic following task set has been specified. Following example shows that how taskset

Volume - 5 | Issue - 1 | Jan Special Issue - 2015 | ISSN - 2249-555X

will be generated.
1. tasks10_5_3_1 .xml file

```
<?xml version="1.0"?>
-<Tasks U="3.87722222222222" type="periodic">
<Task type="periodic" D="160" T="160" C="82" id="1"/>
<Task type="periodic" D="90" T="90" C="4" id="2"/>
<Task type="periodic" D="240" T="240" C="75" id="3"/>
<Task type="periodic" D="180" T="180" C="107" id="4"/>
<Task type="periodic" D="140" T="140" C="118" id="5"/>
<Task type="periodic" D="160" T="160" C="94" id="6"/>
<Task type="periodic" D="240" T="240" C="75" id="7"/>
<Task type="periodic" D="210" T="210" C="47" id="8"/>
<Task type="periodic" D="150" T="150" C="34" id="9"/>
<Task type="periodic" D="150" T="150" C="33" id="10"/>
</Tasks>
```

When run this algorithm on the tool first enter information like algorithm, processor, cluster you want.

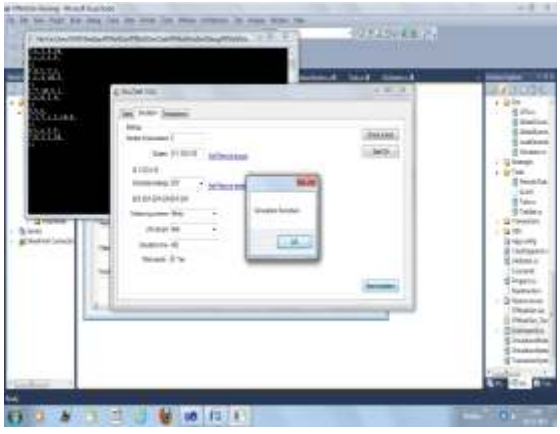


Figure-3 snapshot-1 real-time tool window

After entering all information then after clicking the simulation button all simulation result related data has been stored in database following snapshots show database entry

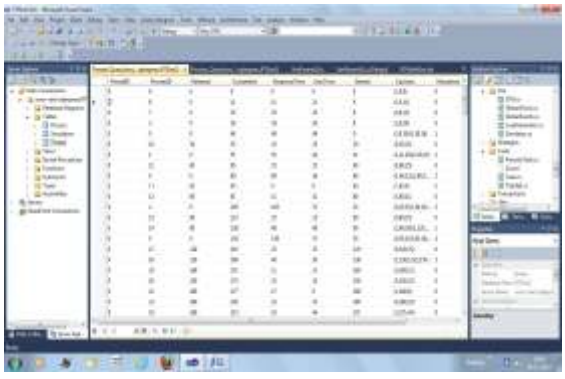


Figure-5 snapshot-2 thread table

In database result can be generated like how many task migration done in simulation?

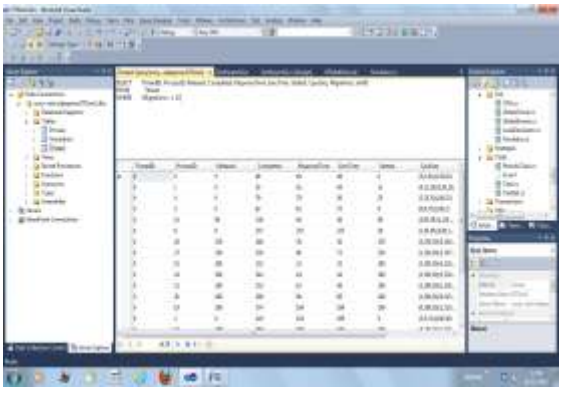


figure-6 snapshot-3 no. of migration in thread table

5.1Experimentalresult

Task-set: 5, No. of tasks per set: 10 , utilization: 3.9

Algorithm	Task- set no.	No. of Processor	No. of cluster	No. of migrations
LWFG	10_5_3	4	0	38
	8_3_2	4	0	14
HEDF using Next-fit	10_5_3	6	3	25
	8_3_2	6	3	12
HLWFG	10_5_3	6	3	20
	8_3_2	6	3	10

Result: In table-2 result has been conducted task-set wise. Following observation has been observed.

Obeservation:1 For task-set 10_5_3 In LWFG algorithm contain 38 tasks having migration & HEDF contain 25 tasks having migrations & HLWFG contain 20 task having migrations.

Obeservation:2 For task-set 8_3_2 In LWFG algorithm contain 14 tasks having migration & HEDF contain 12 tasks having migrations & HLWFG contain 10 task having migrations.

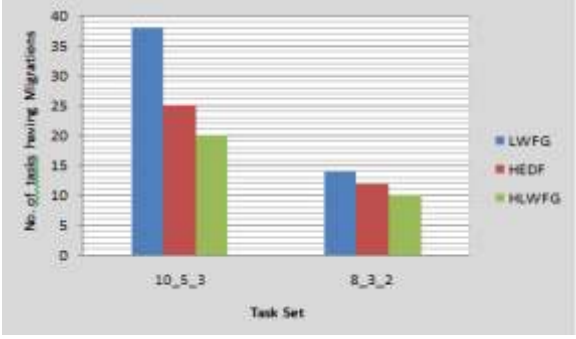


Figure-7 graph result

VII . CONCLUSION

In this paper HLWFG algorithm solve cache awareness problem like task migration. Using this algorithm bin-packing problem will be solved with shared memory architecture in Multi-core system. This paper define new definition of partition algorithm with global algorithm policy which reduced cache system overhead.

Volume - 5 | Issue - 1 | Jan Special Issue - 2015 | ISSN - 2249-555X

REFERENCE

[1] James H. Anderson, John M. Calandrino, and UmaMaheswari C. Devi. Real- time scheduling on multicore platforms. In Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium, pages 179–190, Washington, DC, USA, 2006. IEEE Computer Society. | [2] John M. Calandrino and James H. Anderson. Cache-aware real-time scheduling on multicore platforms: Heuristics and a case study. In Proceedings of the 20th Euromicro Conference on Real-Time Systems, ECRTS '08, pages 299–308, Washington, DC, USA,2008. IEEE Computer Society. | [3] John M. Calandrino and James H. Anderson. On the design and implementation of a cache-aware multicore real-time scheduler. In Proceedings of the 21st Euromicro Conference on Real-Time Systems, pages 194–204, July2009. | [4] J.C. Saez, J.I. Gomez, and M. Prieto. Improving priority enforcement via non- work- conserving scheduling. In Parallel Processing, 2008. ICPP '08. 37th International Con- ference on, pages 99 –106, sept. 2008. | [5] S.K.Dhall and C.L.Liu, "On a real-time scheduling problem," Operations Research, vol.26, no. 1, pp. 127-140, Jan/Feb. 1978. | [6] Y.Oh and S.H.Son, "Tight performance bounds of heuristics for a real-time scheduling problem," Technical Report (2-93-24), Univ. of Virginia, Dept. of Computer Science, May 1993. | [7] A.Burchard, J.Liebeherr, Y. Oh and S.H.Son, "New Strategies for Assigning Real- Time Tasks to Multiprocessor Systems," IEEE Transactions on Computers, 44(12), pp. 1429–1442, 1995. | [8] S. Davari S.K. Dhall, "On a Periodic Real-Time Task Allocation Problem", Proceedings of the 19th Annual International Conference on System Sciences, pp.133-141, 1986. | [9] J.M.Lopez, M. Garcia, J.L. D´iaz, and D.F. Garcia, "Worst-case utilization bound for EDF scheduling on real-time multiprocessor systems", In Proc. 12th EuroMicro Conference on Real-Time Systems, pp, 25–33, Stockholm, Sweden, June 19–21, 2000. | [10] H.Beitollahi, G.Deconinck, "Fault-Tolerant Partitioning Scheduling Algorithms in Real-Time Multiprocessor Systems," Proceedings of the 12th Pacific Rim International Symposium on Dependable Computing, p.296-304, December 18-20, 2006. | [11] Aaron C. LindsayLWFG: A CacheAwareMulti-core Real-Time multi-coreReal-Time SchedulingAlgorithm,june-2012. | [12] Y.Oh and S. H. Son, "Allocating Fixed Priority Tasks Multiprocessor Systems", Journal of Real Time Systems, pp. 207-239, 1995.[13] A. Block, J. H. Anderson, and U.M. C. Devi, "Task Reweighting under Global Scheduling on Multiprocessors," Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 2006. | [14] T. P. Baker, "Comparison of Empirical Success Rates of Global vs. Partitioned Fixed- Priority and EDF Scheduling for Hard Real Time," Technical Report TR-050601, Department of Computer Science, Florida State University, 2005. | [15] Real-time chron os linux wiki. https://rt.wiki.kernel.org/. AccessedFabruary 6, 2012. | [16] Ajitramachandran,JegadishManoharan&omanathanChandrakumar,"Real- Time Scheduling methods for High Performance Signal Processing Applications on Multicore platform ",2012 August