OPEN ACCESS

International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

Process Scheduling in Heterogeneous Multicore System using Agent based Graph Coloring Algorithm

G. Muneeswari , Antony Puthussery

***Abstract**: In any heterogeneous multicore system, there are numerous amount of processors with different platform and all the processing units are fabricated on a common single unit preferably on a System on Chip. As there is a tremendous amount of parallelism encompassed in a multicore system, proper utilization of the cores is a big challenge in the current era. Hence a more automated software approach is required like an agent based graph coloring algorithm to find the free processor and schedule the tasks on the respective cores. Predominantly the entire process of scheduling the tasks on multicore system is based on arrival time of process. This paper incorporates the scheduling on the linux 2.6.11 kernel and GEMS simulator for multicore implementation. The core utilization in this type of agent scheduling is 50% more than the existing scheduling mechanism.*

***Keywords:** Agent based graph coloring, Processor allocation, Heterogeneous Multicore system, scheduling, Process classification.*

I. INTRODUCTION

Multicore system is a new innovation and a very big alternate to sequential processing. Massive parallel processing is the greatest advantage of this SoC processors. However, lot of challenges imposed towards the design of many core system. Predominantly the legacy code, compiler modification, operating system scheduling are the major areas of consideration for multicore design. In homogeneous multicore system, process scheduling has little complexity rather than the heterogeneous multicore processors in the sense that we cannot allocate the process based on affinity based scheduling. Considering all the difficulties involved with hardware and software, a more efficient approach is required for heterogeneous processor scheduling. A more intelligent approach is to use an agent inside the operating system scheduler. The basic idea is to incorporate an agent based graph coloring algorithm to find the number of processor required to execute the given set of processes from the ready queue. Each color from the resultant graph will represent the unique heterogeneous processor and the overlapping time interval of process is resolved by allocating the interleaving process to different cores. The overall paper organization is as follows. In section II, the related work is discussed and in section III, an insight into working methodology is proposed. Section IV describes the results and discussions and finally section V concludes the paper.

Manuscript published on 30 September 2019

* Correspondence Author

Dr.G.Muneeswari*, Department of Computer Science and Engineering, Faculty of Engineering, CHRIST (Deemed to be University), Bengaluru, India. Email: muneeswari.g@christuniversity.in

Antony Puthussery, Department of Science and Humanities, CHRIST (Deemed to be University), Bengaluru, India. Email:frantony@christuniversity.in

II. RELATED WORK

The algorithm starts from a value equal to m(G), calculated as given in Algorithm 1 [1]. A self-stabilizing edge-coloring algorithm [2] was proposed by Kauther and Dekar which works well for small graphs. Another method to solve the graph coloring problem was introduced by Wang and Qiao which use a non-monotonous activation function [4]. With regular graphs [5] In [6], it was proved that, this bound can be lowered to 2d3, that the b-chromatic number on an arbitrary d-regular graph with girth g=5 is at least (d+1) / 2, and that every d-regular graph, where d is greater or equal to 6, with a diameter at least d and with no 4-cycles can be b-colored with d+ 1 colors. Outer planer graphs [7,12] also taken for scheduling criteria. Some works even tried finding its exact value for some types of graphs [8]. There are many b-coloring based applications, such as clustering, web analysis, information retrieval, and medical diagnostics [10]. The authors of [11] enhanced this result when showing that $\varphi(G)=d+1$ for a d-regular graph G that contains a diameter ≥ 6 and no cycle of order 4. The hard and soft affinity scheduling and the related performance metric is discussed in [13]. In [14] an equitable antimagic labelling approach with minimum running time has been discussed.

III. WORKING METHODOLOGY

The main idea behind this proposed work is to find an efficient scheduling algorithm for processor allocation using intelligent agents. The core concept is divided into three sub components wherein the ultimate goal is to improve the processor utilization in heterogeneous multicore system. The first component is the main processor allocation algorithm which is incorporated with Process_Classify function and Agent_Coloring function. Initially the set of processes with different arrival time is taken from the ready queue of memory by the scheduler.

All the three components as part of the graph coloring algorithm is incorporated as part of the scheduler. The automatic agent scheduling is required to reduce the average waiting time of the process in the ready queue.

In Fig.1, For example P1 to P10 process with three sets of interleaving arrival time is taken for evaluation.

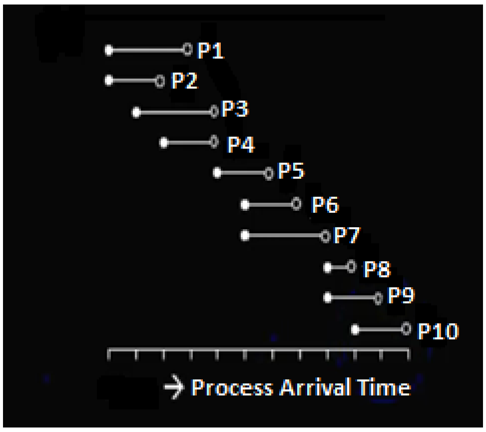

Fig. 1. Process Arrival Time

The main algorithm for processor allocation is shown in Fig.2.

Main Algorithm for Processor Allocation:
For every new process Pi to Pn perform the following operation:
Begin

Step 1:	Parent Agent (PA) is initialized and interacts with the scheduler of operating system.
Step 2:	PA gets the number of process **Pi to Pn** from the ready queue
Step 3:	PA obtains the burst time **Bi** and arrival time **Ai** for all the processes **Pi to Pn.**
Step 4:	PA classifies all the processes based on arrival time by calling the algorithm using function **Process_Classify(ATi)**
Step 5:	PA gets the process sets from the Process_Classify function.
Step 6:	Every set of processes are represented as a different graph (Gi) and colored using **Agent_Graph_Coloring (Si)** function.
Step 7:	Each color will be designated as a different processor PR-i in the heterogeneous multicore system.
Step 8:	Agent with the same color from different set can be assigned to the same core in the multicore system by the Parent Agent.
Step 9:	Repeat steps 7 and 8 until no more agent is left out in the graph set.
Step 10:	Parent Agent schedules all the process (Pi to Pn) to the respective processor which then starts execution.

Fig. 2. Main Algorithm for Processor Allocation

The Parent Agent (PA) is an intelligent which schedules the tasks on different heterogeneous cores. It calls another component Process_Classify function which returns a graph with set of processes P1 to pn. The outcome of this component is shown in fig.3 with different graphs with different set of processes. Every vertex of the graph is designated as Agent Vertex.

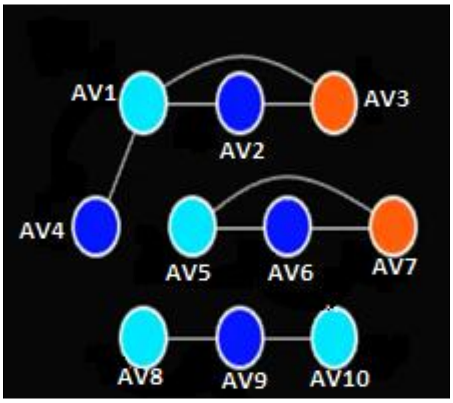

Fig. 3. Process Classification and Graph Coloring

In Fig.4, The detailed algorithm for Process Classification is depicted. After the process classification is enabled, the different graphs are constructed with each vertex is being nominates as an Agent Vertex.

In our example shown in Fig.1, Process P1, P2, P3, P4 have overlapping time interval. So they can be represented as the first graph. Secondly Processes P5, P6, P7 have again the interleaved time interval. They will be put up in a different graph set. Finally, Process P8, P9, P10 have the overlapping time interval. So, they again form a separate group of graph set. In the NUMA architecture, affinity based scheduling does not required as the same configuration is implemented in every core. But in the heterogeneous multicore system, affinity based scheduling must be incorporated at the back ground during context switching. Both soft and hard affinities can be incorporated along with agent coloring scheme. Suppose if process P1 executed in Core1 before context switching, then P1 should be scheduled on same Core1 after context switching also. Process priority is not considered in the evaluation as we do not bother about the real time task.

Ideally the proposed algorithm design is much simpler compared to the existing scheduling algorithm for multicore system since there is no much complexity involved with the hardware design. The average waiting time of the process is calculated based on the arrival time of all the process in the ready queue. This parameter is taken as a performance metric for evaluating the proposed algorithm. Finally, none of the processor are kept in the idle state as and when large number of processes are waiting in the ready queue.

Algorithm for Process Classification:
For every new process Pi to Pn with Arrival Time Ai perform the following operation:
Process_Classify (Pi, ATi)
Begin

Step 1: Process Classify Agent (PCA) gets the input of Process ID (Pi) and Arrival Time (ATi).
Step 2: For each process Pi
Begin
Step 2.1: PCA finds out any other process overlaps with the Arrival Time (ATi) of Process Pi
Step 2.2: If there is an overlap then, it will be put up in the same set Si by the PCA.
Step 2.3: Else (no overlap), then another set Si is framed by PCA.
Step 2.4: Repeat steps from 2.1 to 2.3 until no more process in the list.
End
Step 3: PCA obtains the different set of processes S1 to Sn and assigns every process Pi from set Si to an Agent Ai.
End

Fig. 4. Process Classification Algorithm

Based on the arrival time Process Classify Agent (PCA) identifies the different sets of processes. Finally, the different processors are identified with different colors and there is no overlapping of processes within the same set of processors. It is illustrated with a diagram in Fig.5.

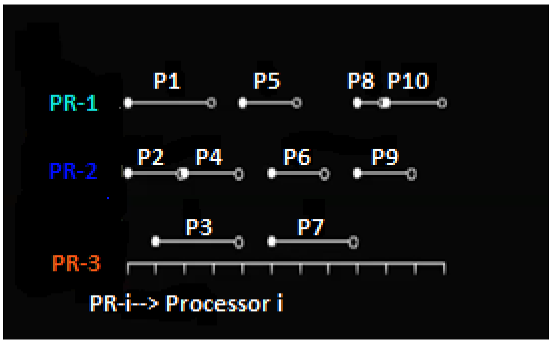

Fig. 5. Processor allocation based on coloring

The agent graph coloring algorithm is designed with a Vertex Coloring Agent (VCA) whose functionality is to assign the color to the Agent Vertex. The constraint for this algorithm is that no adjacent vertices are colored with the same color with the minimum chromatic number.

The final algorithm for graph coloring is elaborated in Fig.6.

Algorithm for Graph Coloring:
For every new set Si perform the following operation:
Agent_Graph_Coloring (Si)
For each Agent Ai from set Si
Begin
Step 1: A Vertex Coloring Agent (VCA) assigns any random color for the first Agent Vertex (AV1).
Step 2: From each set Si process is taken and colored with some random color.
Step 3: Repeat the coloring method with a constraint that no adjacent process is colored with the same color.
Step 4: Repeat the same method for all the Agent Vertex in each set Si for all processes **Pi to Pn.**
Step 5: Every color is represented as a different core in HMC system.

End

Fig. 6. Agent Graph Coloring Algorithm

The final chromatic number would be the number of cores in the heterogeneous multicore system.

IV. RESULT AND DISCUSSION

For the evaluation results Linux kernel version 2.6.11 along with FLAME tool is used for agent based scheduler design for operating system scheduling. Table-I shows the number of heterogeneous cores and the average waiting time.

Table- I: Number of cores vs Average Waiting Time of processes

Total Cores	Average Waiting Time of Processes(ms)
10	9.4
20	8.4
30	8.2
40	7.5
50	6.3
60	5.6
70	4.2
80	3.7
90	2.6
100	1.2

From this analysis, it is found that the number of core utilization is very high with the proposed agent based graph coloring algorithm.

Fig.7 shows the processor utilization in the heterogenous multicore system with the comparative study between affinity based scheduling and agent based graph coloring algorithm.

Retrieval Number C6410098319/2019©BEIESP
DOI: 10.35940/ijrte.C6410.098319
Journal Website: www.ijrte.org

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

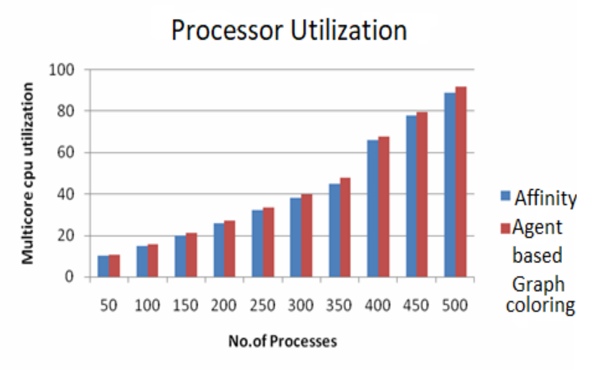

Fig.7. Processor utilization in heterogeneous multicore system using agent based graph coloring algorithm

V. CONCLUSION

The proposed methodology of agent based graph coloring algorithm is a simplified and powerful mechanism for thread scheduling in heterogeneous multicore system. From the evaluation results we proved that as and when the number of cores are increased, then the utilization of the cores are 50% more than the existing scheduling algorithm since we considered the arrival time of the process for the evaluation. a conclusion our new agent thread assignment algorithm eliminates the complexity of the hardware and improved the CPU utilization to the maximum level. The average waiting time of the process is tremendously reduced in the proposed novel algorithm.

REFERENCES

1. S. Labed, A. Kout, S. Chikhi, "Solving the graph b-coloring problem with hybrid genetic algorithm", 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS) Algeria, October 2018.
2. Drira Kauther, Dekar Lyes, Kheddouci Hemamache, "A Self-Stabilizing (delta+ 1)-Edge-Coloring Algorithm of Arbitrary Graphs", International Conference on Parallel and Distributed Computing Applications and Technologies PDCAT, pp. 312-317, 2009.
3. M. Jakovac, S. Klavzar, "The b-chromatic number of cubic graphs", graph Combin., vol. 26, pp. 107-118, 2010.
4. Wang Xiuhong, Qiao Qingli, "Solving Graph Coloring Problems Based on a Chaos Nueral Network with Non-monotonous Activation Function", Fifth International Conference on Natural Computation ICNC, vol. 1, pp. 414-417, 2009.
5. A. El Sahili, H. Kheddouci, M. Kouider, M. Mortada, "The bAchromatic number and f-chromatic vertex number of regular graphs", Discrete Appl. Math., vol. 179, pp. 79-85, 2014.
6. S. Cabello, M. Jakovac, "On the b-chromatic number of regular graphs", Discrete Applied Mathematics, vol. 159, pp. 1303-1310, 2011.
7. F. Maffray, A. Silva, "b-colouring outerplanar graphs with large girth", discrete Math, vol. 312, pp. 1796-1803, 2012.
8. D. Francis Xavier, "b-Chromatic Number of Line Graphs of Certain Snake Graphs", International Journal of Computing Algorithm, vol. 03, pp. 700-703, February 2014.
9. R. Javadi, B. Omoomi, "On b-coloring of the Kneser graphs", Discrete Math., vol. 309, pp. 4399-4408, 2009.
10. D. Gaceb, V. Eglin, F. Lebourgois, H. Emptoz, "Robust Approach of Adress block Localization In Business Mail by Graph Coloring", The international Arab Journal of Information Technology, vol. 6, july 2009.
11. S. Shaebani, "On the b-Chromatic Number of Regular Graphs without 4-Cycle", Discrete Applied Mathematics, vol. 160, pp. 1610-1614, 2012.
12. M. Kouider, M. Zaker, "Bounds for the b-chromatic number of some families of graphs", Discrete Math., vol. 306, pp. 617-623, 2006.
13. Muneeswari.G, Shunmuganathan K. L, "A Novel Hard-Soft Processor Affinity Scheduling for Multicore Architecture using Multiagents" European Journal of Scientific Research, Vol.55 No.3, pp.419-429, 2011.
14. A. Puthussery, I.S Hamid, A. Anitha, "An equitable antimagic labeling of graphs: Algorithmic Approach", International journal of Computer Sciences and Engineering, Vol 7, Issue5, pp: 846-851, 2019.

AUTHORS PROFILE

Dr.G.Muneeswari is working as an Associate Professor in the department of Computer Science and Engineering, CHRIST (Deemed to be University). She has more than 20 years of teaching experience in various institutions. She has more than 50 publications and her research area focused on scheduling in multicore system.

Fr.Antony Puthussery is working as an Assistant Professor in the department of Science and Humanities and campus director in CHRIST (Deemed to be university). He has many research publications and his area of research is inclined towards graph theory and mathematical modeling.