xIlkogretim Online - Elementary Education Online,2021; Vol 20 (Issue 1): pp.2179-2185
http://ilkogretim-online.org
doi: 10.17051/ilkonline.2021.01.240

Energy Efficient Task Scheduling Algorithms for Performance Asymmetric Multicore Architectures

Gomatheeshwari B, Department of Electronics and Communications Engineering, College of Engineering and Technology SRM Institute of Science and Technology, SRM Nagar, Kattankulathur, Kanchipuram, Chennai, Tamil Nadu, India, gomatheeshwari_balasekaran@srmuniv.edu.in
Selvakumar J, Department of Electronics and Communications Engineering, College of Engineering and Technology SRM Institute of Science and Technology, SRM Nagar, Kattankulathur, Kanchipuram, Chennai, Tamil Nadu, India, selvakumar.j@ktr.srmuniv.ac.in

Abstract-Attaining high-performance and power efficiency has become a critical issue in modern embedded systems. To address this issue, computer architects have integrated the single-ISA heterogeneous cores on the same chip known as asymmetric multicore processors (AMPs), which also satisfies the diverse computational requirements efficiently in real-time. Despite its benefits, resource administration in terms of scheduling and mapping the tasks onto appropriate cores become a challenging task. In this paper, we targeted such AMP systems for emulation and develop an intelligent scheduling heuristic for MiBench workloads execution. The proposed semi-dynamic energy-efficient scheduling algorithm (SD-EESA) comprises workload modeling, prioritization, and mapping stages. Workloads are modeled as directed acyclic graphs (DAG) with combination of nodes and edges. In the second phase, these nodes are prioritized in the non-increasing order of its cost value and allocated to active processors based on the online profiling data in terms of core utilization, which is a novel method compare to traditional models. The simulated environment shows an improvement of 30.6% and 14.6% in overall execution time and 34.9% and 19.2% in energy reduction for MiBench workloads than traditional heterogeneous earliest finish time (HEFT), Robust-HEFT.

Keywords: DAG, Energy-Time tradeoff, Periodic tasks, Performance Asymmetric multicore, and SD-EESA technique

I. INTRODUCTION

In recent days, Multi-core architectures are extended from homogeneous to heterogeneous structures. Heterogeneous multi-core is the combination of computational cores, graphical cores and accelerators are integrated into the same system on a chip. Many high-end applications make use of the multi-core architectures to attain parallel processing and high performance [1]. Parallel processing in such multi-core architectures requires efficient scheduling algorithms to control and manage the execution of multiple tasks in the system-on-chip. The problem identified in such parallel processing is energy consumption [2]. Power dissipation of such multi-core circuits includes the sum of leakage and switching power which consumed more during the processing of large computational tasks. This high power dissipation leads to system failure and poor performance in real-time [3]. Intel manufactures survey shows the power dissipation [4] of each core leads up to 35-40W and consumes[5], 1$ per watt in the multi-core system-on-chip [6].This technique increases the total electricity cost grow to 35% in 2020 for high-end applications [7].This paper addresses the design and deployment of the effective scheduling method known as semi-dynamic energy efficient scheduling algorithm (SD-EESA) for periodic tasks in asymmetric multicore to the tradeoff of time and energy constraints concurrently.DAG Modeling phase - It modeled the periodic tasks into the directed acyclic graph structure with its attributes. A vertex of the graph represents the task and links between two vertices represents the communication cost. Periodic workloads are transformed into DAG model using TGFF tool. Task prioritization phase –Periodic task graphs are prioritized in non-increasing order based on the cost value of each task, which is calculated using equation 2.Core mapping phase -This phase dynamically selects the processors for each task execution based on the online profiling data of the active cores at run-time. In this article, we adopted the DVFS technique to manage the operating frequency and voltage based on the load at each processor. The remaining sections are, sections two explains about the related survey and section 3-System model and application model and in part 4 explains about the SD-EESA framework and chapter 5, 6 is simulation environment with results and discussion and Conclusion.

II. RELATED WORK

This section briefly describes related research in the same field of task scheduling, which modeled tasks into multiple Dag formats with various parameters. Task scheduling in heterogeneous systems described

in detail. HEFT and Critical Path on processor [12], Robust- HEFT [14], Duplication based HEFT [15], all are traditional task scheduling methods for heterogeneous systems. In HEFT [12], authors developed a rank based schedule which includes two stages (i) Task prioritize phase which arranges tasks as per its rank value into a ready queue and (ii) Map the tasks in the available processors as per round robin format. CPOP scheduler calculates both upward rank and downward rank value to prioritize the tasks and followed the HEFT for execution. Robust HEFT [14] is a planning algorithm which splits the following functions into independent task sets and executes independently to improve utilization ratio and Make span. Duplication based HEFT utilizes the slack time for duplication tasks and improves the concept of mapping with feedback data [15]. These scheduling algorithms considered total system execution time minimization of multi-core systems.AI-based scheduling techniques such as ant colony optimization algorithm; genetic algorithm and particle swarm intelligent were developed to optimize the system execution time by reducing the search space of optimal solution for multi-core scheduling [8].

III. Design Principles

A. Periodic DAG Model

An implicit periodic task modeled as a directed acyclic graph (DAG). A DAG is classically defined as G = (V, E) combination of vertices and edges. For our scheduling problem, we changed the regular DAG graph into G'= (V, E, C_i, p_i, d_i) where V represents the task nodes and E represents the edge cost between V_i and V_j when both tasks are allocated in a different processor [9]. $C_i'$is the computation cost matrix of V*M in which C_{ij} is the estimated execution time of task V_i on M_j.pi is the inter-arrival time of each task (Period), and d_i is the relative constrained deadline ($d_i < p_i$) and if $d_i = p_i$ then it is called implicit deadline. Such (V_i) is an immediate successor of task V_i, and pred (V_i) is the direct predecessor of task V_i. A node without a successor is known an exit task, and a node without predecessors is known as an entry task. For example, a MiBench workload named as recognition code is modeled as a ten-node graph with its C_{ij} and D_{ij} showed in Figure.1

B. System Model

In this article we considered performance asymmetric multicore structure with three cores represented as M ε {M_1, M_2, M_3}. Each core is having the same instruction set architecture but differ in its pipelined, operating frequency, and voltage. We assumed the cache, I/O and other multicore resources are included in the core. Operating frequency and voltage levels such as (f_{min}, f_{max}) and (V_{min}, V_{max}) are denoted by the user. In this paper, we assumed the voltage value is varied from 0.2 to 0.85 and frequency values are varied in 1.0GHz to 1.8GHz respectively [10].

IV. Proposed Methodology

The proposed framework SD-EESA comprises the workload modeling phase, which initially structured the periodic tasks in terms of computation costs as nodes and communication cost as edges. Likewise, the MiBench workloads are modeled with different degrees of parallelism. Each periodic task is generated with the respective constrained deadlines and its job instance units (period). In the first phase, we calculate the worst-case execution cost as per equation (1) based on the core frequency. To utilize the heterogeneity nature of cores, we initially calculated the core utilization factor at every't' sampling period before allocating each task at runtime [11]. The tasks are allocated to the appropriate processors and executed before its deadline. For example, a k-task with utilization is 30% ready for execution, and the three different core utilization factors are 40%, 20%, and 10% utilized at the arrival time of k-task. The proposed algorithm will allocate that task into least utilized core (i.e., core with 10%) to achieve the deadline and also minimize the scheduling overhead (waiting time).

A. SD-EESA Framework

The estimated average execution time of each task'V_i' on the available processor 'M_j' is denoted as C_i, ' and cost values are given in Table 1. Periodic tasks are varied from 1 to 'n' and cores are varied from 1 to m where i denote the task and j denotes the core in an execution cost matrix [13]. The data transfer time between two cores denoted in E_{ij}' of size M*M. Communication cost matrix between two tasks is represented as an edge (i, j). This edge cost is obtained during the transmission of data from memory to cores.

$$C_i' = \sum_{j=1}^{M} C_{ij}/M \quad (1)$$

$$Eij' = N + \frac{data\,(i,j)}{E(m,n)} \quad (2)$$

$$rc(V_i) = C_i' + max_{Vj \in succ\,(Vi)}(E_{ij}^{'} + rc(V_j)) \quad (3)$$

$$rc(V_{exit}) = C_{exit}^{'} \quad (4)$$

WhereE_{ij} ′ the average communication cost is 0 if M_w= M_k and C_i' is the standardexecution cost of taskV_i. Equation 3 illustrates the upward rank value of each task if traversed from V_{exit} to V_{entry} tasks [12].

B. Periodic Task Prioritize Phase

Each task is prioritized based on the estimated optimized rank cost value [20]. The optimized rank value is calculated based on the summation of the tasks on the allocated core is divided by the total number of processors.

$$ORT(V_i) = \frac{\sum_{K=1}^{M} rc\,(V_i, M_k)}{M} \quad (5)$$

Periodic tasks are prioritized in decreasing order in task prioritization stage. Table 1illustrates the cost value of sample periodic task graph. Task V_4 is prioritized before task V_2, and task V_{10} is the least cost value, which is the lowest priority task

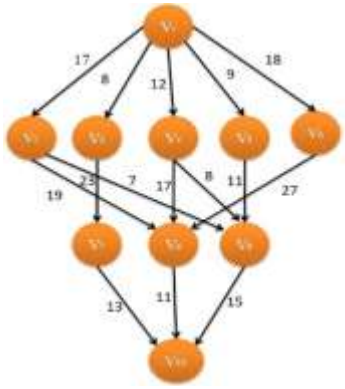

Figure 1. Periodic DAG

C. Core Mapping Phase

We assumed asymmetric multicore architecture as targeted for the execution of periodic tasks. Ready tasks are mapped into active cores based on the online profiling data such as core utilization factor which is updated every 4ms at run-time. Core utilization factor is varied in two conditions (i) task V_i utilized the core M_j for T_i time units and (ii) total utilization of core j at T_i time units. We checked both the conditions at run-time to identify the appropriate processor for each task V_i. Core utilization data's are defined in two matrices. First matrix $ATU_{M_k}^{t_i}$, illustrates the task utilization value on each core at t_i separately. For example, a task V_1 utilized core 1 for 3ms and same task consumed 8ms by core 2 at time instance t_i. This matrix is calculated as per equation 5. The second matrix shows the core utilization at time instance t_i which is updated for every 4ms. We assumed the upper bound of core util. is less than 100. The upper bound of the execution cost is 50ms and lower bound of execution cost is 1ms.

Table 1. Task Prioritization Phase

Tasks	M1	M2	M3	Rank cost (RC)	Optimized rank cost(ORC)
V1	12	6	12	93	31
V2	7	12	3	58	19.3

V3	4	10	19	45	15
V4	12	19	15	62	20.6
V5	11	9	4	49	16.3
V6	17	4	9	65	21.6
V7	3	10	10	27	9
V8	11	13	12	28	9.3
V9	12	8	15	30	10
V10	6	5	2	4	1.3

$$ATU_{M_k}^{t_i} = \sum_{i=1}^{n} \frac{C_i^{'}}{p_i} / f_k \quad (6)$$

$$TU_{M_k}^{t_i} = \sum_{k=1}^{M} ui_{Mk}^{ti} \quad (7)$$

$$Core\ util.\left(Mk^{ti}\right) = TU_{Mk}^{ti} + ATU_{M_k}^{t_i} \quad (8)$$

$$CS\ (V_i, M_k) = C_i^{'}(V_i) \leq core\ util.\left(Mk^{ti}\right) \quad (9)$$

Based on this core utilization factor value, we mapped the ready tasks at run-time. It guaranteed the task execution and reduced the scheduling overhead.

D. Energy Model

Power dissipation in multi-core architecture is the sum of static power and dynamic power [15]. More energy consumed during the computational process and total power consumption is given in equation (10).

$$P_{dyn} = ceff_k * {V^2}_k * f_k \quad (10)$$

$$EC_{ij} = \sum_{i \in N, j \in M} P_{dyn} * {C_{ij}}' \quad (11)$$

$$Makespan\left(ET_{sys}\right) = Max\left(AFT\ (N_{exit})\right) \quad (12)$$

$$CS\ (V_i, M_k) = C_i^{'}(V_i) \leq core\ util.\left(Mk^{ti}\right) \quad (13)$$

***Algorithm 1**: SD-Energy Efficient Scheduling Algorithm (SD-EESA)*

---

1. *Core initialized with its voltage and frequency.*
2. *Calculate* $C_i' = \frac{C_{ij}}{f_j}$ *to find estimated execution time of task i on core j.*
3. *Calculate* $E_{ij} = N + \frac{data\ (i,j)}{E(m,n)}$ *data transmission cost*
4. *for i = 1 to m do:*
5. *for j= 1 to m do:*
6. $rc(V_i) = C_i^{'} + max_{Vj \in succ\ (Vi)} \left(E_{ij}^{'} + rc\left(V_j\right)\right) \quad \forall\ i \in N$
7. $rank\ cost\ value(N_i) = \frac{\sum_{k=1}^{M} ORT\ (V_i, M_k)}{M}$
8. *end*
9. *end*
10. *Tasks are sorted in decreasing order and inserted into ready queue*
11. *Calculate the utilization of each task* $u_i = \frac{C_i'}{p_i}$.
12. *Calculate the actual core utilization at run-time*

$$ATU_{M_k}^{t_i} = \sum_{i=1}^{n} \frac{C_i^{'}}{p_i} / f_k \quad (14)$$

$$TU_{M_k}^{t_i} = \sum_{k=1}^{M} ui_{Mk}^{ti} \qquad (15)$$

13. *for j = 1: k do:*
14. *Update the* $Core\ util.(Mk^{ti})$ *every* t_i *time units.*
15. $Core\ util.(Mk^{ti}) = TU_{Mk}^{ti} + ATU_{M_k}^{t_i}$
16. *end*
17. *If* $C_i'(V_i) \leq core\ utilization\ (Mk^{ti})$ *: Task assigned to the current core for execution.*
18. *Else the task is sent back to the re-mapping queue. end*
19. *Repeat for and if loop until all tasks are allocated at least in one core.*
20. *Tasks are re-mapped to the cores in a dynamic model based on its utilization requirement.*
21. *for j=1: M*
22. *Calculate* $(ET_{sys}) = Max\ (AFT\ (N_{exit}))$
23. *Energy consumption is obtained using equation 12.*
24. *end*

V. EXPERIMENTAL SET-UP

Raspberry pi-3 b+ SoC is used for experimental verification which includes quad core multicore as ARM cortex-A53 (Armv8) with 1.4 GHz maximum frequency, 1 GB internal memory, 64GB external SD card, 5V/2.5 A DC Power, 2.4 GHz IEEE 802.1 Ethernet support, USB ports, camera, DSI display etc. The operating frequency variation assumed in between {0.8-1.4 GHz} and voltage {0.85-3.3V} in steps of 2.IoMT workloads such as "histogram, sensor tasks, aes, cryptography tasks" are used for evaluation. We also used sensor programs which developed on arudino board such as temperature, pressure, humidity, camera sensing code, etc. , on the real-time with regular period and deadline metrics. Each task is assumed with individual deadline and period (inter-arrival time) in the range of {min, max} values. TGFF [12] is used to generate synthetic task graphs with limited nodes. Periodic task features such as several instructions in a program, arrival time, computing time, data rate, period, and deadline are pre-defined for synthetic data sets, which details are given in table 2.

VI. SIMULATION ENVIRONMENT

Simso [24] is a Python based multiprocessing scheduling simulator which is utilized for developing the 'SESA' in Ubuntu-18.04 OS with Linux kernel and the memory is 64GB. Psutils and perf tools are used for performance observations in terms of total execution time, core utilization, energy consumption with average power consumed.

Table 2. Parameters of Task Graph using TGFF

Parameters	Range
No. of. task nodes	{5-20} in each graph
No. of. edges	{2-10} are varied randomly
Deadline and period	{0.1s-15s} and {0.5s-25s}
CCR, out degree,	{0.1-3}

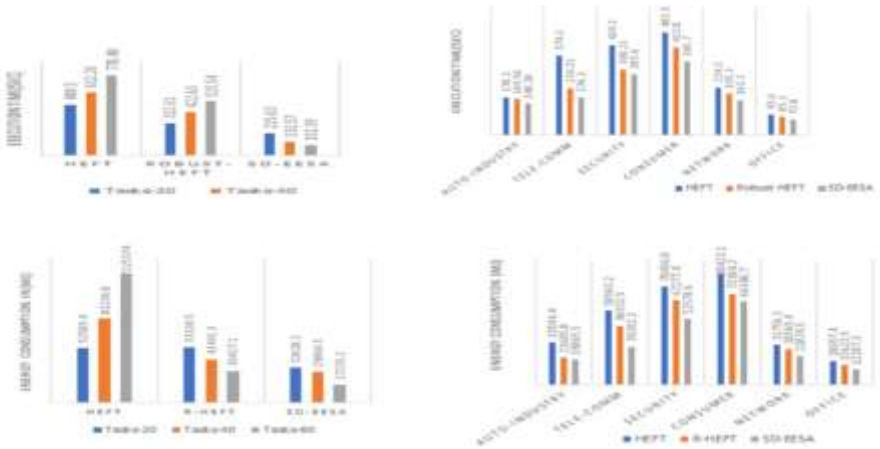

Figure 2. Performance of Proposed Mapping Techniques for MiBench Workloads

VII. Performance of Proposed Mapping Techniques for MiBench Workloads

SD-EESA framework is developed for periodic (IoT sensor) applications on asymmetric multicore. Simulated results are shown in figure 2. Synthetic task graphs and benchmark workloads are tested using the proposed algorithm. SD-EESA scheduled the periodic tasks based on the online profiling data of active cores, which updated every't= 4ms' at runtime. The ready tasks are mapped to the appropriate processors based on the runtime core utilization factor. SD-EESA scheduler.SD-EESA improved the make span reduction up to 73.7% in average than HEFT and 60.1% on average than R-HEFT method and improved the make span for MiBench workloads up to 30.6% in average than HEFT and 14.6% on average than R-HEFT method. For MiBench workloads, energy consumption (reduction) is improved up to 34.9% than HEFT method and 19.2% than R-HEFT. Energy consumption (reduction) is improved in an average of 61.8% than HEFT and 36.3% than R-HEFT method. The time complexity of the SD-EESA framework is O (V^2.M), where V is the total number of tasks, and M is the number of active cores.

VIII. Conclusion

This article addresses the energy efficient scheduling techniques to achieve an optimal solution in bothsystem execution time and energy consumption of asymmetric multicore architecture. Semi-dynamic energy efficient scheduling algorithms are designed and developed for periodic applications. The proposed algorithm includes three different stages, such as task modeling, task prioritization, and core mapping. The proposed scheduling framework is achieved an optimal solution in both time and energy with less time complexity O (V2. M). this scheduling framework is evaluated with traditional methods such as HEFT and robust HEFT. Simulated results show an improvement of 30.6% and14.6% in overall execution time and 34.9% and19.2% in energy reduction for MiBench workloads.

References

1. Liu, "Scheduling Functionally Heterogeneous systems with utilization balancing," IEEE Transaction on Parallel and Distributed Processing Symposium, (1187-1198), 2011.
2. H.Yazdanpanah, Amin Shouraki, "Evaluation Performance of Task Scheduling Algorithms in Heterogeneous Environments," International Journal of Computer Applications (0975 – 8887) Volume 138 – No.8, March 2016.
3. H.Chen, Cheng AMK, "Assigning real-time tasks to heterogeneous processors by applying ant colony optimization," Journal of Parallel and distributed computing (32-42), 2011.
4. Lee WY "Energy efficient scheduling of periodic real-time tasks on lightly loaded multicore processors," IEEE Transaction on Parallel and distributed system, 233:530-537, 2012.
5. Huang L, Yuvan F, Xu Q "On task allocation and scheduling for lifetime extension of platform-based MpSoc designs," IEEE Transaction on Parallel and distributed system, 2212:2088-2099, 2011.
6. Omar Kermia, An efficient approach for the multiprocessor non-preemptive strictly periodic task scheduling problem, Journal of System architecture, vol.79, 2017.

7. Ergin O “Circuit techniques for power-aware micro cores." Master thesis, the State University of New York, USA
8. Guoqi Xie, Gang Zeng, Liangjiao Liu, Renfa Li, Kequin Li, Mixed real-time scheduling of Multiple DAGs based applications on heterogeneous multi-core processors, Journal of system architecture, 2016.
9. Benjamin, Steven, Isabell. Tightening contention delays while scheduling parallel applications on multi-core architectures, International Conference on Embedded Software(EMSOFT), 2017.
10. Songyun wang, Zhuo Zhong qi, A DVFS Based Energy-Efficient Tasks Scheduling in a Data Center. IEEE ACCESS Special Section On Emerging Trends, Issues, And Challenges In Energy-Efficient Cloud Computing Vol 5, 2017.
11. Guoqi Xie, Gang Zeng, Xiongren Xiao, Energy-Efficient Scheduling Algorithms for Real-Time Parallel Applications on Heterogeneous Distributed Embedded Systems. IEEE Transactions on parallel and distributed systems, vol. 28, no. 12 December 2017.
12. H.Topcuoglu, S.Hariri “Performance-effective and low-complexity task scheduling for heterogeneous computing," IEEE Transaction on Parallel and distributed systems, 13(3), (2002), 260-274.
13. Hamid Arabnejad and Jorge "List Scheduling Algorithm for Heterogeneous Systems by an Optimistic Cost Table" IEEE Transaction on Parallel and distributed systems, 25(3), March 2014.
14. Jyothi Thaman, Manpreet Singh, "Green Cloud environment by using robust planning algorithm," Egyptian informatics of journal, 2017.
15. Neetesh Kumar “A GA based energy aware scheduler for DVFS enabled multicore systems," Journal of Computing, 2017.