Energy-aware Scheduling based Tasks dynamic Priority on Homogeneous Multiprocessor Platforms

Shahira M. Habashy
Electronic and Communication Department
Engineering Collage, Helwan University, Cairo, Egypt

ABSTRACT

Today the computation capability of modern computational systems increased. Multi-core processors achieve improved performance with lower power consumption. Dynamic Voltage and Frequency Scaling (DVFS) technique, permits processors to dynamically change their supply voltages and execution frequencies so it can work on many power/energy levels. This scheme is considered as an efficient technique to fulfill the goal of saving energy. This paper, considered scheduling task set on homogeneous multiprocessor platforms using Dynamic Voltage and Frequency Scaling. Achieving minimal overall system energy consumption was our goal. We propose an integrated approach that assigns a dynamic priority to each task in ready queue related to every ready processor based on task deadline and processor load. We are experimentally inspected the effect of our dynamic priority algorithm using feasibility, energy and feasibility/energy performance measurement. Our evaluation results show considerable energy gains with acceptable performance when compared with other well-known heuristics.

Keywords

Multi-core processor, Dynamic priority Scheduling, Dynamic Voltage and Frequency Scaling, Energy-Aware Scheduling

1. INTRODUCTION

Chip Multi-Processors (CMPs) that submit more than one processing core on a single chip have rapidly become prevalent. Pioneer chip makers now have CMPs with 2, 4 or 8 cores [1,2]. Further, comprehensive research is underway to build chips with prospect hundreds of cores [3,4]. Energy consumption was considered as one of the main encourage factors leading to CMP architectures. It was the unendurable ever-increasing frequency and power density orientations of traditional single core architectures. As a result, CMPs become close fitting with Dynamic Voltage and Frequency Scaling (DVFS) as it provide multiple operating point steps [5-7]. Processor energy consumption effected by two components: dynamic and static power. The first refers to the power that is wasted due to switching activity, while the second is related to leakage current. A simple way for reducing dynamic power is to employ Dynamic Voltage Frequency Scaling (DVFS) [8]. DVFS alternates the processor supply voltage and the clock frequency simultaneously, which result in the ability for energy consumption reducing. Shutoff the processor cores while they are idle was considered as a method for reducing the static power consumption [9-12]. Applying the mechanism of dynamic shutdown of processor cores required a supported hardware. The energy-efficient scheduling problem of hard real-time systems with DVFS and/or dynamic shutdown-capable processors used to decrease energy consumption however, guarantee that all the real-time tasks meet their deadlines. Task scheduling process on multiprocessor scheme can be categorized into two classes, namely, partition-based scheduling and global scheduling. In partition-based scheduling, each task is allocated immobile to one processor. Partition based scheduling allows scheduling to be established by mature uniprocessor analysis techniques. In global scheduling, there is monosyllabic job queue from which jobs are sent to any ready processor according to a global priority scheme [13]. Tasks in real time system must be processed and produced functionally correct results in a defined time manner. This requires that the tasks, delivered to the system, have known timing parameters. A lot of these real-time tasks are periodic and the processing time of each task instance must be completed before the end of the task's period, it's called the task deadline. Traditionally, a periodic real-time task τ_i has a model which is characterized by its period T_i and its worst-case computation time C_i. Ensuring that each task will complete before its deadline must be done. An admission control and a scheduling policy should be used to ensure that criteria for the real-time system. The admission control can be considered as an algorithm that uses scheduling policy and makes sure that tasks will meet their deadlines. A task set that is meet its deadline is called feasibly scheduled. In the other hand the scheduling policy main job, is determining which task to be processed next. Task utilization is defined as C_i / T_i and the task set utilization is the sum of all the tasks' utilization of the task set. The admission control compares the utilization of the tasks in the task set and determines a set of m tasks that will not miss their deadline [14-18]. The problem of scheduling periodic preemptive task set on an identical multiprocessor platform considering DVS capability aiming to minimize energy consumption, is presented in this research. We assign dynamic priorities for each task in ready queue related to every ready processor. Those priorities depend on each processor load and task dead line. Each task priority can be changed during task scheduling period related to that task start executing on different processor and that processor load. Global multiprocessor scheduling using dynamic reclaim can be considered as an open problem. In contrast, avoiding task migrations overhead is incentive, which let as focus on the partitioned approach. Lectures show that task allocation has an effective and significant impact on the system overall energy consumption. Also a speed control scheme can achieve good improvement in reducing the energy consumption. At that point, a simple scheme can be used to scale up the effective task utilization by a factor related to the processor speed. This paper considers on line scheduling problem (where tasks arrive to the system dynamically). We present an analysis to that difficult (online scheduling) problem. In our work, we used feasibility and energy consumption to measure the system performance. Assigning tasks to schedule on a multiprocessor system, is our aim, with low energy consumption while preserving feasibility. Lectures introduced an additional hybrid scheme (called feasibility/energy scheme) which inherent tradeoff between feasibility and energy performances. It's measured as the percentage of feasible tasks to the total system energy consumption. This scheme upholds techniques with low energy consumption and high feasibility performance.

The result shows that our proposed algorithm achieved better feasibility/energy measurement compared to other heuristic algorithms. The rest of this paper is organized as follows. We summarize previous research related to this work in Section 2. Section 3 describes the processor, task, and energy models that we are used in this paper. In Section 4, Multiprocessor admission control and scheduling policy are introduced. Section 5, discusses our solution approach. Section 6, presents the experimental evaluation of our proposal-dynamic priority task scheduling algorithm. We conclude the paper in Section 7.

2. RELATED WORK

Lecture survey [19] introduced three methods for optimizing energy consumption in multi-cores system. First method depends on designing an algorithm usage DVFS and dynamic energy management techniques. Second method implements thermal aware scheduling. Third method bases mainly on asymmetric aware scheduling. Yao, and et al. [20] introduced a static scheduling technique. They assume knowing the average switch activity previously, Then the technique used DVFS/DPM. The result shown 70% minimizing in energy consumption. Aydin, and et al. [21] implemented DVFS technique to manage dynamic power. At the operating mode, they statically predestined the power consumption and performance of the processor. Coskun, and et al. [22] Mapped tasks and cores with the aid of integer linear programming (ILP). They depend on setting voltage and frequency. Stavrou and Trancoso [23] introduced a model based on assigning a new dispatch threads to each core. This model did not restrict migration and the result showed acceptable performance. Coskun, and et al. [24] Suggested a unique technique based on asymmetric aware scheduling. The algorithm distinguished between slow and fast cores then, assigned threads to cores in order to maximize total system performance. Kessaci, and et al. [25] Proposed two models of multi-objective parallel GA. Those models are based on multi-start GA and island GA. Energy aware scheduling approaches are hybridized to improve the system performance. Xin Huang, and et al. [26], introduced a novel scheduling algorithm based on Dynamic voltage and frequency scaling (DVS) technique. This algorithm can save energy by 3% to 12%. Xuan T. Tran [27] proposed an approach aimed to improve energy-efficient scheduling policy. The proposed technique yields considerable energy savings in the system while the performance is not traded off. Tran Thi Xuan, and Tien Van Do [28], investigated a job scheduling problem implemented to multicore system. They applied a Dynamic Power Management technique. Their numerical results show that scheduling policy must investigate machine parameters for achieving best efficiency. Navonil Chatterjee, and et al. [29], proposed energy aware dynamic task scheduling for multi-core platform using task deadline. The simulation results show reduction in communication energy by 28% compared to communication-aware energy that based on nearest neighbor algorithm. The proposed algorithm performed an intelligent resource allocation and improved the rate of deadline for real-time dynamic applications. Ying Li, and et al. [30], proposed an Accelerated Search (AS) algorithm which depends on Dynamic Programming (DP). The proposed algorithm considered data migration energy and guaranteed probability technique. They used multi-core architectures which supported Dynamic Voltage and Frequency Scaling scheme. The experimental results demonstrated 30.7% maximum improvement in the system performance. Neetesh Kumar, and Deo Prakash Vidyarthi [31], introduced a new technique based on managing CPU cycle and latency cycle. Energy consumption at the core level was scaled by using Dynamic voltage frequency scaling (DVFS) technique. Experimental results, show exhibited scalable and energy efficient over other contemporary models.

3. SYSTEM MODEL AND ASSUMPTIONS

3.1 Processor and Power Model

The processor model used in this paper consist of a homogeneous multi-core processor which has m processors $\{p_1, ..., p_m\}$. Each processor can be operated at frequency ranged from f_{min} to f_{max}. All the processors are supplied from the same clock signal. This last assumption inverts many current processors such as NVIDIA's Tegra 2 processor. We cannot consider run time frequency scaling because it is required large coordinated scaling across all cores during a single domain and which led to endure high overhead. Using varying processor voltage model resulted dynamic power consumption function. This function was proportional to f^{α}, where α is a constant [32]. The processor leakage power is a non-negative constant and denoted by β_2. Then, the accurate power consumption function is $(\beta_1 f^{\alpha}+\beta_2)$ eq. 1. From [33], the power consumption function constant values, can be estimated to be $\alpha = 3.94565$, $\beta 1 = 3.89462 \times 10^{-26}$ and $\beta 2 = 0.8453 \times 10^{-9}$.

3.2 Task Model

A task set consists of *n* independent real-time tasks $T = \{T_1, ..., T_n\}$. Considering that each task $T_i = (C_i, P_i, D_i)$ has three parameters: the worst-case execution time C_i, a period P_i, and deadline D_i. This study assumed that the task relative deadline D_i is equal to the task period P_i. As using a processor that can be operated on varying speed, the worst-case task execution time C_i is calculated relative to the worst-case number of cycles.

The total task set utilization is given by $U_{tot}(S) = C_i /(P_i S)$, where S is the processor running. However, the maximum task utilization can be obtained when the processor run at maximum relative speed (i.e. $S = 1.0$). The scheduling model used is the non-preemptive technique. The condition for feasibility test, was calculated with the aid of task set total utilization U_i and check if this summation does not exceed the computing capacity ($U_{tot} \leq m$). The largest utilization in the task set, called the *utilization factor* $\alpha = max(u_i) \quad \forall \ T_i \in T$, was used as an indication to the task load.

3.3 Energy Minimization and Partitioning

Energy -partition scheduling is considered as NP hard problem. In energy -partition problem the objective goal is minimizing the total energy consumption on all the Processors. Those processors subject to the constraint that preserving task feasibility. Our work consider a task set *T* of real-time tasks and a set *M* of identical processors, the problem is finding a suitable processor to run each task and at the same time trying to minimize that processor speed without effecting task feasibly.

4. MULTIPROCESSOR ADMISSION CONTROL AND SCHEDULING

The problem of task allocation is considered as an NP-Hard problem [34]. The aim is finding an allocation algorithm and a scheduling test to ensure that a given task set is scheduled (feasibly) on the available processors. Bin-Packing techniques were considered as one of the famous scheduling techniques. In this problem, the processor is considered as the recipient bin. The utilization bound of each local processor is known as the bin capacity. It is required to put *n* tasks with utilization u_k into the minimum number of bins (processors), so that the total utilization of the tasks on each bin does not override its maximum capacity [35]. The best-known Bin-Packing techniques found in the literature are [34]:

- First-Fit (FF): allocates a new task to a non-empty bin which has the lowest index, so that the utilization of all the already allocated tasks to that bin plus new task utilization do not exceed the capacity of the bin. If the new task addition overflows the bin capacity, then FF assigns the new task to an empty bin.

- Worst-Fit (WF): allocates a new task to a non-empty bin with the greatest capacity. If the new task addition exceeds the bin capacity, then WF allocates this new task to an empty bin.

- Best-Fit (BF): allocates a new task to a non-empty bin with the smallest capacity. If the new task addition exceeds the bin capacity, then BF allocates this new task to an empty bin. BF chooses the bin with the smallest index if there is more than one bin having the same capacity.

- Earliest Deadline First (EDF): arranges the task according to its dead line and the task with the earliest dead line will be allocated first to the smallest index bin with constraining, of not exceed that bin capacity.

Bin capacity in all the above techniques calculated a processor utilization U_j which is the sum of all the tasks' utilization allocated to that processor and cannot exceed 1,

$$U_j = \sum_{i=1} u_{ij} \ \le 1 \qquad eq.2$$

This condition is considered as a sufficient test for task feasibility. On the other hand lectures provided other feasibility test categories. Those tests introduce sufficient conditions for feasibility.

• Exact Liu-Layland test (ELL) is the most frequently used test for feasibility in RMS. In that test, a task set included *n* tasks can be scheduled on certain processor if the total task set utilization U_{tot} *doesn't exceed* $U_{bound.}$

.

$$U_{bound}(n) = n(2^{1/n} - 1) \qquad \text{eq. 3}$$

• Hyperbolic test (HYP), introduced by Buttazzo, and et al. [9], it tasks into consider the individual task utilizations to provide a tighter bound than ELL.

$$\prod_{i=1}^{n}(1+u_i) < 2 \qquad \text{eq.4}$$

• Burchard test (Burc): based on the concept that having harmonic tasks can help in improving the utilization bound. That bound was expressed as a function of two parameters. The first one is the number of tasks and other parameter achieves harmonic between the close tasks [36-38].

5. THE PROPOSED DYNAMIC PRIORITY ENERGY AWARE TECHNIQUE

Traditionally, real-time scheduling on a single-core platform has been classified into two categories: static and dynamic scheduling. The idea of dynamic scheduling is to continuously select the task with the highest priority out of the incoming active tasks to be executed, while the priorities of those active tasks are changing continuously (e.g., Earliest Deadline First scheduling algorithm (EDF)) [39]. By contrast, for static scheduling, the incoming active tasks have priorities that cannot be changed (e.g., Rate-Monotonic scheduling algorithms (RM) [39-42]. Static scheduling techniques were not considered optimal for real-time scheduling on multi-core platforms. As a consequence, much published researches have recently focused on real-time scheduling on multiprocessor platforms using dynamic scheduling [43-44]. Our proposed algorithm focuses on solving the scheduling problem of real-time tasks on multi- processor platform. We propose a dynamic-priority scheduling algorithm and use DVFS for saving energy. In practice, real-time partitioned scheduling for multi-core processor is NP-hard problem. Many effective partitioning heuristics are simple and have reasonable average-case performance [45-48]. ENERGY-PARTITION is an optimization problem where the objective goal is minimizing the total energy consumption on all the active processors. According to the relationship between the CPU speed and the power consumption, minimizing energy consumption across DVS- multiprocessors suggests implementing load balancing technique [49]. The next example illustrates this criteria.

Motivational Example: Consider two identical processors and a set of six real time tasks $T = \{T1,\dots, T6\}$. The individual task utilizations U= {0.24, 0.2, 0.12, 0.07, 0.05, 0.04}, under maximum speed. This leads to total utilization U_{tot} = 0.72 which is not exceeded Liu-Layland bound, so tasks' feasibility can be achieved under any scheduling partitioning. Figures 1-3, show three different scheduling partitions and their energy consumption patterns.

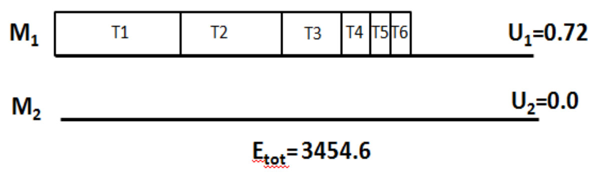

Figure 1. A feasible partition (Partition 1)

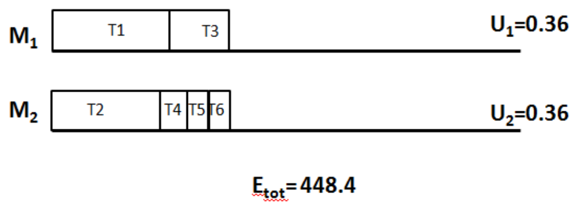

Figure 2. A feasible partition (Partition 2)

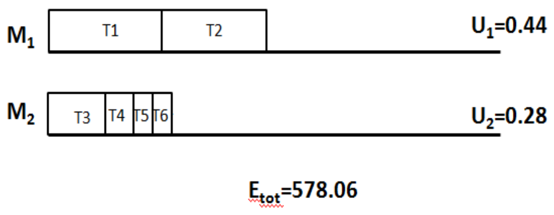

Figure 3. A feasible partition (Partition 3)

Figure 1, represents the schedule that may be introduced by unbalancing load heuristics like First-Fit or Best-Fit. The schedule partitioning shows in Figure 2, is the exactly balanced one. In figure 3, the largest tasks utilization are exclusively assigned to the first processor, while all the remaining tasks are dedicated to the second processor. The workload in all the three cases is considered feasible but the energy pattern proceeds a slightly different condition. The energy consumption E_{tot} of each partition was calculated using eq. 1. Figures 1 to 3 show that schedule partition 1, has the maximum energy consumption while Partition 2, (the perfectly balanced partition) gives the minimum energy consumption. This example explains that variable feasible partitions can have significantly different energy pattern. Among the possible scheduling partition techniques, load balancing scheduling result in reducing the energy consumption. According to the computational complexity, ENERGY- PARTITION is NP-Hard problem. ENERGY partitioned scheduling using multiprocessor is considered as a more complex problem than achieving feasibility. As shown from the above example, ENERGY partitioned stays NP Hard *even* if the task set can be scheduled on one processor. At that point, finding the optimum scheduling with the minimum energy consumption is considered as a hard problem. Our proposed algorithm, dynamic priority scheduling calculates for all the available tasks in ready queue, a set of priority related to each processor (task priority vector). When the resources are available, tasks with the highest priorities will be scheduled firstly. During task processing, the algorithm updated task priority vector for each task (still staying in ready queue) related to all available processor. If certain task can't meet its dead line on one processor, that processor speed could be increased to meet that task deadline. The proposed priority equation (eq. 5) consists mainly from two terms. The first team is one minus the ratio between processor end processing time and task dead line. The second term is the ratio between task energy if it is processed on that processor to the processor consumed power. Those two terms are worked together for consuming load balance on all the available processor.

$$priority_{ij} = 1 - \frac{Delay\_processor_j}{deadline\_task_i} + \frac{E\_task_i}{E\_processor_j} \qquad eq.5$$

The optimal processor speed is calculated initially using the ready tasks in ready queue. During the execution time, each processor speed is updated. Processor speed updating depends on the chosen task processor pair (according to eq. 5) and tasks already in task queue. If the scheduling algorithm cannot find any core to execute the ready task without missing its deadline, that task was considered as infeasible. The proposed algorithm called three functions. The first one- Get medium task utilization ( ) - which returns medium utilization for the available tasks. That returns value was used in second function- Get optimum processor speed ( ) - to determine the current optimum speed for each processor. The third one -Get task priority ( ) - calculates task priority vector according to eq. 5.

Dynamic Priority Task scheduling algorithm

Input: task _queue set: **τ**= {τ_1, τ_2, . . . , τ_N} ,those tasks deadline, set of processors p = {P_1, P_2, . . . , P_M}, and set of the available processor speed.
Output: return the tasks assign to each processors and each processor speed during executing their assignment tasks.

```
1: if task queue.empty( ) == FALSE then
2:     Get medium task utilization ( ).
                 //relative to tasks in task_queue.
3:     Optimal speed =Get optimum processor
         speed ( ).
                 //relative to tasks in task_queue.
4:     Ready task group = task queue.min deadline( );
5:     Set task queue by arranging the tasks according to their
         deadline
6:     Find new task period according to optimal speed.
7:     Get task priority ( )
8: End
9: Find the max task processor pair having max priority ($P_{max}$)
10: if $P_{max}$ > 0
11:     Allocate chosen task to the chosen processor
12: else
13:     for each task in ready task group
14:         for each processor speed in s $_{selected}$ group greater than
                 the present speed
15:             Find new task period
16:             Get task priority ( )
17:         End
18:     End
19:     Find the max task processor pair having max priority
           ($P_{max}$)
20:     if $P_{max}$ > 0
21:         Allocate chosen task to the chosen processor
22:     else
23:         Add that task to unscheduled task group.
24:     End
25: End
26: Go to step 4.
```

Get medium task utilization ( )

```
1: Begin
2: Calculate utilization sum for all tasks.
3: Calculate medium task utilization by dividing that sum on
     the number of the available Processors.
4: Return medium task utilization
5: End
```

Get optimum processor speed ( )

```
1: Begin
2: Set the different processor speed as a relative multiple from
     the minimum speed  S={1, …, $s_{max}$}.
3: Find the set of processor speed $s_{selected}$ which is greater than
     or equal to the medium task utilization.
```

```
4: Set the initial speed to all the processor to the first speed in
      $s_{selected}$ group
5: End
```

Get task priority ( )

```
1:  Begin
2:        for each task in ready task group
3:             for each processor
4:                  Calculate possessor end time after Adding that
                    task to that processor
5:                if   possessor end time <= task dead line
6:                        Calculate task processor priority according
                                    to eq. 5
7:                else
8:                     Task processor priority=0;
9:                End
10:            End
11:      End
12:  End
```

6. EXPERMINTAL RESULTS

We used three parameters (feasibility, energy, and feasibility/ energy) to measure the performance of our proposed dynamic priority algorithm. Task utilization factor α and the total task set utilization U_{tot} are used for comparing those parameters. The number of processor M used during our experimental doesn't change and it is chosen to be 8. Total task set utilization U_{tot} varied between M/ 10 (light task load condition) and M (heavy task load condition). Task utilization factor α is adjusted to take two values (0.5 and 1). α equal 1 means there is no constraint on individual task upper bound utilization. During the experiments 1000 task sets was generated with varying task period P_i,, deadline d_i, and utilizations u_i. Each task can take any period which range from (1-10ms) short task period, (10- 100ms) medium task period, or (100-1000ms) long task period. Task utilizations u_i for each task is adjusted to be in the range [0.001, α]. Lectures show that improving the feasibility performance can be done by ordering tasks according to its task utilization value. In practical, this lead to increasing the total system performance [50]. Example in section 5, investigates the effect of load balancing on both feasibility and energy. We start by comparing our proposal dynamic priority algorithm with the result from bin-picking techniques (FF, BF, and WF). WF tends to produce balanced partition, while FF, and BF tend to produce unbalanced partitions. Those techniques produced different feasibility and energy behaviors. Studying bin-packing partitioning techniques show that FF and BF collecting as many tasks as possible on a small number of processors. That covetous action improves the feasibility [23, 51]. From the energy point of view, load balancing result in reducing the energy consumption. Dividing the load among all the available processors (load balancing) resulting in the possibility of using DVS to lower all the processor speed. As a result, the system total energy will be reduced. An unbalanced load partition resulted heavily load on some of the processors and lightly load on the remaining processors. This result in increasing the heavy loaded processors speed to achieve the task set feasibility. This led to increase the system total energy. In on-line partitioning, the scheduler algorithm cannot pre-order tasks before the task allocation phase. The scheduler algorithm assigns task when it arrived to the system and it takes into consideration the previous tasks in ready queue. Any one of the traditional partitioning heuristics FF, BF, and WF, cannot be considered as a clear winner with respect to the overall performance. . FF, and BF give good feasibility performance, but on the other hand, they appear bad energy and feasibility/energy performance, especially at low utilization values. WF, gives good energy performance, but its feasibility/energy performance has very low. Figure 4, shows that our proposed dynamic priority algorithm still has the best feasibility performance while, WF gives low feasibility performance mostly at high utilization.

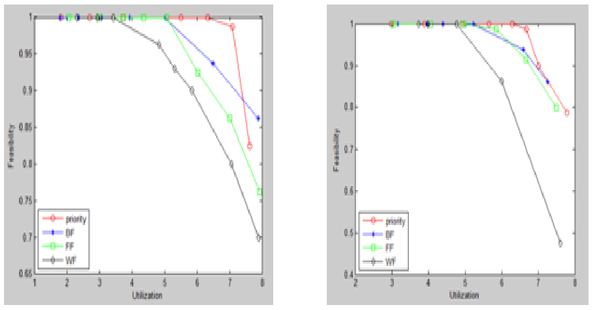

Fig 4, Feasibility of online partitioning heuristic algorithms compared to dynamic priority algorithm for α = 0.5 (left) and α=1.0 (right)

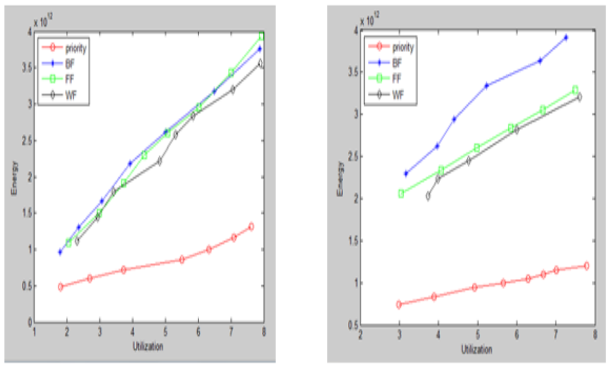

Fig 5, Energy performance of online partitioning heuristic algorithms compared to dynamic priority algorithm for α = 0.5 (left) and α=1.0 (right)

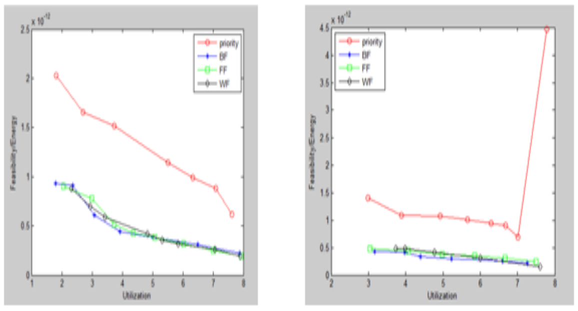

Fig 6, Feasibility/energy performance of online partitioning heuristic algorithms compared to dynamic priority algorithm for α = 0.5 (left) and α=1.0 (right)

Figure 5, shows high energy consumption of BF technique because of its unbalanced partitions. WF technique has the lowest energy consumption, while our proposed algorithm is the great winner. Figure 6, shows the feasibility/energy performance. There is no obvious winner throughout the utilization spectrum from the bin-packing techniques. However, our dynamic priority algorithm has achieved consistently good overall performance.

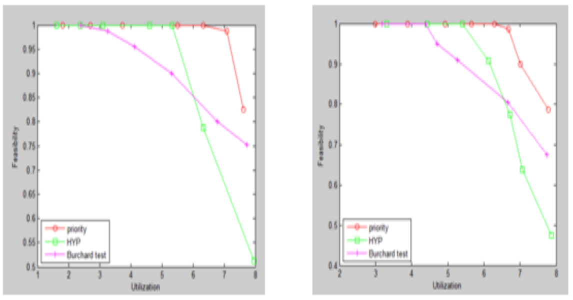

Fig 7, Feasibility of online feasibility test techniques compared to dynamic priority algorithm for α = 0.5 (left) and α=1.0 (right)

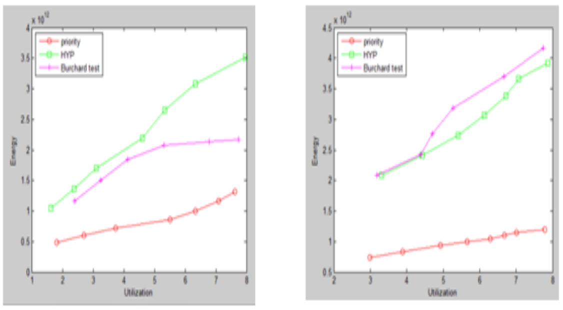

Fig 8, Energy performance of online feasibility test techniques compared to dynamic priority algorithm for α = 0.5 (left) and α=1.0 (right)

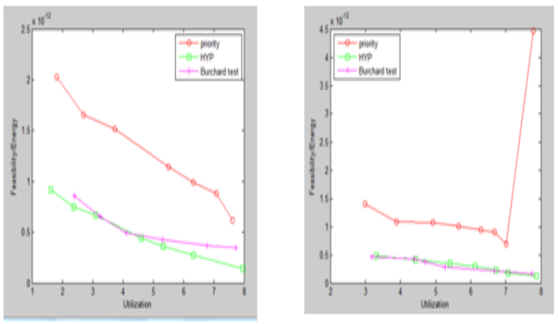

Fig 9, Feasibility/energy performance of online feasibility test techniques compared to dynamic priority algorithm for α = 0.5 (left) and α=1.0 (right)

Fig. 7-9, compare some of the feasibility test techniques with our proposed dynamic priority technique. Figure 7, compares the different feasibility tests for α=0.5, and 1. Dynamic priority exhibits the best feasibility performance, while HYP takes the second best performance scheme during low and medium utilization. In term of energy consumption, (Figure 8), Dynamic priority is still the best performance scheme. The second best one is HYP for large α values. However, for small values of α, burchard test appears to be the second best scheme Figure 9, presents results for feasibility/energy performance of different techniques. Dynamic priority is still significantly better performance since the feasibility performance deteriorates at high load.

7. CONCLUSION

Multi-core technology benefits making it an essential trend in many real-time systems. Job scheduling to those appropriate cores, is an NP-hard problem. The main challenge introduced here is the trade-off between system performance and energy efficiency. Dynamic voltage frequency scaling (DVFS) technique can be used to scale energy consumption at the core level. This study aims to achieve optimal energy usage of multicore systems by effectively uses of DVFS technique and applies dynamic priority scheduling algorithm. The assigned task to processor dynamic priority is based on task deadline and the appropriate processor load. On-line partitioning case problem is considered: where task set characteristic are unknown previously, and task allocation decision was made in order. We compare our proposed algorithm to a number of heuristics scheduling algorithms. Feasibility, energy and feasibility/energy performance measurement are used as a comparison tools. Experimental results, exhibit that our proposed model accomplishes energy efficient with acceptable performance over other contemporary models.

8. REFERENCES

[1] J. Dorsey, S. Searles, M. Ciraula, S. Johnson, N. Bujanos, D. Wu, M. Braganza, S. Meyers, E. Fang, and R. Kumar, “An integrated quad-core opteron processor”, In Solid- State Circuits Conference, ISSCC 2007. Digest of Technical Papers. IEEE International, feb. 2007, pages 102 –103.

[2] http://www.intel.com/products/processor/corei7/ specifications.

[3] http://www.intel.com/design/intarch/xeon/ specifications xeon.htm.

[4] R. Kumar and G. Hinton, “A family of 45 nm ia processors”, In Solid-State Circuits Conference - Digest of Technical Papers, ISSCC 2009. IEEE International, feb. 2009, pages 58 –59.

[5] Li, Dawei, et al., “Energy-aware scheduling on Multiprocessor Platforms”, Springer Briefs in Computer Science, 2013.

[6] Tom Guérout, et al., “Energy-aware simulation with DVFS”, Simulation Modelling Practice and Theory, Elsevier, 39 (2013) 76–91.

[7] L. Mosley, “Power delivery challenges for multicore processors”, In Proceedings of CARTS, 2008.

[8] C.-Y. Yang, J.-J. Chen, T.-W. Kuo, and L. Thiele, “An approximation scheme for energy-efficient scheduling of real-time tasks in heterogeneous multiprocessor systems,” in Design, Automation Test in Europe Conference and Exhibition, April 2009, pp. 694–699.

[9] S. Irani, S. Shukla, and R. Gupta, “Algorithms for power savings”, ACM Trans. Algorithms, 2007, 3(4):41.

[10] MARIO BAMBAGINI, et al., “Energy-Aware Scheduling for Real-Time Systems: A Survey”, ACM Transactions on Embedded Computing Systems, Vol. 15, No. 1, Article 7, Publication date: January 2016.

[11] Rajkumar K, and Swaminathan P, “Optimized energy aware scheduling to minimize makespan in distributed

systems", Biomedical Research, India, 2017; 28 (7): 2877-2883.

[12] A. Rowe, K. Lakshmanan, H. Zhu, and R. Rajkumar, "Rate-harmonized scheduling for saving energy", In RTSS '08: Proceedings of the 2008 Real-Time Systems Sympo- sium, Washington, DC, USA, 2008. IEEE Computer Society, pages 113–122.

[13] 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.

[14] YAN WANG,et al., "Energy-Aware Data Allocation and Task Scheduling on Heterogeneous Multiprocessor Systems With Time Constraints", IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING, Digital Object Identifier 10.1109/TETC.2014.2300632, January 2014.

[15] Weiwei Lin, et al., "A Heuristic Task Scheduling Algorithm for Heterogeneous Virtual Clusters", Hindawi Publishing Corporation Scientific Programming Volume 2016, Article ID 7040276, 10 pages. http://dx.doi.org/10.1155/2016/7040276

[16] Weicheng Huai, et al., "Energy Aware Task Scheduling in Data Centers", Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, July 2013, volume: 4, number: 2, pp. 18-38

[17] S. Saewong and R. Rajkumar, "Practical Voltage-Scaling for Fixed- Priority Real-time Systems", Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'03), May 2003.

[18] Silvana Teodoro, et al., "Energy Efficiency Management in Computational Grids through Energy-aware Scheduling", ACM 978-1-4503-1656-9/13/03, Coimbra, Portugal, march 2013.

[19] Zhuravlev S, Saez JC, Blagodurov S, FedorovaA, PrietoM, " Survey of energy-cognizant scheduling techniques", IEEE Trans Parallel Distributed System , 2013, 24(7):1447–1464.

[20] Yao F, Demers A, Shenker S, "A scheduling model for reduced CPU energy", In: Proceedings of the 36th annual symposium on foundations of computer science (FOCS '95), , 1995, pp 374–382.

[21] Aydin H, Melhem R, Mossé D, Mejia-Alvarez P, "Dynamic and aggressive scheduling techniques for power-aware real-time systems", In: Proceedings of the 22nd IEEE real-time systems symposium (RTSS'01), 2001, pp 95–105.

[22] Coskun AK, Rosing TS, Whisnant KA, Gross KC, "Temperature-aware MPSoC scheduling for reducing hot spots and gradients", In: Proceedings of the Asia and South pacific design automation conference (ASP-DAC '08) , 2008, pp 49–54.

[23] Stavrou K, Trancoso P, "Thermal-aware scheduling for future chip multiprocessors", EURASIP J Embed Syst 2007(1):40–40

[24] Coskun AK, Rosing TS, Gross KC, "Utilizing predictors for efficient thermal management in multiprocessor SoCs", IEEE Trans Comput Aided Des Integr Circuits Syst 28(10) , 2009, pp.1503–1516.

[25] Kessaci Y, Mezmaz M, Melab N, Talbi E-G, Tuyttens D, "Parallel evolutionary algorithms for energy aware scheduling", In: Bouvry P, Gonzalez-Velez H, Kołodziej J (eds) Intelligent decisions systems in large-scale distributed environments, studies in computational intelligence series, Chap 4, vol 362. Springer, Berlin, 2011 pp 75–100.

[26] Xin Huang, KenLi Li, RenFa Li, "A Energy Efficient Scheduling Base on Dynamic Voltage and Frequency Scaling for Multi-core Embedded Real-Time System", International Conference on Algorithms and Architectures for Parallel Processing, 2009, pp 137-145.

[27] Xuan T. Tran, "Resource-Aware Scheduling in Heterogeneous, Multi-core Clusters for Energy Efficiency", International Conference on Advances in Information and Communication Technology, 2016, pp 520-529.

[28] Tran Thi Xuan, Tien Van Do, "Job Scheduling in a Computational Cluster with Multicore Processors", Advanced Computational Methods for Knowledge Engineering, May 2016, Vienna, pp 75-84

[29] Navonil Chatterjee, et al., "Deadline and energy aware dynamic task mapping and scheduling for Network-on-Chip based multi-core platform", Journal of Systems Architecture, Volume 74, March 2017, Pages 61-77.

[30] Ying Li, Jianwei Niu, Mohammed Atiquzzaman, and Xiang Long, "Energy-aware scheduling on heterogeneous multi-core systems with guaranteed probability", Journal of Parallel and Distributed Computing, Volume 103, May 2017, Pages 64-76.

[31] Neetesh Kumar, Deo Prakash Vidyarthi, "A GA based energy aware scheduler for DVFS enabled multicore systems", Computing, Springer, October 2017, Volume 99, Issue 10, pp 955–977.

[32] Jejurikar, and Rajesh Gupta, "Energy-Aware Task Scheduling With Task Synchronization for Embedded Real-Time Systems", IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 25, NO. 6, JUNE 2006

[33] Henri Casanova, et al., "Algorithms and Scheduling Techniques to Manage Resilience and Power Consumption in Distributed Systems", Dagstuhl Reports, Vol. 5, Issue 7, pp. 1–21

[34] O. U. P. Zapata and P. M. Alvarez, "EDF and RM Multiprocessor Scheduling Algorithms: Survey and Performance Evaluation", TR, 2005, pp. 1 - 24.

[35] Weihua Zhang, et al., "A Novel Task Communication and Scheduling Algorithm for NoCbased MPSoC", International Journal of Smart Home Vol. 9, No. 10, (2015), pp. 179-188

[36] Jingcao Hu, and Radu Marculescu, "Energy-Aware Communication and Task Scheduling for Network-on-Chip Architectures under Real-Time Constraints", Proceedings of the conference on Design, automation and test in Europe, 2004, Volume 1, page 10234.

[37] Mohammad Shojafar, et al., "An Energy-aware Scheduling Algorithm in DVFS-enabled Networked Data Centers", In Proceedings of the 6th International

Conference on Cloud Computing and Services Science (CLOSER 2016) - Volume 2, pages 387-397.

[38] Paula Zab, et al., “Energy-aware scheduling mandatory/optional tasks in multicore real-time systems”, international transactions on operational researches, Volume 24, Issue 1-2, January/March 2017, Pages 173–198.

[39] Robert I. Davis,”A Review of fixed priority and EDF scheduling for hard real-time uniprocessor systems”, J. ACM Trans. Embedd. Comput. Syst., 11, 1 (Feb. 2014), 8–19.

[40] Tom Gu´erout, Mahdi Ben Alay, “Autonomic energy-aware tasks scheduling”, Open Archive TOULOUSE Archive Ouverte (OATAO), DOI :10.1109/WETICE.2013.29, June 2013, URL : http://dx.doi.org/10.1109/WETICE.2013.29

[41] Mario Bambagini, et al., “Energy-Aware Scheduling for Tasks with Mixed Energy Requirements”, Proceedings of the 4th International Real-Time Scheduling Open Problems Seminar (RTSOPS 2013)

[42] Lizhe Wang, et al., “Energy-aware parallel task scheduling in a cluster”, Future Generation Computer Systems, Elsevier, 29 (2013) 1661–1670

[43] Marko Bertogna, “Real-Time Scheduling Analysis for Multiprocessor Platforms”, Ph.D. Dissertation, Scuola Seprropre Sant Anna, Pisa, 2007.

[44] Akash Kumar, et al., “ Energy-aware task mapping and scheduling for reliable embedded computing systems”, ACM Transactions on Embedded Computing Systems, No. 72, Volume 13 Issue 2s, January 2014

[45] Namita Sharma, et al., “Energy Aware Task Scheduling for Soft Real Time Systems using an Analytical Approach for Energy Estimation”, IJASCSE, VOL 1, ISSUE 4, 2013.

[46] Vasanthamani KANNAIAN, Visalakshi PALANISAMY, “Energy optimized scheduling for non-preemptive real-time systems”, Turkish Journal of Electrical Engineering & Computer Sciences, (2017) 25: 3085 – 3096

[47] Jing Mei, “Energy-aware task scheduling in heterogeneous computing environments”, Cluster Comput, pringer Science+Business Media New York, 17:537–550S, 2013.

[48] Y. C. Lee and A. Y. Zomaya, “Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling,” in the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, May 2009, pp. 92 –99.

[49] Rajesh Kumar Pal, Ierum Shanaya, Kolin Paul, Sanjiva Prasad, “Dynamic core allocation for energy efficient video decoding in homogeneous and heterogeneous multicore architectures”, Elsevier, Future Generation Computer Systems 56 (2016) 247–261

[50] Amjad Mahmood, et al., “Energy-Aware Real-Time Task Scheduling in Multiprocessor Systems Using a Hybrid Genetic Algorithm”, Electronics 2017, 6, 40; doi: 10.3390/electronics6020040.

[51] F. A. Armenta-Cano, et al., “Min_c: Heterogeneous Concentration Policy for Energy-Aware Scheduling of Jobs with Resource Contention”, ISSN 0361-7688, Programming and Computer Software, 2017, Vol. 43, No. 3, pp. 204–215.