Energy Efficient Scheduling Algorithm for Applying Dynamic Voltage and Frequency Scaling to Mixed Task Set

Sarla Mehariya¹, Dalpat Songara³
Department of Computer Engineering,
Government Women Engineering College, Ajmer,
India

Ved Mitra², Mahesh C. Govil⁴
Department of Computer Science & Engineering,
Malaviya National Institute of Technology, Jaipur,
India

ABSTRACT
Power consumption is one of the most important factors that affect the designing of battery operated real-time system or an embedded system. Various strategies have been made to improve the power dissipation. Dynamic voltage and frequency scaling (DVFS), is one of the most popular technique for reducing power dissipation and a well researched area. This paper presents DVFSMTS, Dynamic Voltage and Frequency Scaling for Mixed Task Set, which gives the working of an Earliest Deadline First (EDF), based priority exchange server. Experimental results show that DVFSMTS reduces power dissipation without compromising on the deadlines of the periodic task. The results of DVFSMTS are compared with a non-DVFS EDF based priority exchange server and an approximate 50% reduction in energy is obtained.

Keywords-- Dynamic Voltage and Frequency Scaling, Real Time Scheduling, Earliest Deadline First, Priority exchange server, Mixed task set.

1. INTRODUCTION
Power consumption has always been an important design constraint in the development of real-time and embedded systems as technology is moving rapidly towards mobile and portable battery operated devices e.g. smart phone, laptop etc. Power consumption problem can be addressed at different abstraction levels – from architectural to operating systems level. At operating systems level, many real-time scheduling algorithms have been proposed in literature [1], [2] to reduce the power consumption. For CMOS based processors, the power consumption is given by the relation-

\[ P = C_{eff} \times V dd^2 \times f \]  

(1)

where, \( P \) is the power consumption, \( C_{eff} \) is the effective capacitance of the transistor gates, \( V dd \) is the supplied voltage and \( f \) is the operating frequency. It is clear from the above equation that a slight decrement in the voltage and frequency may lead to a considerable amount of reduction in power consumption. This method of reducing the power consumption by varying the voltage and frequency is termed as Dynamic Voltage and Frequency Scaling (DVFS).

In section 2, various DVFS scheduling techniques are discussed. In section 3, the proposed algorithm with the help of an example is explained. In section 4, comparative analysis of DVFSMTS, an EDF version of Priority exchange server with non-DVFS EDF based priority exchange server is done. Section 5 concludes our work.

2. RELATED WORK
The application of DVFS algorithm to periodic task set is a well known research area [1], [2], [3], [4]. However, some work in literature focuses on DVFS application to mixed task set comprising of periodic and aperiodic tasks, [5], [6]. The DVFS algorithm focuses on the usage and distribution of available slack time. The total time required by a task to run completely i.e. the actual execution time (aat) is always less than its worst case execution time (wcet). The difference that exists is the slack and in turn, is utilized for reducing the voltage and frequency dynamically.

There are two different approaches of DVFS techniques [7] based on the usage of slack time, Intra-DVFS and Inter-DVFS. In Intra-DVFS, the slack time is calculated between jobs within a task boundary and then utilized for DVFS. However, in Inter-DVFS, the slack time is utilized between the current task and its successor tasks in order to reduce the voltage and frequency. Our DVFSMTS algorithm uses the later approach.

An Inter-DVFS algorithm has two parts – slack estimation and slack distribution. The slack estimation method for mixed task set varies from that of periodic task set. Periodic tasks are known prior about their occurrence and their slack can be determined statically. Unlike periodic tasks, the arrival time of aperiodic tasks are not fixed and hence, they must be executed with full frequency in order to achieve better response times. Slack distribution method is simple and greedy in nature, as all the available slack time is provided for the next ready task.

Various algorithms have been proposed to calculate the slack time dynamically and then assigning the appropriate voltage and frequency, which includes the cycle-conserving EDF (ccEDF) [8], look-ahead RTDV5 etc. In ccEDF, the utilization of the task set is first calculated with its \( wcet \) and when a task has been executed completely utilization is again calculated with the \( aat \) in order to scale the voltage and frequency appropriately. This technique is known as Utilization Updating. Scheduling of Aperiodic task includes the use of bandwidth preserving schemes such as Deferrable Server (DS), Priority exchange server (PES), Constant Bandwidth Server (CBS) etc. [9].
DVFS has also been applied to Real Time Systems with fault tolerance [12], where fault tolerance is achieved using the Checkpointing mechanism. In Checkpointing mechanism, the current system state is preserved after certain intervals called Checkpoints. In case of a fault, the system is rolled back to the most recent checkpointed state and the operation resumes from there [11], [12], [13].

After the successful application of DVFS technique to the uni-core processors, it has also been carried out on some of the multi-core processors with or without splitted task [14], [15], [16], [17]. When using multi-core processors DVFS can be applied as –

i) POST-DVFS – Here, a schedule is first made with the splitted task to evaluate the schedulability and energy consumption. DVFS is then applied to the schedule.

ii) PRE-DVFS – Here, the frequency of each task is first calculated before scheduling in order to achieve better performance on both schedulability and energy consumption. The DVFS is then applied to the schedule.

3. PROPOSED ALGORITHM AND EXAMPLE

The proposed algorithm DVFSMTS is a scheduling algorithm for mixed task set comprising of periodic and aperiodic tasks. It employs EDF decisions at each reference point with Priority exchange server for aperiodic tasks. Aperiodic tasks are executed with the maximum frequency in order to achieve lower response time. For periodic task frequency is calculated using the updating algorithm according to the ccEDF algorithm [8].

A. Task model

Our system comprises of n periodic tasks, as per task set T = {T₁, T₂,......Tₙ}. Each task Tᵢ is described by the parameters {Φᵢ, Cᵢ, Pᵢ, Dᵢ} where, Φᵢ, Cᵢ, Pᵢ, Dᵢ, represents the phase, wcet, period and deadline of the ith periodic task Tᵢ. Eᵢ is the set of Tᵢ. The set for any task is a random variable of the wcet. There are m Aperiodic tasks, A = [A₁, A₂,…, Aₘ]. Each aperiodic task is described by Ai = [Aᵢ, Ci], where Ai is the arrival time and Ci is the execution time of the ith aperiodic task Ai.

B. Server Task

A Priority exchange server Tₚₑₛ is used which is described by Tₚₑₛ = [Pₑₛ, Cₑₛ]; where, Pₑₛ is the server period and Cₑₛ is the server capacity. When the tasks are released for execution and if there is no aperiodic load, the Cₑₛ is saved at the priority of periodic task executing. Cₑₛ also gets replenished with RA i.e. the replenishing amount when the time equivalent to the Pₑₛ is finished and it is of the highest priority. So, there may be more than one server task i.e. the server task which is replenished and is of the highest priority and the server task which is saved at the priority of the periodic task. The later server task is not replenished when server period is over.

C. Notations used

i) PQ and AQ are the ready queues. Periodic tasks are arranged in PQ according to increasing deadline while AQ is having Aperiodic tasks prioritized according to their arrival time.

ii) Uₑₛ Uₑₛ and Uₑₛ represent the utilization of the periodic task Tₑₛ utilization of Priority exchange server and total utilization respectively.

iii) H is the Hyperperiod defined as the LCM of the deadlines of all the Periodic tasks.

iv) n_r_p is the next reference point.

v) ETL is the execution time left for a task.

vi) Jₚₑₛ is the job at the head of the periodic queue PQ and Jₚₑₛ is the job at the head of the aperiodic queue AQ.

vii) Vₑₛ and fₑₛ represent maximum voltage and frequency, respectively. Vₑₛ and fₑₛ represents the estimated voltage and frequency.

viii) FT is the finish time for any task.

ix) PR(Tᵢ) is the priority of the task Tᵢ.

x) Aₑₛ and Aₚₑₛ represent arrival time of the next periodic and Aperiodic task, respectively.

xi) RA is the amount by which server task is to be replenished.

xii) ST a new server task. Cₑₛ(ST) and Pₑₛ(ST) the capacity and period of ST respectively.

NOTE: Cₑₛ is the server capacity of Tₑₛ and Cₑₛ(ST) is the server capacity of a new server task ST. Total server capacity available for an aperiodic task → Cₑₛ=Cₑₛ+Cₑₛ(ST).

D. Assumptions

1. All periodic tasks are released at t=0.

2. The period and deadline of all the periodic tasks are equal and known in prior.

3. Frequency and voltage ranges are pre-determined.

4. Decisions at each instant as to execute or preempt a task are based on Earliest Deadline First algorithm.

E. Input

T, Aₑₛ, PQ, AQ, t=0, H.

F. Output

Next job (Periodic or Aperiodic) is fed to the processor with the estimated scaled frequency fₑₛ and voltage Vₑₛ. An ERROR message is generated upon missing of the deadline of any periodic task.

DVFSMTS Scheduler(T, A, PQ, AQ, t=0, H)
BEGIN:
for (t=0 to 2H)
{
  ST=null; //ST is a server task
  if(t==Pₑₛ OR t==0)
    Cₑₛ=RA;
  if(Uₑₛ≤1), then
  {
    Print(Task set schedulable);
    if (AQ != null) and (Cₑₛ > 0), then
    {
      n_r_p ← t+min(Cₑₛ, ETL(Jₚₑₛ));
      run Jₚₑₛ from t to n_r_p at fₑₛ Vₑₛ;
      Cₑₛ ← n_r_p – (min(Cₑₛ, FT(Jₚₑₛ)));
      t ← n_r_p;
    }
  }

48
1

PES. o,o
time
Calc_Freq():
BEGIN
IF(Ji completed) //Ji is the periodic job
Uj=E/Pj
Else
Uj=C/Pj
IF(AQ!=null)
U pes=C pes/P pes
Else
U pes=0
U total=\sum_{i=1}^{n} U i + U pes /n is number of periodic
Calc_Freq():
Use lowest frequency fest (f_{i1},...f_{im},f_{i1}<f_{i2}<...<f_{imax})
Such that U total<f_{min}/f_{max}
Return (f_{total},U_{total})
END

Fig. 1. DVFSMTS Scheduler

Fig. 2. Algorithm for Frequency Scaling

G. Example Task Set

TABLE I. PERIODIC TASK SET

<table>
<thead>
<tr>
<th>Task ID</th>
<th>C_i</th>
<th>P_i</th>
<th>D_i</th>
<th>act of each job</th>
<th>Priority PR(T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_{pes}</td>
<td>2</td>
<td>10</td>
<td>10</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>T_3</td>
<td>9</td>
<td>30</td>
<td>30</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>T_1</td>
<td>17</td>
<td>45</td>
<td>45</td>
<td>9</td>
<td>3</td>
</tr>
</tbody>
</table>

Table-I describes the periodic task set which consists of task ID, weet, period, deadline, number of jobs in a Task, act and priority of each job along with details of Priority exchange server. Table-II describes the aperiodic task set with arrival time and execution time.

Table-III and Table-IV shows the resulting schedule of the given example task set in Table-I and Table-II on DVFSMTS and non-DVFS EDF based priority exchange server scheduler, respectively. Schedule in Table-III shows that at time t = 0, Jobs J_{a0} and J_{b0} are released. As no aperiodic tasks are there, the server capacity is saved at priority of J_{a0} and J_{b0} is executed at the priority of T_{pes}. Job J_{a0} is scheduled for execution between t=0 to t=6.6 by calculating the total utilization upon release, frequency and n_r.p. Upon completion of J_{a0}, utilization is calculated as 0.54. J_{b0} starts execution at t=6.6 and is preempted at t=10 by A_{0}. At t=10, when server is replenished, the saved C_{pes}(ST) is added to the replenished C_{pes} and hence, C_{pes} becomes 4 which is then utilized for executing the aperiodic task A_{0}. A_{0} is completed at t=14. The schedule proceeds in this way. Similarly, Table-IV shows schedule of non-DVFS EDF based priority exchange server for the same example task set.

TABLE II. APERIODIC TASK SET

<table>
<thead>
<tr>
<th>Aperiodic Task ID</th>
<th>Arrival (A_t)</th>
<th>Time (C_t)</th>
<th>Time (Total)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A_0</td>
<td>12</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>A_1</td>
<td>20</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

TABLE III. DVFSMTS SCHEDULE

<table>
<thead>
<tr>
<th>T</th>
<th>R</th>
<th>C</th>
<th>P(ET)</th>
<th>U</th>
<th>f_{act}</th>
<th>A_{et}</th>
<th>Act</th>
<th>Total(ES)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>J_{a0},J_{b0}</td>
<td>-</td>
<td>-</td>
<td>0.67</td>
<td>0.75</td>
<td>J_{a0}(6. 6)</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>6.6</td>
<td>-</td>
<td>J_{a0}</td>
<td>-</td>
<td>0.54</td>
<td>0.75</td>
<td>J_{b0}(12)</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>A_0</td>
<td>-</td>
<td>J_{a0}(6. 5)</td>
<td>-</td>
<td>1</td>
<td>A_{b0}(4)</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>-</td>
<td>A_0</td>
<td>-</td>
<td>0.54</td>
<td>.75</td>
<td>J_{a0}(8. 6)</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>A_1</td>
<td>-</td>
<td>J_{a0}(2)</td>
<td>-</td>
<td>1</td>
<td>A_{b0}(3)</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>-</td>
<td>-</td>
<td>A_{b0}(1)</td>
<td>.54</td>
<td>.75</td>
<td>J_{a0}(2. 6)</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>24.6</td>
<td>-</td>
<td>J_{b0}</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>D</td>
<td>L</td>
<td>E</td>
</tr>
<tr>
<td>30</td>
<td>J_{b0}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>A_{b0}(1)</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>-</td>
<td>A_1</td>
<td>-</td>
<td>0.36</td>
<td>.5</td>
<td>J_{b0}(10)</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>-</td>
<td>J_{b0}</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>D</td>
<td>L</td>
<td>E</td>
</tr>
<tr>
<td>45</td>
<td>J_{b0}</td>
<td>-</td>
<td>-</td>
<td>0.36</td>
<td>0.5</td>
<td>J_{a0}(18)</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>J_{b0}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Dynamic V oltage and Frequency Scaling based on the Priority exchange server (DVFSMTS) has been proposed and implemented for mixed task set. The results are compared with a non-DVFS EDF based priority exchange server. Experimental results show that a considerable amount of energy is saved. Future work will emphasize on implementing the algorithm on multi-core processors.

6. ACKNOWLEDGMENT
I would like to extend my thanks and gratitude to my teachers who have been always helping me with their valuable experiences.

7. REFERENCES


