International Journal of Computer Applications (0975 – 8887)
Volume 100 – No.15, August 2014

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

Sarla Mehariya[1*], Dalpat Songara[3]
Department of Computer Engineering,
Government Women Engineering College, Ajmer, India

Ved Mitra[2] , Mahesh C. Govil[4]
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-

$$P = Ceff \times Vdd^2 \times f \qquad (1)$$

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 (*aet*) is always less than its worst case execution time (*wcet*). The difference that exists is the slack and it 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 RTDVS 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 *aet* 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-DVS – Here, the frequency of each task is first calculated before scheduling in order to achieve better performance on both schedulabilty 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 utilization updating according to the ccEDF algorithm [8].

A. Task model

Our system comprises of *n* periodic tasks, as per task set T = { T_1, T_2,……,T_n }. Each task T_i is described by the parameters {Φ_i, C_i, P_i, D_i,} where, Φ_i, C_i, P_i, D_i represents the phase, *wcet*, period and deadline of the i^{th} periodic task T_i. E_i is the aet of T_i. The aet for any task is a random variable of the wcet. There are *m* Aperiodic tasks, A = {A_1, A_2,…, A_m}. Each aperiodic task is described by A_i = {At_i, C_i}; where At_i is the arrival time and C_i is the execution time of the i^{th} aperiodic task A_i.

B. Server Task

A Priority exchange server T_{pes} is used which is described by T_{PES} = {P_{PES}, C_{PES}}; where, P_{PES} is the server period and C_{PES} is the server capacity. When the tasks are released for execution and if there is no aperiodic load, the C_{pes} is saved at the priority of periodic task executing. C_{pes} also gets replenished with RA i.e. the replenishing amount when the time equivalent to the P_{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) U_i, U_{PES} and U_{Total} represent the utilization of the periodic task T_i, 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) J_{PQ} is the job at the head of the periodic queue *PQ* and J_{AQ} is the job at the head of the aperiodic queue *AQ*.

vii) V_{max} and f_{max} represent maximum voltage and frequency, respectively. V_{est} and f_{est} represents the estimated voltage and frequency.

viii) ***FT*** is the finish time for any task.

ix) $PR(T_i)$ is the priority of the task T_i.

x) A_p and A_a 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. $C_{pes}(ST)$ and $P_{pes}(ST)$ the capacity and period of ST respectively.

NOTE: C_{pes} is the server capacity of T_{pes} and $C_{pes}(ST)$ is the server capacity of a new server task ST. Total server capacity available for an aperiodic task–› C_{pes}=C_{pes}+C_{pes}(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, A, T_{PES}, PQ, AQ, t=0, H.

F. Output

Next job (Periodic or Aperiodic) to be fed to the processor with the estimated scaled frequency f_{est} and voltage V_{est}. 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==Ppes OR t==0)
       Cpes=RA;

    if (UTotal<= 1), then
    {
         Print( Task set schedulable);
         if (AQ != null) and (CPES > 0),then
         {
              n_r_p ← t+min(CPES, ETL(JAQ));
              run JPQ from t to n_r_p at fmax,Vmax;
              CPES ← n_r_p – (min(CPES, FT(JAQ)));
               t ← n_r_p;
```

```
        }
        elseif (PQ != null) then
        {
            n_r_p ← min(t+ETL(J_PQ), A_p, A_a);
            if((n_r_p-t)<C_pes)
                C_pes(ST)←C_pes-(n_r_p-t)  //((n_r_p-t)→
                    //time upto which J_PQ will    execute
            else
                C_pes(ST)←C_pes
            PR(ST)←PR(J_PQ);
            C_pes=C_pes-C_pes(ST);
            Save ST;               //Save ST as a 2nd
                                         // Server task
            execute J_PQ at PR(T_PES) and at f_est and V_est;
            t ← n_r_p;
        }
        else
            Processor is free until the arrival of next
            periodic job or the aperiodic job or the
            replenishment of the server capacity;
    }
    else
        Print(Task set is not schedulable);
            //ERROR MESSAGE
}
END
```

Fig. 1. DVFSMTS Scheduler

```
BEGIN
    If(J_i completed)  )     //J_i is the periodic job
        U_i=E_i/P_i
    Else
        U_i=C_i/P_i
    If(AQ!=null)
        U_pes=C_pes/P_pes
    Else
        Upes=0
```

$U_{Total}=\sum_{i=1}^{n} Ui + U_{pes}$ //n is number of periodic //task and i is an integer variable

```
Calc_Freq():
    Use lowest frequency fest Ɛ (f1,...fmax| f1<f1<…<fmax)
    Such that U_Total<=f_est/f_max
    Return (f_est,U_Total)
END
```

Fig. 2. Algorithm for Frequency Scaling

G. Example Task Set

TABLE I. PERIODIC TASK SET

Task ID	C_i	P_i	D_i	*aet* of each job	Priority PR(T)
T_{PES}	2	10	10	-	1
T_0	9	30	30	5	2
T_1	17	45	45	9	3

TABLE II. APERIODIC TASK SET

Aperiodic Task ID	Arrival Time (At_i)	Execution Time (C_i)
A_0	12	4
A_1	20	3

Table-I describes – the periodic task set which consists of task ID, wcet, period, deadline, number of jobs in a Task, *aet* 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 $J_{o,o}$ and $J_{1,0}$ are released. As no aperiodic tasks are there, the server capacity is saved at priority of $J_{o,o}$ and $J_{o,o}$ is executed at the priority of $T_{PES.}$ Job $J_{o,o}$ 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 $J_{o,o}$, utilization is calculated as 0.54. $J_{1,0}$ 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 III. DVFSMTS SCHEDULE

T	R	C	P(ETL)	U	f_{est}	*Aet*	Total($C_{PES)}$)
0	$J_{o,o}$,$J_{1,0}$	-	-	0.67	0.75	$J_{0,0}$(6.6)	2
6.6	-	$J_{o,o}$	-	0.54	0.75	$J_{1,0}$(12)	2
10	A_0	-	$J_{1,o}$(6.5)	-	1	A_0(4)	4
14	-	A_0	-	0.54	.75	$J_{1.0}$(8.6)	0
20	A_1	-	$J_{1,0}$ (2)	-	1	A_1(3)	2
22	-	-	A_1(1)	.54	.75	$J_{1.0}$(2.6)	0
24.6	-	$J_{1,0}$	-	I	D	L	E
30	$J_{0,1}$	-	-	-	1	A_1(1)	2
31	-	A_1	-	0.36	.5	$J_{0,1}$(10)	1
41	-	$J_{0,1}$	-	I	D	L	E
45	$J_{1,1}$	-	-	0.36	0.5	$J_{1,1}$ (18)	1
63	-	$J_{1,1}$	-	-	-	-	-

TABLE IV. NON-DVFS EDF BASED PRIORITY EXCHANGE SERVER SCHEDULE

T	R	C	P(ETL)	*aet*	C_{PES}
0	$J_{o,o}$, $J_{1,0}$	-	-	$J_{o,o}$ (5)	2
5	-	$J_{o,o}$	-	$J_{1,0}$(9)	2
10	A_0	-	$J_{1,0}$(4)	A_0(4)	4
14	-	A_0(4)	-	$J_{1,0}$ (4)	0
18	-	$J_{1,0}$	I D	L	E
20	A_1	-	-	A_1(3)	0
22	-	-	I D	L	E
30	$J_{0,1}$	-	-	A_1(1)	2
31	-	A_1	-	$J_{0,1}$(5)	1
36	-	$J_{0,1}$	I D	L	E
45	$J_{1,1}$	-	-	$J_{1,1}$ (9)	3
54	-	$J_{1,1}$	I D	L	E

t–Time instant, R–Release, C–Completion of a task, P(ETL) – Execution time left due to preemption, U–Utilization upon Release or Completion, f_{est} – Estimated frequency, *aet* – Actual execution time of a job, C_{PES} – Server capacity.

4. COMPARATIVE ANALYSIS

The performance metrics that have been used for comparison are energy consumption, average response time and idle time. DVFSMTS optimizes energy by scaling down the frequency and voltage because energy is directly proportional to the frequency and square of the voltage. Also that idle time leads to both static and dynamic power consumption. DVFSMTS reduces both by reducing the duration of idle intervals. Hence, scaling down the frequency and voltage to a smaller amount helps in reducing the power consumption.

Periodic task deadlines are not compromised but their response time increases due to the increased execution time. DVFSMTS reduces energy with slight increase in the number of preemption points and response time for periodic task.

A. Experimental Setup

We have implemented both the algorithms for mixed task set in JAVA language. The periodic tasks are generated on a wide range of Hyperperiod and utilization (ranging from 30% to 80%). The number of aperiodic task in a task set can be between 2 to 8. The value of the parameter *aet,* for every periodic task, is determined with the help of random distribution. *aet*'s value lies between 40% – 85% of *wcet.*

Frequency ranges used are – 0.5, 0.75, 1.0 and the corresponding voltage ranges are – 3.0, 4.0, 5.0.

B. Experimental Results

Comparison of DVFSMTS is done with non-DVFS EDF based priority exchange server for the given periodic and the aperiodic task sets.

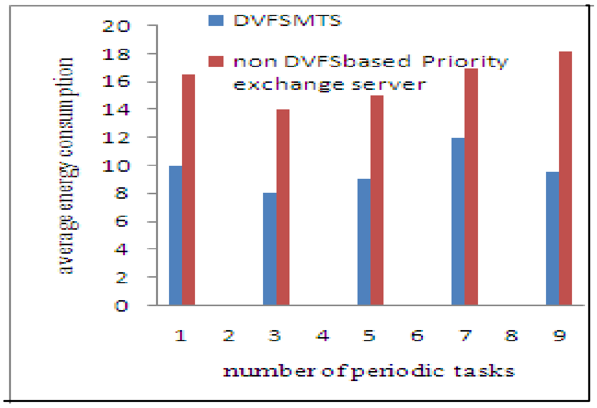

Fig. 3. Average energy consumption

5. CONCLUSION

An energy conscious Real time scheduling algorithm with Dynamic Voltage 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

[1] Shin and Choi, "Power conscious fixed priority scheduling for hard real time systems", Design Automation Conference, Proceedings 36th, 1999.

[2] Gang Quan, Xiaobo Sharon Hu, "Fixed priority scheduling for reducing overall energy on variable voltage processor", Real-Time Systems Symposium, Proceedings 25th IEEE International, 2004.

[3] Guy Martin Tchamgoue, Kyong Hoon Kim, Yong-Kee Jun," Dynamic Voltage Scaling for Power-aware Hierarchical Real-Time Scheduling Framework", Computational Science and Engineering(CSE), IEEE 15th International Conference, 2012.

[4] K. Hakkim ansari, P. Chitra and P. Sonaiyakarthick," Power-aware scheduling of fixed priority tasks in soft real-time multi-core systems", International Conference on Emerging Trends in Computing, Communication and Nanotechnology(ICE-CCN), 2013.

[5] D. Shin and J. Kim, "Dynamic voltage scaling of periodic and aperiodic tasks in priority-driven systems," Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC), Yokohama, Japan, pp. 653–658, 2004.

[6] D. Shin and J. Kim, "Dynamic Voltage Scaling of Mixed Task set in Priority Driven Systems," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 25, No. 3, 2006.

[7] Woonseok Kim, Dongkun Shin, Han-Saem Yun, Jihong Kim and Sang Lyul Min, "Performance Comparison of Dynamic Voltage Scaling Algorithms for Hard Real-Time Systems",8th IEEE Real Time and Embedded Technology and Application Symposium, 2002.

[8] P. Pillai and K.G. Shin, "Real-Time Dynamic Voltage Scaling for Low Power Embedded Operating Systems", 18th ACM Symposium on Operating Systems Principles, 2001.

[9] Real-Time Systems by Jane W.S. Liu, Pearson publications.

[10] Ying Zhang and Krishnendu Chakrabarty," Energy-Aware Adaptive Checkpointing in Embedded Real-Time Systems", Design, Automation and Test in Europe Conference and Exhibition, 2003.

[11] Ying Zhang and Krishnendu Chakrabarty, "A unified approach for fault tolerance and Dynamic power management in Fixed priority real Time Embedded systems", Computer-Aided Design of Integrated Circuits and Systems, IEEE transactions, 2006.

[12] Sasikumar Punnekkat1, Alan Burns, Robert Davis," Analysis of Checkpointing for Real-Time Systems", The International Journal of Time-Critical Computing Systems, 20, 83–102, 2001.

[13] Viacheslav Izosimov, Paul Pop, Petru Eles, Zebo Peng," Synthesis of Flexible Fault-Tolerant Schedules with Preemption for Mixed Soft and Hard Real-Time Systems", Digital System Design, Architectures, Methods and Tools. 11th Euromicro Conference, 2008.

[14] N. Guan, M. Stiggie, W. Yi, G.Yu, "Fixed priority multiprocessor scheduling with Liu and Layland's utilization bounds", Real-Time and Embedded Technology and Applications symposium (RTAS), 2010.

[15] K. Lakshmanan, R Rajkumar and J Lehoczky, "Partitioned Fixed priority preemptive scheduling for multi-core processors", ECRTS '09, 21st Euromicro Conference on Real_time Systems, 2009.

[16] Junyang Lu, Yao Guo, "Energy-Aware Fixed-priority Multi-Core scheduling for Real-Time Systems", Embedded and Real-time Computing Systems and Applications (RTCSA), IEEE 17th Interational Conference, 2011.

[17] Cristina s. Stangaciu, mihai v. Micea, vladimir i. Cretu." Energy efficiency in real-time systems: a brief Overview", IEEE 8th International Symposium on Applied Computational Intelligence and Informatics(SACI), 2013.