**Abstract- —It has been widely known that various benefits can be achieved by reducing energy consumption for high end computing. This paper aims to develop DVFS Enabled Task Scheduling for parallel tasks in a cluster. In this paper, formal models are presented for precedence constrained parallel tasks, DVFS enabled clusters, and energy consumption. This paper puts jobs in a cluster, assign that cluster to a processor so that overall time can be optimize, studies the slack time for jobs, extends their execution time and reduces the energy consumption without increasing the cluster’s execution time as a whole. This paper develops a DETS algorithm for parallel task scheduling Simulation results justify the design and implementation of proposed energy aware scheduling heuristics in the paper.**

1. **INTRODUCTION**

Typically size of data centers are consists of hundreds or thousands of servers and resources, and the size of data centers are getting a huge expansion due to the rapid increase in use of Cloud computing technology. This rapid growth has made a increase in energy consumed in clouds. With power densities doubling every 18-24 months and large-scale HPC systems continuing to increase in size, the amount of heat generated (and hence, temperature) continues to rise. And as a rule of thumb, Arrhenius’ equation as applied to microelectronics notes that for every 10oC increase in temperature, the failure rate of a system doubles. Thus in order to reduce energy consumption there is a need to effectively utilize the resources and executes the requests.

The Dynamic Voltage Frequency Scaling (DVFS) technique enables processors to be operated at multiple frequencies under different supply voltages. Therefore it is possible to reduce the energy consumption of high performance computing by scaling processor supply voltages via the DVFS technique. Our research is devoted to developing scheduling heuristics which reduces energy consumption of parallel task by clustering the tasks to reduce communication energy consumption and using the DVFS mechanism to reduce execution energy consumption. A parallel task in our study is a set of jobs with precedence constraints. Jobs in a parallel task may have some slack time for their execution due to their precedence constraints.

1. **RELATIVE WORK**

This section discusses background and related work of task scheduling, DVFS, and power aware cluster computing.

1. **Parallel task scheduling**

Task scheduling techniques in parallel and distributed systems have been studied in great detail with the aim of making use of these systems efficiently. Task scheduling algorithms[1] are typically classified into two subcategories: static scheduling algorithms and dynamic scheduling algorithms. In static task scheduling algorithms, the task assignment to resources is determined before applications are executed. Information about task execution cost and communication time is supposed to be known at compilation time. The task graph clustering technique [2]is an effective static scheduling heuristic for scheduling parallel tasks. Given a task graph, “clustering” is the process of mapping task graph nodes onto labeled clusters. All tasks of the same cluster are executed in the same processor. In traditional task scheduling heuristics, the process of clustering tasks is an optimization of reducing the makespan of the scheduled graph. In this paper, we proposed a DETS Algorithm whose process of clustering tasks is guided by reducing the makespan of the scheduled graph.

1. **Energy reduction via DVFS technique**

Dynamic voltage and frequency scaling (DVFS) has been proven to be a feasible solution to reduce processor power consumption [3][4]. By lowering processor clock frequency and supply voltage during some time slots, for example, idle or communication phases, large reductions in power consumption can be achieved with only modest performance losses. A DVFS-enabled cluster is a compute cluster where compute nodes can run at multiple power/performance operating points. The DVFS techniques have been applied in the high performance computing fields, for example, in large data centers, to reduce power consumption and achieve high reliability and availability. Our research in this paper make a study on scheduling DAGs on multiple processors in a cluster with DVFS techniques.

1. **MODELS**

The scheduling problem of parallel tasks consists of heterogeneous computing environment l. In this section, we formally describe the mathematical model used to represent the resources in computing environment, precedence-constrained parallel task application and DVFS- enabled energy consumption model, which acts as base for problem deﬁnition in Section IV and scheduling algorithm given in Section V. Table 1 describes the parameters used in the paper to improve the readability.

1. Resource Model

We considered the system represented as set R ={ r1, r2,...,rm}, consisting of m number of DVFS-enabled heterogeneous computing resources interconnected by fully connected communication links. Here the terms resource and processor are used interchangeably. The resources may have different processing capabilities that can operate at different supply voltages (at different frequencies).The frequency transition overhead is considered negligible and is not taken into account in this study. Task executions once started on the processor is considered non-preemptive. Communication time remains unchanged,even after the resource’s supply voltage is scaled down. A DVFS-enabled computing resource can operate on set of supply voltages V and operating frequencies F. For each resource rj ∈ R, the available discrete voltage supply levels are represented by a set Vj ={ vj,1,vj,2,...,vj,N(j)},where N(j) is the number of voltage supply levels or operating points for resource rj. Here vj,min = vj,1 ≤ vj,2 ≤,···≤ vj,N(j) = vj,max. Similarly, the set of operating frequencies are represented as Fj ={fj,1, fj,2,..., fj,N(j)} and fj,min = fj,1 ≤ fj,2 ≤···≤ fj,N(j) = fj,max.

|  |  |
| --- | --- |
| **Attribute** | **Definition** |
| EST | The earliest time is the time the activity can begin |
| EFT | The earliest time of finishing a task on a processor |
| ET | The execution time is the time spent by the system executing the task |
| Wi | The average computational time of the task |
| Ci,j | The communication cost between tasks |
| Pi | The priority of the task |
| N={t1,t2,……,tn} | No. of the tasks |
| PR={pr1,pr2,………,prp} | No of the processors |
| C= {c1,……..cc} | Clusters of the tasks |
| R = {r1,r2,…..,rr} | No of the resource |

TABLE 1

1. TASK MODEL

A parallel application/job consisting of bag of tasks with precedence constraints (workﬂow) is represented by directed acyclic graph(DAG), G=(T,E) where T is the set of vertices (nodes) representing n different tasks ti ∈ T, (1 ≤ i ≤ n) that can execute on any available processor. E is the set of edges eij = (ti, tj) ∈ E, (1 ≤ i ≤ n, 1 ≤ j ≤ n, i ≠ j) representing dependencies among the tasks ti and tj with ti as parent task and tj as the child task. A task without any predecessor is called an entry task, and a task without any successor is called an exit task. The weight w(ti) assigned to task ti represents the number of instructions (MI) to be executed for the task, and weight w(eij) assigned to edge eij represents the amount of data transfer from task ti to tj if they are not executed on the same resource. Figure 1 shows an example of a simple DAG, which consists of ten nodes/tasks. The weight on the edges represents the amount of data transfer or intertask communication cost among the ten tasks. Before giving the formal deﬁnitions of the research issues, the important parameters and notations used are formally deﬁned as follows:

• Earlier Start time (EST):- EST is the time the activity can begin. In other word we can say the earliest point in time when the schedule activity can begin (based on preceding logic and constraints). Alternative names: Early Start Date (ESD) or Early Start (ES). In our algorithm the EST can be calculated based on availability of the resource. If it is starting task then by allocating that task on the resource then EST of that task is the minimum time at which the resource is available. Otherwise the start time of the task will be the time at which all its predecessor tasks has been scheduled and after that the minimum time at which resource is available. if task is starting task then EST(ti,rk) ←Time(rk) else EST(ti,rk) ←Time(rk) + ESTpred Here ESTpred is the maximum of ESTk + Wk + Ck,i of every predecessor of task ti and Time(rk) is the earliest time when resource k is ready for task execution.

EST(ti,,rk)={ Time(rk) ,if ti is starting task

Time(rk) + ESTpred , otherwise}

Execution Time (ET):- The execution time or CPU time of a given task is defined as the time spent by the system executing that task. In our algorithm the execution time represent the estimate execution time of the task. It is defined by –

ET(ti,rj) ← (W(ti) \* CPI) / fj,k

where task ti executing on resource rj at some frequency fj,k and CPI is the number of cycles per instruction of rj which is based on on architecture of the computer and instruction set.

Earlier Finish time (EFT):- EFT is the time the activity can end. In other word we can say the earliest point in time when the schedule activity can complete (based on preceding logic and constraints) Alternative name: Early Finish Date (EFD) or Early Finish (EF). In our algorithm the EFT can be calculated based on EST of the task and ET (Execution Time) of the task on a processor. EFT is calculated by adding task execution time with task start time.

EFT(ti) ← EST(ti) + ET(ti)

Computation cost (W):- Computation cost of the task is the ratio of number of instructions in the task and power consumed by the resource. Calculate the computation cost of every task according to power consumed by available resources.

Wi,r =instructioni\*CPIr/fr

Here instructioni is the number of instructions in the task and CPIr is cycle per instruction of resources and fr is the frequency of resource.

Communication Cost (C):- Communication cost is the time for passing the message between tasks. Communication cost is the cost between task ti and tj using link between resources and task. Communication cost between task ti and task tj is calculated by   
 Ci,j = datai,j / bandwidthr,t.

1. DVFS-Enabled Energy Consumption Model

The main power consumption of the processor is composed of dynamic power (Pdynamic) and static power (Pstatic). The dynamic power consumption arises due to the charging and discharging of node capacitances corresponding to logic gates (capacitance energy) and is approximated as: Pdynamic = A×C ×v2 × f =Ceff ×v2 × f (1)

where A is number of switching activities per clock cycle and C is the total capacitance. Ceff is the average switched capacitance per clock cycle, which is a proportional constant determined by processor/resource and is deﬁned as Ceff =A×C. v is the supply voltage, and f is the operating frequency.

Equation (1) speciﬁes that voltage v is the dominant factor therefore , its reductions lead to reduction in power consumption the most.

The static power consumption is due to running, bias and leakage current. Normally, static power consumption (Pstatic) is proportional to Pdynamic , which is usually less than 30%. Further, overall power consumption of the processor is dominated by the dynamic power. Thus, to simplify our energy consumption model, we focused on the dynamic energy of the processor that can be calculated as:

ξR = Pdynamic ×∆t =Ceff ×v2 × f ×∆t where ∆t is the execution period.

Hence, the energy consumption (ξRi) of the executing precedence-constraint parallel task ti on computing resource rj is deﬁned as:

ξRi = Ceff ×v2j,k × fj,k ×ET(ti,rj) = Ceff ×v2j,k × w∗(ti) where vj,k is the operating voltage of the resource rj at which task ti is executed and w∗(ti) represents the number of cycles required by the task.

1. **RESEARCH PROBLEM DEFINITION**

In this study, we considered is the process of allocating set of n tasks to the set of m number of heterogeneous resources/processors with the aim to achieve minimum energy consumption along with completing the execution of the workﬂow within user-speciﬁed overall deadline D without violating the precedence constraints. Based on the fact that a considerable amount of energy can be saved by slowing down the processor volt- age/frequency, each processor is permitted to execute at different voltage levels (or operating frequencies) such that total energy consumption can be minimized without violating the deadline constraint.

1. **DVFS Enabled Task Scheduling Algorithm**

The first step of the algorithm is to compute rank for each task, some attributes based on information given by the middleware and by the DAG specification. The programming model and/or the middleware can provide information about the size of each task to the scheduler. These sizes can be obtained by application benchmarks, by a history of executions/input data/data sizes, by estimatives given by the programmer, or by estimatives according to past execution and current data sizes. Initially we assign the priority of the tasks and select the task with highest priority and make clusters. then we put that successor of the task whose all parents are clustered and highest priority are put in same clusters in which cluster their parents resides then schedule clusters on different processor so that it gives overall minimum finish time then increase execution time of non-critical tasks using DVFS to reduce energy consumption.



**Algorithm 1. DVFS Enabled Task Scheduling Algorithm**



1. BEGIN

2. FOR each task ti DO

3. Calculate priority pi by calling Priority() function

4. ENDFOR

5. Do clustering of tasks by calling Clustering( ) function

6. FOR each cluster cj DO

7. Schedule cluster cj on processor which gives minimum EFT by calling Schedulingcluster() function

8. ENDFOR

9. FOR each processor pk

10. DO DVFS on pk by calling DVFS() function

11.ENDFOR

12.END



1. **Prioritizing of tasks**

Priority decides the importance of the task. If two tasks that are scheduled for the same time and the same resource, then we need to rank them to decide which one to execute first. This 'might' be due to time (ie: one affects other downstream work), or it could be by duration (one will take a day while the other a week). In our algorithm we assign priority to the tasks based on computation time and communication time of the tasks.



**Algorithm 2. Priority() function**



1. BEGIN

2. FOR each task ti DO

3. if ti is the last task

4. then pi ← wi

5. else

6. pi ← wi + max (ci,j+pj) {tj Ɛ succ(ti)}

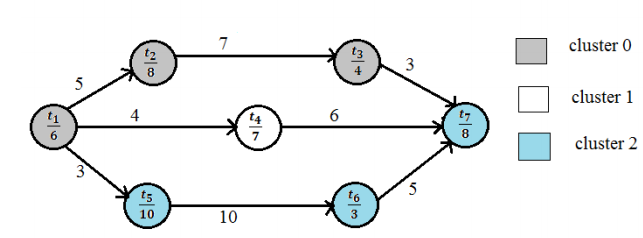
7. ENDFOR

8. END

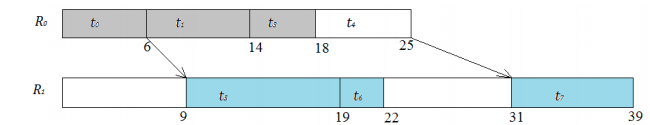


1. **Clustering of tasks**

This section discuss how to create clusters of tasks to begin the scheduling process. Figure 1 is an example of parallel task to be scheduled. In Figure 1, job IDs and job ` execution costs are marked inside the jobs and the communication costs are labeled on the links. The scheduled task graph is shown in Figure 2 as a Gantt chart. For clustering , priority attribute select the first task to be added to the first cluster (clustering phase). The first node (or task) ti selected to compose a cluster clk is the unscheduled node with the highest priority. It is added to clk, then starts a depth-first search on the DAG starting on ti, selecting ts ∈ succ(ti) with the highest priority and adding it to clk, until ts has a non scheduled predecessor. The task ts that has a non scheduled predecessor is not included in the clk. With this, clusters always have only tasks with all predecessors already scheduled or to be scheduled along with them.



**Fig-1 Clustering of Tasks**



**Fig-2 Scheduling of Task on processor**

**Algorithm 3. Clustering() function**



1. BEGIN

2. sort tasks acc. to their priority

3. FOR each task tj

4. Label:

5. Add task tj to Cluster clk

6. Sort children of task tj

7. FOR each child ti of task tj DO

8. IF ti is not in cluster and has no non scheduled predecessor THEN

9. tj←ti

10. Goto label

11. ENDFOR

12. IF( j=p+1)

13. K++;

14. ENDIF

15. ENDFOR

16.END

1. **Schedulingclusters to processor**

For each cluster created, selects a resource to schedule it. After creating clusters then it must decide where it will be scheduled. In the processor selection phase, it looks for the resource which minimizes the EFT of the cluster. Thus, the criterion to choose the processor to a cluster clk is to minimize the EFTclk , defined as EFTclk = maximum of EFTti where ti∈ clk



**Algorithm 4. Schedulingcluster() function**



1. BEGIN

2. FOR each cluster cli DO

3. EFTi ← INT\_MAX, chosenvm

3. FOR each virtual machine vmj

4. EFTi,j← INT\_MIN

5. FOR every task tk of cluster cli

6. Calculate EFT of tk,i on vmj

7. If EFT(tk,i)>EFTi,j

8. EFTi,j ← EFT(tk,i)

9. ENDFOR

10. IF EFTi,j<EFTi

11. THEN EFTi←EFTi,j

12. chosenvm←j

13. ENDIF

14. ENDFOR

15. ENDFOR

16. END



1. **DVFS on scheduled task**

After assigning a processor to every cluster. We need to find total energy consumption. So that by using DVFS total energy consumption can be minimized.So for every task in cluster find execution time and find total execution time of that cluster by adding execution time of each task in that cluster. Idle time of each cluster can be calculated by subtracting total execution time of each cluster from time at which its corresponding processor works. For Execution time and idle time processor runs on maximum and minimum frequency respectively. So overall energy can be calculated by-

Overall Energy Consumption = coefficientofprocessor\*(maxvoltage)2\*executiontime + coefficient of processor\*(minvoltage)2\*idle time .

Now the energy can be optimize using DVFS. In DVFS the execution time of a task on a processor gets increased ,by lowering the frequency of the processor for that particular task. For that we find set the frequency of processor so that task finish time gets nearest less than or equal to the start time of next task on that processor.

Execution Time=Weight of Task\*Cycles Per Instruction of Vm/Frequency of that Vm.

Overall new Energy Consumption = for every task i on that processor(coefficient of processor\*(DVFS voltagei)2\*execution timei)+coefficientofprocessor\*(minvoltage)2\*idle time

|  |  |  |
| --- | --- | --- |
| **Level** | **Voltage** | **Relative Speed(%)** |
| **0**  **1**  **2**  **3**  **4**  **6** | **1.5**  **1.4**  **1.3**  **1.2**  **1.1**  **1.0** | **100**  **90**  **80**  **70**  **60**  **50** |



**Algorithm 5. DVFS() function**

1. BEGIN

2. FOR each cluster ct DO

3. newETi ← oldETi + ESTi+1 – EFTi

4. fp ←oldETi \* max\_fp /new ETi

5. round off fp to nearest frequency level

6. ENDFOR

7. END

1. COMPLEXITY

We now analyze the time complexity of our proposed EATS algorithm. We consider that the EATS algorithm receives workﬂow (DAG) with n number of tasks and e number of edges. Further, we consider that there is m number of available resources.First of all, Algorithm 1 calculates priority of each task in line(2-4).

1. REFERENCES

[1] Z. &. S. W. Yu, "An adaptive rescheduling strategy for grid workflow applications," IEEE International Parallel and Distributed Processing Symposium, vol. 3, no. 4, pp. 18, 2007.

[2] L. F. &. M. E. R. Bittencourt, "Towards the scheduling of multiple workflows on computational grids," Journal of grid computing, vol. 3, no. 8, pp. 419-441, 2010.

[3] L. T. J. v. L. G. &. C. D. Wang, "Power Aware scheduling for parallel tasks via task clustering," Parallel and Distributed Systems (ICPADS), pp. 629-634, 2010.

[4] L. K. V. B. &. V. S. Goh, "Design of fast and efficient energy-aware gradient-based scheduling algorithms heterogeneous embedded multiprocessor systems," IEEE Transactions on Parallel and Distributed Systems, vol. 1, no. 20, pp. 1-12, 2009.