International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 2 Issue: 4 911 – 914

A Survey on Various Parallel Power Aware Task Scheduling Algorithms for Reducing Power Consumption

Shruthi S[1]
PG Student, Dept. Of CS&E
Acharya Institute of Technology
Bangalore, India
Shruthi_s16@yahoo.co.in

Nagaveni V[2]
Assistant Professor, Dept. Of CS&E
Acharya Institute of Technology
Bangalore, India
nagaveni@acharya.ac.in

***Abstract*—** A recent issue in computing systems is Power aware scheduling problem. The Scheduling problem reduces the system reliability and availability and increases the operational cost. As recent commodity processors support multiple operating points under various supply voltage levels, Dynamic voltage and frequency scaling (DVFS) is a commonly used power Aware Management technique that can reduce power consumption by decreasing the clock frequency of a processor it leads to reduction in the supply voltage and it saves power on a wide range of computing systems. In this paper, we provide three different power aware scheduling algorithms to minimize power consumption of parallel task. The three algorithms are Power Aware List Scheduling (PALS), Power Aware Task clustering (PATC) and Low Energy Scheduling (LEneS) algorithms. Using these algorithms we can achive maximum energy saving.

Keywords-- *Power aware scheduling, Parallel task, PALS, PATC, LEneS.*

*****

I. INTRODUCTION

Power consumption of computing system has become a major concern in nowadays technologies. The objective of power aware computing is to improve power consumption using power aware ability of system devices, such as processors, disks, and communication links. There are two main reasons for need of power aware computing in systems: operational cost and system reliability. One dominating factor in the operational cost of data centers comes from electricity cost consumed by server systems [1]. Dynamic voltage and frequency scaling (DVFS) is a commonly used power aware management technique [2] where the clock frequency of a processor is decreased to allow a corresponding reduction in the supply voltage and it saves power on a wide range of computing systems, from embedded, laptop and desktop systems. Power aware task scheduling algorithms schedules the task to reduce power consumption of parallel task execution by using the DVFS mechanism. A parallel task 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.

This paper makes a study of task scheduling algorithms to reduce power consumption. First issue is to minimize power consumption of parallel task using power aware techniques. This paper introduces three list based power aware task scheduling heuristic for parallel tasks— Power Aware List Scheduling (PALS) [3], Power Aware Task Clustering (PATC) [3] and Low Energy Scheduling (LEneS) [4] algorithms. The task has been scheduled using these algorithms to find how much power has been saved from each algorithm.

A. Dynamic voltage and frequency scaling (DVFS) Technique

Dynamic voltage and frequency scaling is a commonly used power management technique [2] where the clock frequency of a processor is decreased to allow a corresponding reduction in the supply voltage and it saves power on a wide range of computing systems, from embedded, laptop and desktop systems to high performance server class systems. This reduces power consumption, which can lead to significant reduction in the energy required for a computation, particularly for memory bound workloads.

This survey is organized as follows. Section II introduces basic concepts Power aware scheduling. Section III, IV and V covers Power Aware Task scheduling algorithms called PALS, PATC and LEneS respectively. Section VI describes comparison study of power aware scheduling algorithms.

II. POWER AWARE TASK SCHEDULING

Task scheduling techniques in parallel and distributed systems have been studied with the aim of making use of systems efficiently. Task scheduling algorithms are typically classified into two categories:

- Static Task Scheduling.
- Dynamic Task Scheduling.

A. Static Task Scheduling.

In Static Task Scheduling, the task assignment to resources is determined before applications are executed. Information about task execution cost and communication time is to be known at compilation time [3]. Static task scheduling algorithms are non pre-emptive a task. It is always running on the resource to which it is assigned. Static scheduling sometimes results in lower execution times than dynamic scheduling [5] and it allows only one process per processor. Static scheduling can be used to predict the speedup that can be achieved by a particular parallel algorithm on a target machine.

B. Dynamic Task Scheduling

Dynamic Task scheduling schedule tasks to resources in the runtime to achieving load balance among Processing Elements (PEs) are based on the redistribution [3].

The list scheduling algorithm is the most popular algorithm in the Static Task scheduling [6, 7]. List based scheduling algorithms assign priorities to each tasks and sort tasks into a list ordered in decreasing order. Then tasks are scheduled based on the priorities. The three list based scheduling heuristic for parallel tasks are — Power Aware List Scheduling (PALS), Power Aware Task Clustering (PATC) and Low Energy Scheduling (LEneS) algorithms. These algorithms helps to reduce the power consumption of parallel task, hence the name indicates Power aware scheduling algorithms

III. PALS ALGORITHM

PALS (Power Aware List Scheduling) algorithm is a power aware scheduling heuristics for parallel tasks and a list based scheduling algorithm to find the best effort task response time and Non critical time slot voltage scaling algorithm to scale down non-critical jobs and it tries to reduce the energy consumption [3]. Lizhe Wanga et al. propose PALS. The PALS algorithm employs the ETF (Earliest Task First), a list based scheduling algorithm to find the best effort task response time for T. Then, it tries to reduce the energy consumption with the following methods.

- Scale down PE's voltages to a proper level, thus extending the execution time of the non critical jobs without affecting the critical path.
- Scale the PE's voltage when it is idle or when it is in the data communication phase.

A. The PALS algorithm

1. Schedule tasks via the ETF scheduling Algorithm
2. Scale down PE's voltages for all non-critical jobs with Non-critical time slot voltage scaling algorithm

B. The ETF (Earliest Task First) scheduling algorithm

ETF algorithm is Earliest Task First, a list based scheduling algorithm [3] to find the best effort task response time for T.ETF allocates each job with a priority. ETF selects ready jobs with the highest priority and schedules it on the PE with earliest task starting time. The algorithm of ETF is shown below.

1 job_n.level: priority of task jobn ∈ J
2 ready_job_list: list of jobs that are ready to be executed
3 PE_list: list of PEs
4 $pe_k.t^{available}$ PE's available time.
5 BEGIN
6 FOR each job job_n ∈ J DO
7 compute job_n.level
8 ENDFOR
9 put all ready jobs into ready_job_list
10 sort all jobs job_n ∈ ready_job_list in decreasing order of job_n.level
11 put all PEs into PE_list
12 sort all PEs $pe_k.t^{available} = 0$
13 REPEAT
14 IF (ready_job_list≠ ∅) THEN
15 get a job, job_n, from ready_job_list
16 get a PE, pe_k, which has the earliest available time $pe_k.t^{available}$
17 schedule job_n on pe_k
18 Arrange the communicate phase, calculate starting time and finish time of job_n on pe_k
19 delete the task from ready_job_list
20 update PE_list with increasing order $pe_k.t^{available}$
21 ENDIF
22 update ready_job_list
23 UNTIL (every job job_n ∈ J has been scheduled)
24 END

B. Non-critical Time slot Voltage scaling algorithm

The Non-critical Time slot Voltage Scaling algorithm shows how to scale down non-critical jobs [3]. For each PE, it scans all time slots. When the PE is idle or transfers data in a time slot, algorithm scales the PE's operating frequency to the lowest. When a time slot executes a non-critical job, it calculates its slack time, extends the job's execution time to the slack time, and scales down the PE's operating frequency to a proper value. The algorithm of Non-critical Time slot Voltage Scaling is shown below.

1 BEGIN
2 FOR each PE pe_k DO
3 FOR each time slot in pe_k's Gantt chart DO
4 IF pe_k is idle or it executes a communication phase THEN
5 Scales down pe_k's operating frequency to lowest
6 ENDIF
7 IF pe_k executes a non-critical job job_n THEN
8 Calculate job_n.slack as using
$job_n * slack = job_n * t^{end} - job_n * t^{st}$
9 Scale pe_k's frequency to $pe_k.f^{op}$ as
$pe_k * f^{op} = f_{max} * job_n * t^0 / job_n * slack$
10 END IF
11 ENDFOR
12 ENDFOR
13 END

IV. PATC ALGORITHM

PATC (Power Aware Task Clustering) algorithm is also a list based power aware scheduling heuristics for parallel tasks, which guides the edge zeroing process with objective of reducing power consumption. Lizhe Wanga et al. propose PALS [3]. The PATC algorithm firstly marks all edges as unexamined and allocates each task a separate cluster. After sorting all edges in descending order of communication time, the PATC algorithm repeatedly merges tasks by zeroing the edges with high communication cost if the total power consumption is not increased. The algorithm of PATC is shown below.

1 BEGIN
2 Initially all edges are marked unexamined and each task forms a separate cluster
3 Sort all edges in a descending order according to their communication costs
4 REPEAT
5 Zero the highest unexamined edge in the sorted list if the power consumption of the scheduled task graph does not increase
6 Mark the edge examined
7 When two clusters are merged, the tasks are ordered according to their b_level.
8 UNTIL all edges are marked examined.

A. b_level calculation

b_level calculation used in PATC algorithm. b_level allocates each job in bottom level with a priority. The algorithm of b_level is shown below.

1 BEGIN
2 r_list ←a list of all jobs $Job_i \in J$ sorted in a reversed partial order
3 Initialize all jobs in rtopo_list: b_level(Job_i) ← 0
4 FOR each Job $Job_i \in$ rtopo_list DO
5 max_length ← 0
6 FOR each immediate succeeding job Job_j of job Job_i DO
7 length ← b_level(Job_j) + $e_{i,j}$.cost
8 IF (length > max_length) THEN
9 max_length ← length
10 ENDIF
11 ENDFOR
12 b_level(Ti) ← Job_i.weight + max_length
13 END FOR
14 END

V. LENES ALGORITHM

LEneS (Low Energy Scheduling) approach is based on a list-scheduling heuristic [4]. Gruian et al. propose a LEaneS. It smartly introduces enhanced task graphs (ETG) and energy gain in the list based scheduling [8] with dynamic recalculation of priorities, and assumes a given allocation and assignment of tasks to processors. This approach minimizes the energy by choosing the best combination of supply voltages for each task running on its processor. The set of experiments of LEnesS is based on list scheduling with a energy sensitive priority function. In every scheduling step, the node priorities change and have to be recalculated. Moreover, the priority function is tuned during several scheduling attempts. Whenever a scheduling attempt fails (the deadline is violated), the priority function is adjusted and a re-scheduling is attempted. The algorithm of LEneS is shown below.

1 procedure LEneS(ETG)
2 set all $\{a_i\} = 0$**;**
3 while list-scheduling (ETG, g) fails do
4 let $\{a_k\}$ be the coefficients of the nodes on the critical path
5 if all $\{a_i\} >= Max\alpha$ then scheduling fails
6 else if all $\{a_k\} = Max\alpha$ then increase all a_i by10%
7 else increase only $\{a_k\}$ by 10%
8 end procedure LEneS

VI. PERFORMANCE STUDY

This section demonstrates how the power aware task scheduling algorithms can effectively save energy without performance degradation.

A. *Performance of PALS Algorithm*

The performance study on the PALS algorithm is described. Several task sets are generated with the Synthetic DAG generation tool. The scheduling algorithm used how much energy is saved in various parallel tasks and PE numbers. We define the resource competition to execute a parallel task, ζ (T), in a cluster as follows:

$$\zeta (T) = N / P$$

where, T is the parallel task, N is the job number of T, and P is the PE number for executing T. Resource competition shows the task execution situation, like how many precedence exist between jobs, how many jobs are scheduled, and how many jobs are executed on each PE.

- The PALS reduce the energy consumption during the communication phase.
- The PALS reduce power consumption when a PE is idle, and
- The PALS try to extend job slack time whenever it is possible.

B. *Comparision Study of Various Power Aware Task Scheduling Algorithms*

Table 1 describes the comparison of power savings between different Power aware task scheduling algorithms.

Table 1 Comparison of power savings between different Power aware scheduling algorithms.

Power Aware Task Scheduling Algorithms	Maximum Power Saving (%)
PALS (3)	44.3
PATC (3)	39.7
LEneS (4)	28.0

The maximum power saving from PALS is 44.3%.
The maximum power saving from PATC is 39.7%.
The maximum power saving from LEneS is 28.0%.

The PATC and PALS reduce the power consumption during the communication phase, when a PE is idle, and when it tries extend job slack time whenever it is possible. The PALS and PATC saves upto 44.3% and 39.0% respectively. LEneS handles designs with dependent tasks mapped onto dynamic supply voltage processors. LEneS uses list scheduling and a special priority function to derive static schedules with low energy consumption. LEneS saves up to 28% power without any performance loss.

VII. CONCLUSION

Minimizing energy for precedence constrained parallel task execution using scheduling heuristic algorithms has proven to be a highly effective techniques to achieve low power consumption for computing system. Among three power aware scheduling algorithms PALS is more efficient to reduce the power consumption because it saves 44.3% power than other two scheduling algorithms.

REFERENCES

[1] Kyong Hoon Kim, Rajkumar Buyya, Jong Kim “Power Aware Scheduling of Bag-of-Tasks Applications with Deadline Constraints on DVS-enabled Clusters”.

[2] Etienne Le Sueur and Gernot Heiser “Dynamic Voltage and Frequency Scaling: The Laws of Diminishing Returns” NICTA and University of New South Wales.

[3] Lizhe Wanga,b, Samee U. Khanc, Dan Chena, Joanna Kołodziej d, Rajiv Ranjan e, Cheng-zhong Xuf, Albert Zomayag “Energy-aware parallel task scheduling in a cluster” Future Generation Computer Systems 29 (2013) 1661–1670.

[4] Flavius Gruian, Krzysztof Kuchcinski, LEneS: task scheduling for low-energy systems using variable supply voltage processors, in: Proceedings of Asia and South Pacific Design Automation Conference, 2001, pp. 449–455.

[5] Esquivel S.C., Gatica C. R., Gallard R.H. ”Solving the Parallel Task scheduling problem by means of genetic Approach” UNSL-338403.

[6] Abdellatif Mtibaa, Bouraoui Ouni, Mohamed Abid, An efficient list scheduling algorithm for time placement problem, Comput. Electr. Eng. 33 (4) (2007) 285–298.

[7] Rongheng Li, Huei Chuen Huang, List scheduling for jobs with arbitrary release times and similar lengths, J. Sched. 10 (6) (2007) 365–373.

[8] F. Gruian and K. Kuchcinski, “Low-energy directed architecture selection and task scheduling for system-level design,” Proceedings of the 25th Euromicro Conference 1999, pp. 296-302.