Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2016) 24: 4117 – 4128

doi:10.3906/elk-1505-112

Energy-aware stochastic scheduling model with precedence constraints on DVFS-enabled processors

Mohammad SAJID*, **Zahid RAZA**
School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India

Received: 14.05.2015 • **Accepted/Published Online:** 11.07.2015 • **Final Version:** 20.06.2016

Abstract: The stochastic scheduling of precedence-constrained jobs on a heterogeneous processor is a challenging problem that requires solutions with one or more optimized QoS parameters. In this work, an energy-aware stochastic algorithm is proposed to schedule the batch of precedence-constrained jobs on heterogeneous DVFS-enabled processors with the objective of optimizing turnaround time and energy consumption. The processing time of tasks in all jobs and their precedence-constraint times are governed by independent probability distributions. The performance of the proposed stochastic algorithm is compared with SHEFT and ECS based on randomly generated batches of different sizes. The experimental study reveals that the proposed algorithm significantly outperforms the SHEFT and ECS algorithms in terms of turnaround time and energy consumption.

Key words: Stochastic scheduling, batch of stochastic precedence-constrained jobs, slack sharing, DVFS-enabled processors, turnaround time, energy consumption

1. Introduction

The fastest supercomputer, Tianhe-2, consists of 16,000 physical nodes, each node having 2 Intel Xeon Ivy Bridge and 3 Xeon Phi processors with overall power consumption of 17.808 MW [1]. The production of 1 kW of electricity power consumes approximately 0.4 kg of coal and 4 L of water, and produces 0.272 kg of solid powder, 0.997 kg of CO_2, and 0.03 kg of SO_2. The data centers consist of thousands of supercomputers, which results in very high power consumption; therefore, the power consumption of data centers is a considerable issue due to the associated negative environmental effects, high monetary costs, and reduced reliability of the systems [2]. To address this issue, various techniques, including resource hibernation, dynamic voltage-frequency scaling (DVFS), resource consolidation, and memory optimization, have been proposed [2]. DVFS is a useful technique that automatically adjusts the inputs of CMOS-enabled processors by scaling the frequency up and down to reduce power consumption [2]. For DVFS-enabled processors, the scheduling algorithms must consider the assignment of jobs to processors as well as the selection of frequency for execution in order to exploit the idle slots that occur due to precedence constraints. In general, the scheduling of precedence-constrained jobs has been proven to be NP-complete [3]. Furthermore, the scheduling on heterogeneous DVFS-enabled processors to optimize turnaround time as well as energy consumption makes it an even more challenging problem [4]. A plethora of energy-aware and nonenergy-aware algorithms have been proposed considering deterministic processing and communication times [5–11]. For a deterministic scheduling problem, all the parameters regarding the job and system are known beforehand and cannot deviate from the given parameters.

*Correspondence: sajid.cst@gmail.com

For real environments, the parameters of the job as well as the system can deviate considerably due to unknown operating system conditions, memory access time, and so on. The stochastic scheduling model considers the parameters as a random variable with a given probability density function and makes an effort to optimize the expected performance [12–15]. In realizing this, this work proposes an energy-aware stochastic algorithm that schedules the batch of precedence-constrained jobs on many heterogeneous DVFS-enabled processors to optimize the turnaround time and energy consumption. The processing times and precedence-constraint times follow the independent normal probability distributions.

The organization of the remaining paper is given as follows. Section 2 discusses the related research for energy-aware and stochastic scheduling. Section 3 formulates the energy-aware stochastic scheduling problem on a heterogeneous DVFS-enabled computation system. Section 4 explains the ESHEFT algorithm, while Section 5 presents the experimental study for different batches. Section 6 presents the concluding remarks.

2. Related research

Scheduling algorithms with the objective of minimizing the turnaround time and energy consumption, an active research area with various energy-aware algorithms, are proposed for a DVFS-enabled heterogeneous computing system. Zhu et al. introduced the concept of slack sharing, which reclaims the time unused by a task to reduce the frequency of processors for reducing the energy consumption on a homogeneous computing system (HCS). The two algorithms using the slack sharing concept, GSSR and FLSSR, were proposed for independent and dependent tasks, respectively [7]. Zhang et al. presented many energy-efficient algorithms for the HCS with changeable continuous and discrete speeds for reducing energy consumption while meeting time deadlines. Simulation studies indicate that for both continuous and discrete speeds computers, the hybrid algorithms have superior performance and offer the best task schedules [8]. Lee and Zomaya presented 2 energy-aware algorithms, ECS and ECS + idle, in order to schedule the dependent tasks on a DVFS-enabled HCS based on the relative superiority metric (RS) and makespan-conservative energy reduction technique (MCER) [9]. Li explored 2 energy-aware models, i.e. a problem to optimize turnaround time with energy consumption as the constraint and a problem to optimize the energy consumption with turnaround time as the constraint, in order to schedule dependent tasks on DVFS-enabled processors. Li proposed 3 types of algorithms (prepower-determination, postpower-determination, and hybrid algorithms) to solve each subproblem efficiently [10]. Zhuravlev et al. presented a survey of energy-cognizant scheduling algorithms considering 3 types of hardware mechanisms: DVFS-enabled processors, thermal management, and asymmetric multicore designs [11].

For stochastic scheduling, the well-known algorithms are SEPT, WSEPT, and LEPT, which take into consideration the expectation of execution times for making scheduling decisions. Skutella and Uetz proposed constant-factor approximation algorithms for precedence-constrained stochastic tasks to optimize the total weighted completion time on the HCS. For precedence-constraint scheduling with and without release dates on m processors, the CMNS [12] algorithm (with $\kappa > 0$) has been proven with a performance guarantee of $(1 + \kappa)(1 + (1/\kappa) + \max\{1, ((m-1)/m)\Delta\})$ and $(1 + \kappa)(1 + ((m-1)/m\kappa) + \max\{1, ((m-1)/m)\Delta\})$, respectively [13]. Tang et al. proposed SHEFT to address the problem of scheduling precedence-constrained stochastic tasks on the HCS employing the average execution time of tasks based on expectation and variance [14]. Li et al. presented the SDLS algorithm to schedule precedence-constraint tasks using the stochastic bottom level and stochastic dynamic level on the HCS for independently normally distributed processing and communication times [15].

It is to be noted that either energy/nonenergy-aware algorithms are proposed for jobs with deterministic processing times or nonenergy-aware approximation/heuristics algorithms have been reported in the literature.

This work presents an energy-aware stochastic scheduling algorithm for a batch of precedence-constrained jobs with stochastic processing times to schedule on per-chip DVFS-enabled heterogeneous processors.

3. Problem formulation

This section explains the computation system employed, batch model, and energy model, along with the problem statement.

3.1. Computation system

The computation system C is a HCS of K different per-chip DVFS-enabled processors represented by $C = \{p_x: 1 \leq x \leq K\}$. Each processor $p_x \in C$ can run on n_x discrete voltage levels, and the set of n_x voltage levels is given by $V_x = \{v_{x,z} : 1 \leq= x \leq= n_x\}$, such that if y $<$ z, $v_{x,y} > v_{x,z}$. For each voltage level $v_{x,z}$, there exists a corresponding frequency $f_{x,z} \in p_x$, and the set of n_x frequency levels for processor p_x is given by $F_x = \{f_{x,z}: 1 \leq z \leq n_x\}$. The total number of frequency/voltage levels (N_C) in computation system C will be equal to the sum of all the voltage/frequency levels of all processors, i.e. $N_C = \sum_{x=1}^{K} |V_x|$. The DVFS-capability allows processors to switch from one voltage level to another with a trade-off between the power consumption and processing time. Therefore, the processing time of any task at the highest frequency (f_x^{max}) of processor p_x is minimum, whereas energy consumption is maximum. It is assumed that all processors are connected using a fast interconnection network and the intracommunication cost between is zero, while intercommunication cost is nonzero. The present computation system C can be extended to multicore DVFS-enabled or multicore per-core DVFS enabled computation systems.

3.2. Batch model

Batch B consists of a collection of M independent jobs to be executed on computation system C and is represented by $B = \{J_i: 1 \leq i \leq M\}$, with each job $J_i \in B$ consisting of multiple dependent tasks. Each job $J_i \in B$ is given in the form of a directed acyclic graph (DAG) as $J_i = (T_i, E_i)$, where $T_i = \{t_{i,j}: 1 \leq j \leq M_i\}$ represents the set of M_i dependent and atomic tasks, and $E_i = \{e_{i,(j,k)}: 1 \leq j \leq k \leq M_i\} \subset T_i \times T_i$ the set of precedence-constrained edges between tasks. The edge $e_{i,(j,k)} \in E_i$ represents the precedence constraint between tasks $t_{i,j}$ and $t_{i,k}$ such that task $t_{i,k}$ starts its execution after completion of task $t_{i,j}$. Task $t_{i,k}$ is the successor task of task $t_{i,j}$, and the set of all successor tasks of $t_{i,j}$ is given by $Succ_{i,j} = \{t_{i,k}: e_{i,(j,k)} \in E_i\}$. The predecessor task of task $t_{i,j}$ is connected by the edge $e_{i,(k,j)} \in E_i$, and the set of predecessors of task $t_{i,j}$ is given by $Pred_{i,j} = \{t_{i,k}: e_{i,(k,j)} \in E_i\}$. A task $t_{i,j}$ having zero predecessor tasks, i.e. $Pred_{i,j} = \Phi$, is called the entry task, and a task $t_{i,j}$ having zero successor tasks, i.e. $Succ_{i,j} = \Phi$, is called the exit task. The level of tasks $t_{i,j}$ is 1 if task $t_{i,j}$ has zero predecessor tasks, i.e. $Pred_{i,j} = \Phi$. On the next level, the successors of the first level's tasks are stretched out, and so on. The level of task $t_{i,j}$ is determined recursively as follows:

$$L_{i,j} = \begin{cases} 1, & Pred_{i,j} = \Phi \\ 1 + \max\{L_{i,k} : \forall t_{i,k} \in Pred_{i,j}\}, & \text{otherwise} \end{cases}. \quad (1)$$

The level of the batch (L_B) is the maximum level of any task $t_{i,j} \in B$ and is computed as:

$$L_B = \max\{L_{i,j} : \forall t_{i,j}, Succ_{i,j} = \Phi\} \quad (2)$$

The tasks of all jobs are randomly distributed over different levels from 1 to L_B.

3.3. Scheduling design principles

Stochastic scheduling makes an effort to optimize the expected performance of the solutions under the assumption that parameters are random variables with known probability distributions. In this work, the processing times of each task $t_{i,j} \in T_i$ and precedence-constraints edge $e_{i,(j,k)} \in E_i$ in a job J_i follow the independent normal probability distributions [12–17]. If the processing time of task $t_{i,j}$ and precedence-constraint time of edge $e_{i,(j,k)}$ follow the normal probability distribution with expectations and variances as $\mu, \mu'; \sigma^2, \sigma'^2$, respectively, then $t_{i,j} \sim N(\mu, \sigma^2)$ and $e_{i,(j,k)} \sim N(\mu', \sigma'^2)$. The processors deployed in computation system C are heterogeneous DVFS-enabled; therefore, the processing time of each task $t_{i,j} \in T_i$ with respect to each frequency level $f_{x,z} \in F_x$ will have different expectations and variances. For example, the processing time of task $t_{i,j}$ w.r.t. $v_{x,z}$ will follow an independent normal probability distribution with expectation $\mu_{i,j,x,z}$ and variance $\sigma^2_{i,j,x,z}$, i.e. $t_{i,j} \sim N(\mu_{i,j,x,z}, \sigma^2_{i,j,x,z})$. In the theory of randomness, the performance of stochastic scheduling depends on the function Δ of expectation as well as variance for random processing time Y where $\mathrm{Var(Y)/E[Y]}^2 \leq \Delta$ [12, 14–17]. For $\Delta > 1$, the performance of stochastic scheduling increases as the variance of random variable decreases. For others, the performance is affected by the sum of expectation and variance of the random variable. Therefore, the processing times and precedence-constraint times are computed based on the expectation and variance of the random variable. The approximate value (AV(Y)) of a random number Y using mean (E[Y]) and variance (Var[Y]) can be determined as:

$$AV(Y) = \begin{cases} E[Y] + \sqrt{Var[Y]} \text{ if } \frac{Var[Y]}{E[Y]^2} \leq 1 \\ E[Y]\left(1 + \frac{1}{\sqrt{Var[Y]}}\right) \text{otherwise} \end{cases}. \tag{3}$$

Let $[w_{i,j,x,z}]$ be a matrix of order $M \times M_{\max} \times K \times N_C$, where $w_{i,j,x,z}$ gives the approximate value $(AV(t_{i,j}))$ of the processing time of task $t_{i,j} \in T_i$ with regard to frequency level $f_{x,z} \in F_x$ on processor p_x using Eq. (3), and $M_{\max}$ represents the maximum number of tasks in any job of the batch. Let $w_z^{max}(t_{i,j})$ give the approximate processing time of task $t_{i,j} \in T_i$ with regard to the fastest frequency (f_x^{max}) of processor p_x. Let $[w_{i,(j,k)}]$ be a matrix of order $M \times M_{\max} \times M_{\max}$; $w_{i,(j,k)}$ gives the approximate precedence-constraint time of edge $e_{i,(j,k)}$ between tasks $t_{i,j}$ and $t_{i,k}$. The average processing time of task $t_{i,j}$ with respect to the DVFS-enabled computation system C can be written as:

$$AP(t_{i,j}) = \sum_{x=1}^{K} \sum_{z=1}^{n_x} w_{i,j,x,z} \Big/ \sum_{x=1}^{K} n_x. \tag{4}$$

The stochastic b-level $(sb_{i,j})$ of task $t_{i,j}$ in the batch B with respect to the DVFS-enabled computation system C can now be rewritten as [5,6]:

$$sb_{i,j} = \begin{cases} AP(t_{i,j}) & Succ_{i,j} = \Phi \\ AP(t_{i,j}) + \max_{t_{i,k} \in Succ_{i,j}} \{sb_{i,k} + w_{i,(j,k)}\} & \text{Otherwise} \end{cases}. \tag{5}$$

To compute the turnaround time of the whole batch, it is required to define the start time $(ST_{i,j,x})$, the finish time $(FT_{i,j,x})$ for task $t_{i,j} \in B$ scheduled on processor $p_x \in C$, the allocation metric $(X_{i,j}^{x,z})$, and the ready time (RT_x) of processor p_x. Let $X_{i,j}^{x,z} = 1$ if task $t_{i,j}$ is scheduled on processor p_x at frequency $f_{x,z}$ level

and $X_{i,j}^{x,z} = 0$ otherwise. If $X_{i,j}^{x,z} = 1$, then the start time ($ST_{i,j,x}$) and finish time ($FT_{i,j,x}$) of task $t_{i,j}$ on $f_{x,z} \in F_x$ can be computed using Eqs. (6) and (7) respectively, and the expected ready time of the processor is given by Eq. (8).

$$ST_{i,j,x} = \begin{cases} RT_x & Pred_{i,j} = \Phi \\ \max\left\{PRT_x, \max_{t_{i,k} \in Pred_{i,j} \&\& x \neq y}\left\{FT_{i,k,y} + e_{i,(k,j)}\right\}\right\}, & \text{otherwise} \end{cases}, \tag{6}$$

$$FT_{i,j,x} = ST_{i,j,x} + w_{i,j,x,z}, \tag{7}$$

$$RT_x = \begin{cases} FT_{i,j,x} & \text{if } X_{i,j}^{x,z} = 1 \\ FT_{l,m,x} & \text{otherwise} \end{cases}. \tag{8}$$

Here, $FT_{l,m,x}$ represents the last scheduled tasks ($t_{l,m}$) on processor p_x.

Due to the simultaneous execution of M precedence-constrained jobs on computation system C, the turnaround time (TAT_B) of batch B will be the maximum of finish times of all the tasks and can be computed as:

$$TAT_B = \max\left\{FT_{i,j,x} : \forall t_{i,j} \in B, Succ_{i,j} = \Phi\right\}. \tag{9}$$

3.4. Energy consumption model

The energy consumption of computational system C depends on all involved resources, namely the processors, cooling system, memory accesses, communication networks, and so on [2]. The processors consume the most significant portion of the total power consumption. The power consumption ($Pow_{x,z}$) of the processor consists of static power (P_{Static}) and dynamic power ($P_{Dynamic}$), and it is given at voltage $v_{x,z} \in V_x$ as [7,9–11]:

$$Pow_{x,z} = P_{Static} + P_{Dynamic}. \tag{10}$$

If processor p_x is idle, the processor consumes the static power (leakage power) constantly, irrespective of voltage and frequency, i.e. $Pow_x = P_{Static}$. With the busy mode, the dynamic power ($P_{Dynamic}$) depends on frequency $f_{x,z}$ and voltage $v_{x,z}$ given as:

$$P_{Dynamic} = \gamma_x \times V_{x,z}^2 \times f_{x,z}, \tag{11}$$

where γ_x is a constant representing the activity factor and the physical capacitance. The constant γ_x is a manufacturing constant, whereas frequency $f_{x,z}$ and $v_{x,z}$ can be decided at the compile or run time.

If $X_{i,j}^{x,z} = 1$, then task $t_{i,j}$ is executed on processor p_x at frequency $f_{x,z}$; the energy consumed by task $t_{i,j}$ can be computed using Eqs. (10) and (11) as:

$$E_{i,j,x,z} = Pow_{x,z} \times w_{i,j,x,z}. \tag{12}$$

The total energy consumption (EN_B) of batch B depends on the energy consumed by all tasks and idle slots created due to precedence constraints. Let N_x represent the total number of idle slots on processor p_x and $IS_{x,i}$ represent the length of the ith idle slot on processor p_x; the energy consumed by the idle slot (IS_i) is computed as:

$$EN_{x,i} = P_{Static} \times IS_{x,i}. \tag{13}$$

Therefore, the total energy consumption (EN_B) of batch B on computation system C is computed as the sum of the energy consumption during idle slots and the energy consumed by the allocated tasks, and it is given as follows:

$$EN_B = \sum_{i=1}^{M} \sum_{j=1}^{M_i} \sum_{x=1}^{K} \sum_{z=1}^{n_x} w_{i,j,x,z} \times X_{i,j,(x,z)} + \sum_{x=1}^{K} \sum_{i=1}^{N_x} EN_{x,i}. \tag{14}$$

3.5. Problem statement

The formulated scheduling problem can be represented in Graham et al.'s notation as $QK\,|t_{i,j} \sim pred, stoc|\min TAT_B \min EN_B$ [4]. The first field, QK, represents the computation system C of K heterogeneous parallel DVFS-enabled processors given in Section 3.1, whereas the second field represents the task $t_{i,j} \in B$ following the precedence constraints with stochastic time requirements as given in Section 3.2. The third field corresponds to the objective, i.e. to optimize the energy consumption and turnaround time of the batch B. Let S_B represent the schedule generated by the scheduling algorithm. The statement of the problem can then be written as:

$$\text{Minimize} \begin{cases} TAT_B \\ EN_B \end{cases}, \tag{15}$$

$$st. \begin{cases} \sum_{x=1}^{K} \sum_{z=1}^{n_x} X_{i,j}^{x,z} = 1, t_{i,j} \\ \sum\limits_{i=1}^{M} \sum_{j=1}^{M_i} \sum_{x=1}^{K} \sum_{z=1}^{n_x} X_{i,j}^{x,z} = \sum_{i=1}^{M} M_i \end{cases}. \tag{16}$$

4. The ESHEFT algorithm

The proposed ESHEFT algorithm executes the batch of precedence-constrained jobs (B) on the computation system (C), based on HEFT [6] and SHEFT [14]. However, HEFT and SHEFT are single-job algorithms with the aim of optimizing turnaround time, whereas the proposed algorithm ESHEFT is a batch- and energy-aware algorithm to schedule the batch of jobs to optimize both turnaround time (TAT_B) and energy consumption (EN_B). Furthermore, the ESHEFT algorithm employs a modified stochastic b-level (Eq. (5)) of the tasks in order to follow precedence constraints between tasks and idle slots to choose the execution frequency that reduces energy consumption without sacrificing turnaround time.

The pseudocode of our energy-aware stochastic heterogeneous earliest finish time (ESHEFT) algorithm is shown in Algorithm 1. The ESHEFT algorithm takes 2 inputs, batch B and computation system C, and returns schedule S_B with expected turnaround time (TAT_B) and energy consumption (EN_B) as output. As the algorithm proceeds, it computes the stochastic b-level ($sb_{i,j}$) and level $L_{i,j}$ for each task $t_{i,j} \in B$ using Eqs. (5) and (1), respectively. For the kth level's tasks, a queue Q_k is formed in decreasing order of stochastic b-level ($sb_{i,j}$), and the algorithm selects the processor and the execution frequency for each task level-wise. For the tasks of the kth level, it removes the first task (say $t_{i,j}$) from queue Q_k and computes the expected earliest starting time ($ST_{i,j,x}$) and expected earliest finish time ($FT_{i,j,x}$) with respect to each processor $p_x \in C$ at maximum frequency (f_x^{max}) using Eqs. (6) and (7) respectively. For task $t_{i,j}$, the algorithm selects the processor that offers the minimum expected earliest finish time ($FT_{i,j,x}$) and adds this task to the scheduling queue ($SCHQ_k$) of the kth level. The algorithm updates schedule S_B information as:

$$S_B\left(Proc_{i,j}\right) = p_x. \tag{17}$$

Following this, the algorithm selects the frequency for each task of the same level and the frequency is chosen using the idle slot before the task, resulting in no effect on turnaround time. For tasks in $SCHQ_k$, it removes the first task (say $t_{i,j}$) and determines its allocated processor p_x using $t_{i,j}$ and S_B. Following this, the algorithm determines the earliest start time ($EST_{i,j,x}$) of task $t_{i,j}$ on processor p_x as:

$$EST_{i,j,x} = \max\left\{FT_{l,m,z}, max_{t_{i,k}\in Pred_{i,j}\&\&z\neq y}\left\{FT_{i,k,y} + e_{i,(k,j)}\right\}\right\}. \tag{18}$$

Next, the algorithm computes the maximum possible total slot time that can be available to task $t_{i,j}$ without violating the precedence constraints as:

$$SlackT_{i,j,x} = FT_{i,j,x} - EST_{i,j,x} - w_x^{max}\left(t_{i,j}\right), if EST_{i,j,x} < ST_{i,j,x}. \tag{19}$$

The algorithm then determines the minimum possible continuous frequency ($f_{new}(t_{ij})$) of the processor p_x while meeting the deadlines of the expected earliest finish time ($FT_{i,j,x}$) as:

$$f_{new}(t_{ij}) = \frac{w_x^{max}(t_{i,j})}{w_x^{max}(t_{i,j}) + SlackT_{i,j,x}} \times f_x^{max}. \tag{20}$$

Algorithm 1. ESHEFT algorithm.
Algorithm 1: ESHEFT
Input: Batch B, Computation System C
Output: Schedule S_B with TAT_B, EN_B
Begin

1. Compute the level $L_{i,j}$ of each task $t_{i,j}\in B$ using Eq. (1)
2. Compute the stochastic b-level ($sb_{i,j}$) for each task $t_{i,j}\in B$ using Eq. (5)
3. Divide batch B into levels from 1 to L_B
4. For each level L_k, form queue Q_k by adding and sorting tasks in decreasing order of $sb_{i,j}$
5. For each level k = 1 to L_B do
6. //Processor Selection

   While there are tasks at level L_k

   a. Remove the task $t_{i,j}\in B$ from the list Q_k
   b. Compute expected earliest starting time ($ST_{i,j,x}$) and expected earliest finish time ($FT_{i,j,x}$) w.r.t. each $p_x\in C$ at f_x^{max} using Eqs. (6) and (7) respectively
   c. Assign processor $p_x\in C$ with minimum $FT_{i,j,x}$ to task $t_{i,j}$
   d. Add task $t_{i,j}$ to $SCHLIST_k$
   e. Update $S_B(Proc_{i,j})$ using Eq. (17)

   End While
7. //Frequency selection without sacrificing the turnaround time

   For each task $t_{i,j}\in SCHLIST_k$

   a. Determine the processor p_x=$S_B(Proc_{i,j})$ and $EST_{i,j,x}$

b. Compute the slack slot $SlackT_{i,j,x}$ using Eq. (19)

c. Compute the continuous frequency $f_{new}(t_{i,j})$ using (20)

d. Schedule task $t_{i,j}$ on processor p_x with discrete frequency $f_{x,z} \bar{>} f_{new}(t_{i,j})$

e. // $\bar{>}$ represents immediately greater than or equal fraction

f. Update $ST_{i,j,x}$ and $FT_{i,j,x}$ corresponding to $f_{x,z}$ using Eqs. (21) and (22)

g. Send precedence-constraints data to all successors $t_{i,k} \in Succ_{i,j}$

h. Update $S_B(freq_{i,j})$ using Eq. (23)

End For

8. End For
9. Get Schedule S_B
10. Compute TAT_B using schedule S_B and Eq. (9)
11. Compute EN_B using schedule S_B and Eq. (14)
12. Return S_B, TAT_B, and EN_B

End

Next, the ESHEFT algorithm selects the discrete frequency ($f_{x,z}$) immediately greater than the continuous frequency f_x^{Cont}; this frequency is assigned to task $t_{i,j}$ for execution. Due to the difference between discrete ($f_{x,z}$) and continuous ($f_{new}(t_{i,j})$) frequencies, the expected earliest starting time ($ST_{i,j,x}$) and expected earliest finish time($FT_{i,j,x}$) will be updated using Eqs. (21) and (22), respectively. The algorithm also updates schedule S_B information using Eq. (23).

$$ST_{i,j,x} = EST_{i,j,x}, \tag{21}$$

$$FT_{i,j,x} = ST_{i,j,x} + w_{x,z}(t_{i,j}), \tag{22}$$

$$S_B(freq_{i,j}) = f_{x,z}. \tag{23}$$

Following scheduling and frequency selection for all tasks, the algorithm computes the turnaround time (TAT_B) and energy consumption (EN_B) of the batch B using Eqs. (9) and (14), respectively.

4.1. Performance evaluation

To evaluate performance, a simulation program is developed using MATLAB with the Intel Core i5-3470 that realizes DVFS-enabled processors of 5 types, namely Intel Core 2 Duo with 4 frequencies, Intel Core 2 Extreme with 4 frequencies, AMD Sempron APUs with 3 frequencies, AMD Athlon APUs with 3 frequencies, and TI DSP with 2 frequencies. A simulation environment of 15 processors consisting of 3 processors of each type is created. To generate the batch B consisting of a random number of jobs (DAGs) and random processing and precedence-constrained tasks, a statistical prediction technique is used to get the normal distribution of processing and precedence-constraint times. The simulation program also takes 4 input parameters: 1) number of stochastic jobs in a batch B; 2) range of stochastic tasks in each job; 3) ranges of expectations and variances of processing times of tasks; 4) ranges of expectations and variances of precedence-constraint times [18]. The

performance of the ESHEFT is compared with ECS [9] and SHEFT [14] with 2 variations. Similar to ECS and SHEFT, ECS-1 and SHEFT-1 schedule jobs from batch B one-by-one randomly, whereas SHEFT-2 and ECS-2 convert the whole batch B into a single DAG by adding the pseudo-entry and -exit tasks with zero processing and communication times to schedule the tasks. Five different batches are created that consist of 10, 20, 30, 40, and 50 jobs with task ranges from 32 to 256, which results in the number of tasks [320, 2560], [640, 5120], [960, 7680], [1280, 10240], and [1600, 12800], respectively. The ranges of expectation and variance of processing times are [1, 3000] and [20, 1000] respectively, whereas ranges of expectation and variance of precedence-constrained times are [1, 500] and [20, 100], respectively. The results corresponding to 5 batches with turnaround time, energy consumption, and their standard deviations are shown in Figures 1–4, respectively.

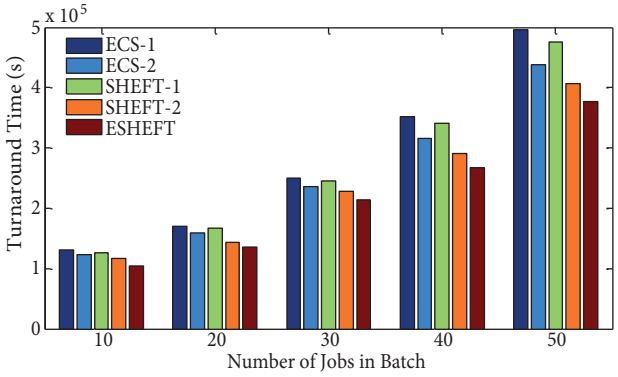

Figure 1. Turnaround time of different batches.

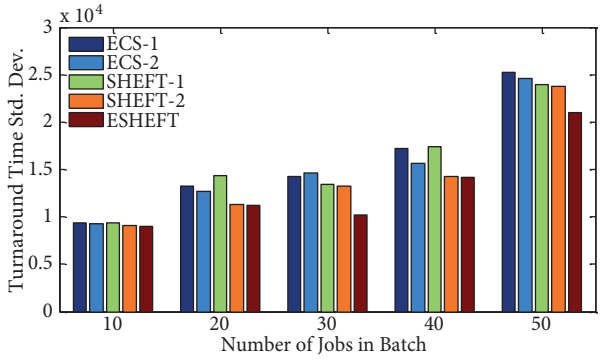

Figure 2. Turnaround standard deviation of different batches.

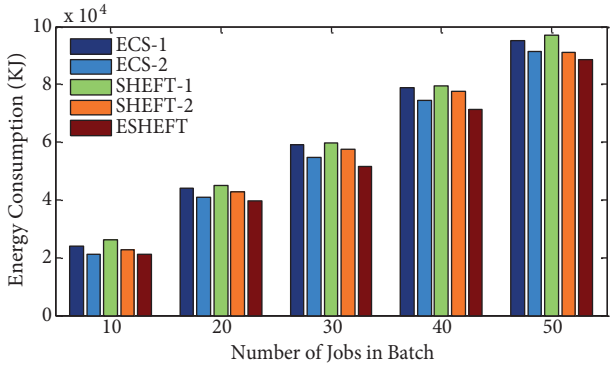

Figure 3. Energy consumption of different batches.

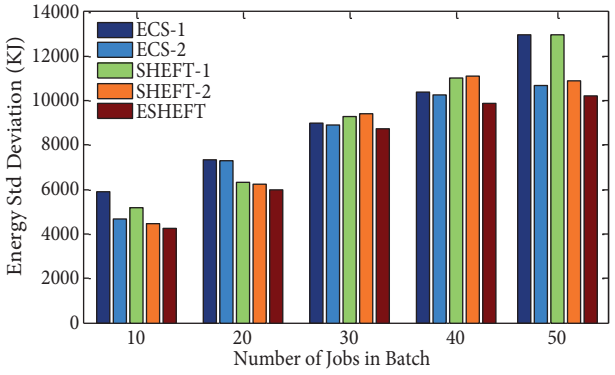

Figure 4. Energy consumption standard deviation of different batches.

Figure 1 represents the turnaround time of ECS-1, ECS-2, SHEFT-1, SHEFT-2, and ESHEFT on the computation systems of 15 processors for 5 batches. It is observed from Figure 1 that the order of turnaround time performance is ESHEFT, SHEFT-2, ECS-2, SHEFT-1, and ECS-1. It can also be seen from Figure 1 that the performance of ESHEFT increases with the size of the batch. For average performance of the 5 batches, the respective percentages for improvement of ESHEFT with respect to ECS-1, ECS-2, SHEFT-1, and SHEFT-2 are 12.5%, 11.5%, 12.2%, and 10.7% in terms of turnaround time. It is observed from Figure 2 that the ESHEFT algorithm offers more stable performance, as the standard deviation of the offered turnaround time is smaller in comparison to other algorithms. Corresponding to the turnaround time presented in Figure 1, the

energy consumption (kJ) of the same 5 batches is shown in Figure 3. It is observed from Figure 3 that the ESHEFT algorithm consumes minimal energy; the order of energy consumption is given as ESHEFT, ECS-2, SHEFT-2, ECS-1, and SHEFT-1. The energy-aware algorithms ECS-1 and ECS-2 both take into consideration the execution time and energy consumption of tasks; hence, both offer lower energy consumption in comparison to SHEFT-1 and SHEFT-2, respectively. For the average energy consumption of 5 batches, the percentages for energy consumption improvement of ESHEFT with respect to ECS-1, ECS-2, SHEFT-1, and SHEFT-2 are 11.1%, 10.4%, 11.4%, and 10.7%, respectively. It is also observed from Figure 4 that the ESHEFT algorithm offers the minimum energy standard deviation, which results in more stable performance in comparison to its peers.

Both ECS-1 and SHEFT-1 algorithms schedule jobs one-by-one randomly and only exploit the parallelism between the tasks of a job rather than making use of parallelism between different jobs, resulting in a greater number of idle slots on the schedule. Therefore, ECS-1 and SHEFT-1 offer higher turnaround time in comparison to the ECS-2, SHEFT-2, and ESHEFT algorithms. More idle slots consume more energy towards the total energy consumption, which causes ECS-1 and SHEFT-1 to consume more energy in comparison to the ECS-2, SHEFT-2, and ESHEFT algorithms. ECS-2 and SHEFT-2 form a single DAG, adding some precedence constraints between jobs as well as tasks, which results in less parallelism and more delay in the execution time of tasks; hence, these algorithms offer greater turnaround time in comparison to ESHEFT. Since the proposed algorithm ESHEFT explores and exploits the parallelism between and within the jobs, the ESHEFT algorithm makes full use of resources, which results in less turnaround time. Additionally, SHEFT-2 and ECS-2 consume more energy in idle slots in comparison to the ESHEFT algorithm.

The main reason behind the good performance of the ESHEFT algorithm is that it combines all of the jobs to exploit the parallelism between and within the jobs of the batch and generates a continuous frequency based on the available idle slot that helps to choose the discrete frequency of the processor. It can be concluded that combining the jobs into batches and scheduling using ESHEFT has a major impact on performance in comparison to peers for the $QK\,|t_{i,j}\sim pred, stoc|\min TAT_B \min EN_B$ problem.

5. Conclusion

An energy-aware stochastic ESHEFT algorithm is proposed to schedule the batch of precedence-constrained jobs (DAGs) on DVFS-enabled processors, incorporating the slack time of tasks to minimize energy consumption without sacrificing turnaround time. Using a randomly generated batch of precedence-constrained jobs, the simulation results demonstrate the superiority of the ESHEFT algorithm over SHEFT and ECS in terms of turnaround time, energy consumption, and their standard deviations. From the experimental study, it is found that scheduling the precedence-constrained stochastic jobs into batches is preferable to scheduling a single job in terms of turnaround time and energy consumption, as it results in higher system utilization, lower monetary costs, and lower negative environmental effects. Additionally, the performance of the ESHEFT algorithm suggests its possible use for scheduling in general multiprocessor, multicore processors, and large-scale parallel computing systems.

Symbol	Explanation
C	Computation system C with K different per-chip DVFS-enabled processor p_x.
V_x/F_x	The set of n_x discrete voltage ($v_{x,z}$)/frequency ($f_{x,z}$) levels for processor $p_x \in C$.
B	Batch of M jobs and each job $J_i \in B$ consisting of multiple dependent tasks.
T_i/E_i	T_i consists of M_i dependent tasks $t_{i,j}$ and E_i consists of precedence-constrained edge $e_{i,(j,k)}$ between tasks.

$Succ_{i,j}$	The set of all successor tasks of $t_{i,j}$.
$Pred_{i,j}$	The set of predecessors of task $t_{i,j}$
$L_{i,j}/L_B$	The level of tasks $t_{i,j}$/batch B.
$[w_{i,j,x,z}]$	A matrix of order$M \times M_{\max} \times K \times N_C$, where $w_{i,j,x,z}$ gives processing time of task $t_{i,j} \in T_i$ w.r.t. frequency level $f_{x,z} \in F_x$.
$[w_{i,(j,k)}]$	A matrix of order $M \times M_{\max} \times M_{\max}$, $w_{i,(j,k)}$ gives approximate precedence-constraint time of edge $e_{i,(j,k)}$ between tasks $t_{i,j}$ and $t_{i,k}$.
$sb_{i,j}$	The stochastic b-level ($sb_{i,j}$) of task $t_{i,j}$ w.r.t. to DVFS-enabled processor.
$ST_{i,j,x}$	Expected start time of task $t_{i,j}$ scheduled on processor p_x.
$FT_{i,j,x}$	Expected finish time of task $t_{i,j}$ scheduled on processor p_x.
RT_x	Expected ready time of processor p_x.
TAT_B	The turnaround time of batch B.
$Pow_{x,z}$	The power consumption of processor p_x consists of static power (P_{Static}) and dynamic power ($P_{Dynamic}$) at voltage $v_{x,z}$.
EN_B	The total energy consumption of batch B.

References

[1] TOP 500 Supercomputers, TIANHE-2 November 2014 List. URL: http://www.top500.org/lists/2014/11/. Accessed on 8 May 2015.

[2] Venkatachalam V, Franz M. Power reduction techniques for microprocessor systems. ACM Comput Surv 2005; 37: 195-237.

[3] Garey MR, Johnson DS. Computers and Intractability: A Guide to the Theory of NP-Completeness. New York, NY, USA: W. H. Freeman & Co., 1990.

[4] Leung JYT. Handbook of Scheduling: Algorithms, Models, and Performance Analysis. New York, NY, USA: Chapman & Hall/CRC, 2004.

[5] Kwok YK, Ahmad I. Static scheduling algorithms for allocating directed task graphs to multiprocessors. ACM Comput Surv 1999; 31: 406-471.

[6] Topcuoglu H, Hariri S, Wu M. Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE T Parall Distr 2002; 13: 260-274.

[7] Zhu D, Melhem R, Childers B. Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems. IEEE T Parall Distr 2003; 14: 686-700.

[8] Zhang S, Chatha KS. Approximation algorithm for the temperature-aware scheduling problem. In: IEEE/ACM 2007 Computer-Aided Design Conference; 4–8 November 2007; San Jose, CA. Piscataway, NJ, USA: IEEE/ACM. pp. 281-288.

[9] Lee YC, Zomaya A. Energy conscious scheduling for distributed computing systems under different operating conditions. IEEE T Parall Distr 2011; 22: 1374-1381.

[10] Li K. Scheduling precedence constrained tasks with reduced processor energy on multiprocessor computers. IEEE T Comput 2012; 61: 1668-1681.

[11] Zhuravlev S, Saez JC, Blagodurov S, Fedorova A, Prieto M. Survey of energy-cognizant scheduling techniques. IEEE T Parall Distr 2013; 24: 1447-1464.

[12] Chekuri C, Motwani R, Natarajan B, Stein C. Approximation techniques for average completion time scheduling. SIAM J Comput 2001; 31: 146-166.

[13] Skutella M, Uetz M. Stochastic machine scheduling with precedence constraints. SIAM J Comput 2005; 34: 788-802.

[14] Tang X, Li K, Liao G, Fang K, Wu F. A stochastic scheduling algorithm for precedence constrained tasks on grid. Future Gener Comp Sys 2011; 27: 1083-1091.

[15] Li K, Tang X, Veeravalli B, Li K. Scheduling precedence constrained stochastic tasks on heterogeneous cluster systems. IEEE T Comput 2015; 64: 191-204.

[16] Moring RH, Schulz AS, Uetz M. Approximation in stochastic scheduling: the power of LP-based priority policies. J ACM 1999; 46: 924-942.

[17] Scharbrodt M, Schickingera T, Steger A. A new average case analysis for completion time scheduling. J ACM 2006; 53: 121-146.

[18] Kasahara H. STG: Standard Task Graph Set. Kasahara Laboratory, Waseda University, Tokyo, Japan. URL: http://www.kasahara.elec.waseda.ac.jp/schedule/. Accessed on 2 May 2015.