2-1-2013

3E: Energy-Efficient Elastic Scheduling for Independent Tasks in Heterogeneous Computing Systems

Xiaomin Zhu
National University of Defense Technology

Rong Ge
Marquette University, rong.ge@marquette.edu

Jinguang Sun
Liaoning Technical University

Chuan He
National University of Defense Technology

Accepted version. *Journal of Systems and Software*, Vol. 86, No. 2 (February 2013): 302-314. DOI.

Journal of Systems and Software, Vol. 86, No. 2 (February 2013): 302-314. DOI.

3E: Energy-efficient elastic scheduling for independent tasks in heterogeneous computing systems

Xiaomin Zhu
Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410073, PR China
Rong Ge
Department of Mathematics, Statistics, and Computer Science, Marquette University, Milwaukee, WI
Jinguang Sun
School of Electronic and Information Engineering, Liaoning Technical University, Huludao 125105, PR China
Chuan He
Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410073, PR China

Abstract

Reducing energy consumption is a major design constraint for modern heterogeneous computing systems to minimize electricity cost, improve system reliability and protect environment. Conventional energy-efficient scheduling strategies developed on these systems do not sufficiently exploit the system elasticity and adaptability for maximum energy savings, and do not simultaneously take account of user expected finish time. In this paper, we develop a novel scheduling strategy named energy-efficient elastic (3E) scheduling for aperiodic, independent and non-real-time tasks with user expected finish times on DVFS-enabled heterogeneous computing systems. The 3E strategy adjusts processors' supply voltages and frequencies according to the system workload, and makes trade-offs between energy consumption and user expected finish times. Compared with other energy-efficient strategies, 3E significantly improves the scheduling quality and effectively enhances the system elasticity.

Keywords

Heterogeneous computing system, Scheduling, Energy-efficient, Elastic, DVFS

1. Introduction

Heterogeneous computing systems constructed by connecting various machines with different capabilities have been wildly employed for compute-intensive and data-intensive applications in scientific research and commercial industries (Goller and Leberl, 2009, Zheng et al., 2006, Donoho, 2004). The employment is mostly attributed to the high processing capability and low cost of commodity-off-the-shelf hardware components including processors, memory, networks, and storage disks (Xie and Qin, 2008).

Large-scale computing systems consume tremendous amounts of energy, which in turn causes high energy bills in data centers, raises environmental concerns, and increases system failures. The monetary cost for energy is very high. For instance, the total energy cost of a single 200 W server (e.g., IBM 1U*300) is $180/year (Bianchini and Rajamony, 2004). A computing system with thousands of compute nodes or more incurs large energy bills. The high energy consumption also has negative environmental impacts. It is estimated that 1 kWh electricity power requires 0.4 kg coal and 4 L water and produces 0.272 kg solid powder, 0.997 kg CO_2, and 0.03 kg SO_2. Last, high energy consumption results in high temperature that greatly affects the system reliability. According to the Arrhenius equation, the failure rate of an electronic component doubles for every 10 °C increased (Feng, 2003). In order to maintain an appropriate operating temperature in data centers, extra energy will be consumed by the cooling devices and facility.

Dynamic voltage and frequency scaling (DVFS) is an energy saving technology that are enabled on most contemporary processors (http, in pressa, http, in pressb). With DVFS, a processor can operate at multiple voltages where each corresponds to a specific clock frequency and processing speeds. Because the energy consumption of a processor is proportional to voltage squared (Chen et al., 2006), processor's energy consumption can be significantly reduced by lowering CPU voltage and processing speed.

The motivations of this paper derive from the following three considerations:

- *Many applications running on heterogeneous computing systems consist of independent tasks without dependencies.* For example, the tasks submitted to a supercomputer center by different users are independent (Braun et al., 2001); the partitioned data blocks from signal data in a software radio system can be considered as independent tasks without precedence relationship (Zhu and Lu, 2008); a parameter-sweep application consists of a set of independent coarse-grained tasks, and such applications can be seen in diverse areas such as bioinformatics, operations research, data mining, business model simulation, massive searches, Monte Carlo simulations, network simulation, electronic CAD, ecological modeling, fractals calculations, and image manipulation (Fujimoto and Hagihara, 2006).•
- *There exist many applications for which users do not have strict requirement in finish time.* For these applications, even though the finish time of a task is a bit later or earlier than its user's expectations, the task execution is still useful. Take earth observation satellite as an example. The satellite data are

firstly processed at a ground data center in one task, then the generated data are analyzed in the next task for a new electronic reconnaissance. If the first task finishes later than the start time of the second task, the generated data are still useful for another new task (Wang et al., 2010). In the traditional real-time computing systems, a task would be rejected if it is unable to finish within its deadline. For non-real-time applications, only the computing throughput matters while the finish times of single tasks can be ignored. Thereby, in order to provide high-quality services for users, each user's expectation (e.g., expected finish time) should be adequately considered.•

- *Emergency tasks and non-emergency tasks show distinct characteristics.* The applications that consist of independent tasks (Xie and Qin, 2008, Doˇgan and Özgüner, 2006, Mehta et al., 2007, Kang and He, 2009) in data centers can be roughly classified into two groups. One group includes applications for which high performance and short execution time are critical. One example is the processing of image data obtained by earth observation satellites in emergency such as earthquake, tsunami or military operations (Wang et al., 2007). In these scenarios, quick response is paramount while energy savings with possible performance impact are out of the questions. The other group includes applications that are not in emergency in nature. For these applications, we can exploit system elasticity, i.e., flexibility and adaptivity with the variety of system workload, in scheduling algorithms to reduce energy use with no or little impact on meeting user expectations.

Scheduling is an effective approach to achieve high performance and energy efficiency for applications running on heterogeneous computing systems. However, to the best of our knowledge, most existing energy-efficient scheduling algorithms do not address both performance and energy cost for non-real-time applications in heterogeneous computing systems. Motivated by the above arguments, in this work, we attempt to incorporate the system elasticity and user expectations into energy-efficient scheduling strategies, and to design and implement a novel energy-efficient elastic scheduling strategy for non-real-time tasks on DVFS-enabled heterogeneous computing systems. Specifically, our approach firstly strives to satisfy the user expectations by adjusting the execution voltages of queued tasks and new tasks, and then reduces the system energy consumption as much as possible.

Contributions: The main contributions of this paper are:

- We construct an energy consumption model that effectively takes advantage of system elasticity and considers different user expectations in terms of expected finish time.
- On the basis of the novel energy consumption model, we develop an energy-efficient elastic (3E for short) scheduling strategy for independent tasks in heterogeneous computing systems to make trade-offs between energy saving and user expectation according to the system workload.
- We demonstrate that, by considering heterogeneous features of multiprocessor computing systems, we can design an energy-efficient elastic scheduling strategy that significantly improves the scheduling quality of conventional scheduling strategies for heterogeneous computing systems.

The rest of this paper is organized as follows. In Section 2, we discuss the related work in literature. Section 3 presents the system model, task model, energy consumption model. In Section 4, we describe the 3E strategy and its main principles. Section 5 presents the experimental results and performance evaluation. Section 6 concludes the paper with a summary and future directions.

2. Related work

Over the last decades, scheduling in distributed computing context has been intensively investigated. As the optimal scheduling solutions are normally NP-complete (Coffman, 1976), near-optimal solutions using heuristic techniques are adopted as practical alternatives (Xie and Qin, 2008, Braun et al., 2001, Kim et al., 2008, Karatza, 2009).

Scheduling can be generally classified into static scheduling and dynamic scheduling according to its design time. The static scheduling makes scheduling decisions in an off-line planning phase, and is usually used

to schedule periodic tasks (e.g., Aydin et al., 2004, Mishra et al., 2003, Yu and Prasanna, 2002, Zhu et al., 2003). The dynamic scheduling is performed in an on-line fashion when tasks arrive at unpredictable intervals, and is usually applied to aperiodic tasks and the system workload that is not known a priori (e.g., Ge et al., 2005, Hu et al., 2008, Zong et al., 2011, Hamano et al., 2009, Zikos and Karatza, 2011, Yan et al., 2005, Zhu et al., 2011, Zhu and Lu, 2009).

Aydin et al. proposed a static solution for periodic tasks to compute the optimal speed at the task level based on the worst-case workload for each arrival (Aydin et al., 2004). Mishra et al. proposed a static power management scheme that used the static slack based on the degree of parallelism in a given static schedule generated from any list scheduling heuristic algorithm (Mishra et al., 2003). Yu et al. investigated an off-line power-aware allocation policy that was firstly formulated as an extended generalized assignment problem, and was solved by an extension of a linearization heuristic for a set of independent tasks in a real-time system consisting of heterogeneous DVS-enabled processing elements (Yu and Prasanna, 2002). Zhu et al. introduced the concept of slack sharing on multiprocessor systems to reduce energy consumption and proposed two novel power-aware scheduling algorithmsbased on slack sharing for task sets with and without precedence constraints executing on multiprocessor systems (Zhu et al., 2003). Additionally, the study in Zhu et al. (2003) assumed homogeneous processors and frame-based tasks. Tavares et al. proposed a pre-runtime scheduling method that considered the DVS technique to reduce energy consumption and took the inter-task relations and runtime overhead into account. In addition, the time Petri nets was employed as a mathematical basis for precise pre-runtime schedule generation (Tavares et al., 2008). Although these scheduling schemes are capable of achieving high scheduling quality in terms of energy saving, they belong to static scheduling and are unable to deal with dynamic environment where the arrival time of a task is not known.

There is a large body of work in designing dynamic energy-efficient scheduling algorithms for distributed computing systems. Ge *et al.* investigated distributed performance-directed DVS scheduling strategies that could produce significant energy savings without increasing execution time by varying scheduling granularity (Ge et al., 2005). Nélis et al. proposed two power-aware scheduling algorithms, i.e., an off-line algorithm $EDF^{(k)}$ and an on-line algorithm MOTE that both addressed sporadic constrained-deadline real-time systems to reduce energy consumption (Nélis et al., 2008). Hu et al. described an approach to reduce energy consumption by employing the live migration of virtual machines to transfer load among the identical nodes on a multilayer ring-based overlay (Hu et al., 2008). Laszewski et al. focused on scheduling virtual machines in a compute cluster to reduce power consumption by dynamic voltage frequency scaling, i.e., DVFS technique, and presented a scheduling algorithm to allocate virtual machines in a DVFS-enabled cluster (Laszewski et al., 2009). Liu et al. developed a cluster-based energy-performance balanced task duplication based clustering scheduling algorithm EPBTDCS that saved energy by reducing communication energy consumption while assigning parallel tasks to compute nodes (Liu et al., 2010). Zong et al. proposed two energy-aware duplication-based scheduling algorithms, namely, energy-aware duplication algorithm EAD and performance-energy balanced duplication algorithm PEBD with the objective of improving both performance and energy efficiency (Zong et al., 2011). These schemes were designed for homogeneous computing environments, not suitable for heterogeneous computing systems with diverse processing elements because a processor with higher processing speed may consume less energy.

There also exist many previous studies investigating power-aware techniques to reduce energy consumption on heterogeneous computing systems. For example, Hamano et al. proposed a scheme to improve energy efficiency by adjusting the schedule based on the acceleration factor for heterogeneous accelerated clusters (Hamano et al., 2009). Zong et al. studied a scheduling strategy which could conserve energy by judiciously shrinking communication energy cost when allocating tasks to heterogeneous compute nodes in a cluster (Zong et al., 2007). Shekar and Izadi addressed the problem of scheduling tasks in a heterogeneous environment and proposed an algorithm called EDLS that favored low-energy consuming processors by introducing a cost factor affecting scheduling decisions (Shekar and Izadi, 2010). However, these approaches did not consider timing requirement.

Research in energy-efficient scheduling for real-time tasks in heterogeneous computing systems has been active. For instance, Zikos and Karatza examined three local resource allocation policies for compute-intensive jobs with unknown service times based on the shortest queue in a heterogeneous cluster (Zikos and Karatza, 2011). Yan et al. developed a scheduling algorithm combining DVS and adaptive body biasing to optimize dynamic power and leakage power consumption jointly for heterogeneous distributed real-time embedded systems (Yan et al., 2005). Chen et al. presented an energy-efficient real-time task scheduling policy that aimed at the provision of approximated solutions with worst-case guarantees (Chen et al., 2007). Hung et al. developed energy-efficient real-time algorithms for the systems equipped with a DVS-enabled processor and a non-DVS processing element with different energy consumption models (Hung et al., 2006). Terzopoulos and Karatza investigated the resource scheduling policies based on energy consumption criteria for a real-time grid system with power-saving capable processors (Terzopoulos and Karatza, 2011). In these real-time scheduling algorithms, if one task cannot be finished within its deadline, it will be rejected. As a result, for those tasks with expected finish times but no deadlines, they are not feasible solutions.

In our earlier work (Ge et al., 2005, Zhu et al., 2011, Zhu et al., 2011, Zhu and Lu, 2009, Zhu and Lu, 2009), we have investigated some scheduling strategies on heterogeneous multiprocessor systems. However, such work did not consider the elastic energy conservation issue and the important case that users have expected finish time but not strict deadlines for tasks. In this paper, we focus on the problem of elastic scheduling for independent, aperiodic tasks with users' expected finish times in heterogeneous computing systems. Also, we propose a novel scheduling strategy 3E to make trade-offs between the energy conservation and users' expected finish times based on the system workload.

3. Mathematical models

In this section, we describe mathematical models used to represent heterogeneous computing systems, independent tasks, and energy consumptions. For future reference, we sum up the main notation used throughout this paper in Table 1.

Table 1. Definitions of notation.

Notation	**Definition**
p_j	The jth compute node in the node set $P = \{p_j, j = 1 \ldots \lvert P \rvert\}$
t_i	The ith task in the task set $T = \{t_i, i = 1 \ldots \lvert T \rvert\}$
a_i	The arrival time of t_i
l_i	The length/size of t_i
eft_i	The expected finish time of t_i
at_{ij}	The available time of t_i on p_j
st_{ij}	The start time of t_i on p_j
ft_{ij}	The finish time of t_i on p_j
et_{ij}	The execution time of t_i on p_j
tt_{ij}	The transmission time of t_i from the scheduler to p_j
x_{ij}	x_{ij} is "1" if t_i is assigned to p_j; otherwise, x_{ij} is "0"
o_{ij}	The execution order of t_i on p_j
w_{ij}	wij is "1" if t_i is waiting in the local queue of p_j, and is "0" else
V_j	The voltage set of p_j
V_{jk}	The kth voltage level of p_j
vij	The selected voltage of t_i on n_j, $v_{ij} \in V_j$
$ecr(v_{ij})$	The energy consumption rate with supply voltage vij
ec_{ij}^{node}	The energy consumed by t_i on p_j
$s(v_{ij})$	The processing speed of p_j when using v_{ij}
τ_j^{idle}	The idle time of p_j

ec_{ij}^{tran}	The transmission energy consumption of t_i from scheduler to p_j
ecr_j^{tran}	The transmission energy consumption rate of the link from scheduler to p_j
tr_j	The transmission rate of the link between the scheduler and p_j
tst_{ij}	The transmission start time of task t_i to p_j
rt_j	The remaining execution time of a running task on p_j
$ecr_j^{link-idle}$	The energy consumption rate of a link between the scheduler and p_j sitting idle
$\tau_j^{link-idle}$	The idle time of link between the scheduler and p_j

3.1. Heterogeneous computing model

A heterogeneous computing system in this study is characterized by a set $P = \{p_j, j = 1 \ldots |P|\}$ of compute nodes that have different processing capabilities. In addition, the interconnection is heterogeneous, i.e., the communication cost from the scheduler to the compute nodes varies. The energy-efficientheterogeneous computing model is depicted in Fig. 1.

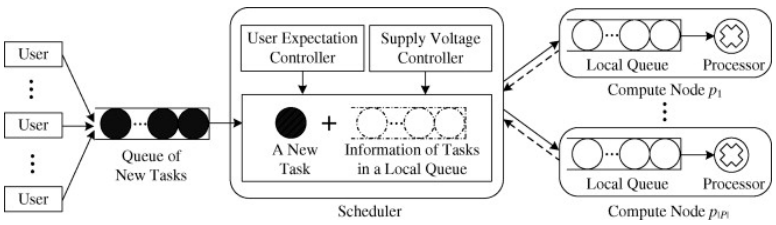

Fig. 1. Energy-efficient heterogeneous computing model.

In Fig. 1, the scheduler includes a user expectation controller and a supply voltage controller. They work together to firstly meet user expected finish times and then explore opportunities for energy savings. There is also a task queue in the scheduler whereas the tasks scheduling decision has been made but yet modified before dispatching to the corresponding compute nodes. Each compute node has a local queue where tasks wait for execution.

When a new task arrives, the scheduler takes two steps to make a scheduling decision. First, it retrieves the scheduling information of each task in the queue including supply voltage, execution time, and execution order. Second, it determines the compute node that is able to meet the user's expectation with least energy consumption. The new task will be enqueued in the scheduler and dequeued to the selected compute note. The system elasticity lies in that the scheduler can adjust the supply voltages for the tasks in its local queue according to the system workload. Previous studies do not consider this elasticity and assume the scheduling decisions that have been made are not subject to modification.

3.2. Task model

The task set is denoted by $T = \{t_i, i = 1 \ldots |T|\}$, where the tasks are independent of each other. A task arrives dynamically, and is indivisible and cannot be distributed to multiple compute nodes for concurrent execution. In addition, there is no communication among tasks. A task is represented by $t_i = \{a_i, l_i, eft_i\}$ where a_i is the arrival time, l_i is the task length/size, and eft_i is the expected finish time. In this work, we neglect the cost of collecting the information about the queue items as the information size is small. The following denotations are used to determine the scheduling of task set T on the compute node set P in this paper.

$AT = (at_{ij})_{|T|\times|P|}$: the available time matrix for the task set T on the compute grid P. Element at_{ij}represents the available time for task t_i on compute node p_j.
$ST = (st_{ij})_{|T|\times|P|}$: the start time matrix for the task set T on the compute grid P. Element st_{ij} represents the start time of task t_i on compute node p_j.
$FT = (ft_{ij})_{|T|\times|P|}$: the finish time matrix for the task set T on the compute grid P. Element ft_{ij} represents the finish time of task t_i on compute node p_j.

$ET = (et_{ij})_{|T|\times|P|}$: the execution time matrix. Element et_{ij} represents the execution time of task t_i on compute node p_j. The values in a row are different because of the heterogeneity in the processing capabilities of compute nodes.
$TT = (tt_{ij})_{|T|\times|P|}$: the transmission time matrix. Element tt_{ij} represents the transmission time of task $t_i$from the scheduler to compute node p_j.
$X = (x_{ij})_{|T|\times|P|}$: the task allocation matrix. Element x_{ij} is "1" if task t_i is allocated to compute node $p_j$and is "0", otherwise.
$O = (o_{ij})_{|T|\times|P|}$: the execution order matrix. Element o_{ij} represents the execution order of task t_i on compute node p_j.
$W = (w_{ij})_{|T|\times|P|}$: the task waiting matrix. Element wij is "1" if task t_i is waiting in the local queue of compute node p_j. Otherwise, wij=0.

3.3. Models of user expectation and energy consumption

The compute nodes in the system are DVFS enabled and the processors are able to operate with multiple pairs of voltage and frequency. Let ec_{ij} be the energy consumption of task t_i running on the compute node p_j. ec_{ij} is the product of the energy consumption rate ecr_j of compute node p_j and execution time of task t_i. The energy consumption rate ecr_j varies with the supply voltage (Xie and Qin, 2008). Commonly, the energy consumption rate is also called power of a compute node (Zong et al., 2011). Given k supply voltages for compute node p_j, the voltage set is denoted by $V_j = \{V_{j1}, V_{j2}, \ldots, V_{jk}\}$. Without loss of generality, we assume that $V_{j1} < V_{j2} < \ldots < V_{jk}$. We use vij∈Vj to denote the scheduled supply voltage when task t_i runs on compute node p_j. Thus, the energy consumption rate with supply voltage vij can be written as ecr(vij).

The energy consumed by task t_i on compute node p_j is (Xie and Qin, 2008, Zong et al., 2011):

(1) $$ec_{ij}^{node} = ecr(v_{ij}) \cdot et_{ij}.$$

Thereby, given a task set T, a compute node set P, an allocation matrix X, a voltage set V, and an execution time matrix ET, the total energy required to execute all tasks is:

(2) $$ec^{node-active}(T,P,X,V,ET) = \sum_{j=1}^{|P|} \sum_{i=1}^{|T|} x_{ij} \cdot ec_{ij}^{node} = \sum_{j=1}^{|P|} \sum_{i=1}^{|T|} x_{ij} \cdot ecr(v_{ij}) \cdot et_{ij},$$

where et_{ij} is the execution time of task t_i on compute node p_j. It can be calculated as Eq. (3):

(3) $$et_{ij} = \frac{l_i}{s(v_{ij})},$$

where $s(v_{ij})$ is the processing speed of compute node p_j when using the supply voltage vij to deal with task t_i.

Eq. (2) does not include the energy consumption when the compute nodes are idle. In this study, we set the supply voltage to the lowest level when compute nodes are idle. Thus, the energy consumption rate of compute node p_j at idle is $ecr(V_{j1})$. The energy consumed by compute nodes at idle is:

(4) $$ec^{node-idle}(T,P,X,V,ET) = \sum_{j=1}^{|P|} ecr(V_{j1}) \cdot \tau_j^{idle} = \sum_{j=1}^{|P|} ecr(V_{j1}) \cdot \left(\max_{i=1}^{|T|} \{ft_{ij}\} - \sum_{i=1}^{|T|} x_{ij} \cdot et_{ij} \right),$$

where τ_j^{idle} is the idle time of compute node p_j. It equals the schedule length of p_j minus the total execution time of all tasks assigned to p_j. $max_{i=1}^{|T|}\{ft_{ij}\}$ is the finish time of the last executed task on p_j, i.e., the compute node p_j's schedule length.

Combining both dynamic energy and energy during idle time, the node energy consumption of the heterogeneous computing system is derived from Eq. (2) and Eq. (4) as:

$$(5)\ pec(T,P,X,V,ET) = ec^{node-active}(T,P,X,V,ET) + ec^{node-idle}(T,P,X,V,ET) = \sum_{j=1}^{|P|}\sum_{i=1}^{|T|} x_{ij}\cdot ecr(v_{ij})\cdot et_{ij} + \sum_{j=1}^{|P|} ecr(V_{j1})\cdot\left(\max_{i=1}^{|T|}\{ft_{ij}\} - \sum_{i=1}^{|T|} x_{ij}\cdot et_{ij}\right).$$

In this study, we also consider the transmission energy consumption of tasks from scheduler to compute nodes. Due to the heterogeneity of networks, we let ecr_j^{tran} denote the transmission energy consumption rate of the link from scheduler to p_j. Hence, the transmission energy consumption of t_i from scheduler to p_j is measured as below:

$$(6)\ ec_{ij}^{tran} = ecr_j^{tran}\cdot tt_{ij} = ecr_j^{tran}\cdot\frac{l_i}{tr_j},$$

where tr_j is the transmission rate of the link between the scheduler and compute node p_j.

Given a task set T, a compute node set P, an allocation matrix X, and a transmission time matrix TT, the transmission energy consumed by transmitting all tasks is:

$$(7)\ ec^{link-tran}(T,P,X,TT) = \sum_{j=1}^{|P|}\sum_{i=1}^{|T|} x_{ij}\cdot ec_{ij}^{tran} = \sum_{j=1}^{|P|}\sum_{i=1}^{|T|} x_{ij}\cdot ecr_j^{tran}\cdot\frac{l_i}{tr_j}.$$

Additionally, the energy consumption when a link is idle (i.e., no message needs to be transmitted in a link) is considered in our energy model. The energy consumption rate of a link between the scheduler and compute node p_j sitting idle is denoted by $ecr_j^{link-idle}$ and we obtain the energy consumed by the link when it is inactive as follows:

$$(8)\ ec^{link-idle}(T,P,X,AT,TT) = \sum_{j=1}^{|P|} ecr_j^{link-idle}\cdot\tau_j^{link-idle} = \sum_{j=1}^{|P|}\left(ecr_j^{link-idle}\cdot\left(\max_{i=1}^{|T|}\{at_{ij}\} - \sum_{i=1}^{|T|} x_{ij}\cdot tt_{ij}\right)\right).$$

where $\tau_j^{link-idle}$ is the idle time of link between the scheduler and compute node p_j.

The link energy consumption can be written as:

$$(9)\ lec(T,P,X,AT,TT) = ec^{link-tran}(T,P,X,TT) + ec^{link-idle}(T,P,X,AT,TT) = \sum_{j=1}^{|P|}\sum_{i=1}^{|T|} x_{ij}\cdot ecr_j^{tran}\cdot\frac{l_i}{tr_j} + \sum_{j=1}^{|P|}\left(ecr_j^{link-idle}\cdot\left(\max_{i=1}^{|T|}\{at_{ij}\} - \sum_{i=1}^{|T|} x_{ij}\cdot tt_{ij}\right)\right).$$

Finally, the total energy consumption can be derived from Eqs. (5) and (9) as:

(10) $tec(T,P,X,V,ET,AT,TT) = pec(T,P,X,V,ET) + lec(T,P,X,AT,TT).$

Given a set *T* of tasks, the assignment should maximize the count of tasks for which the user expectation is met. Thus, the optimization problem can be formulated as:

$$\text{(11)}\quad \begin{array}{ll} \text{maximize} & \sum_{j=1}^{|P|} \sum_{i=1}^{|T|} x_{ij} \\ \text{subjectto} & \begin{cases} ft_{ij} \le eft_i \\ \sum_{j=1}^{|P|} x_{ij} = 1 \\ 1 \le i \le |T| \\ 1 \le j \le |P| \end{cases} \end{array}$$

We would like to emphasize that we weight more on the user expectation than the energy savings in our scheduling. In other words, the energy savings will be largely built upon the solution space from Eq. (11). Thereby, the total energy consumption value needs to be minimized, i.e.,

$$\text{(12)}\quad \begin{array}{ll} \text{minimize} & tec(T,N,X,V,ET,AT,TT) \\ \text{subjectto} & \begin{cases} V_{j1} \le v_{ij} \le V_{jk} \\ ft_{ij} \le eft_i \\ \sum_{j=1}^{|P|} x_{ij} = 1 \\ 1 \le i \le |T| \\ 1 \le j \le |P| \end{cases} \end{array}$$

Energy conservation and user expectation (expected finish time) are two conflicting objectives on a heterogeneous computing system. Minimizing the energy use by a compute node under heavy load could result in a late finish time for current tasks and unmet user expectations for subsequent tasks. Our energy-efficient elastic (3E for short) scheduling strategy makes trade-offs between Eqs. (11) and (12) according to the system workload. When the system is under heavy load, 3E favors user expectations. When the system is under light load, it favors energy savings by lowering supply voltages within guaranteeing users' expectations.

4. The energy-efficient elastic 3E scheduling strategy

In this section, we present 3E strategy for independent and aperiodic tasks with user expected finish time in a heterogeneous computing system. Firstly, we introduce some rules to facilitate the presentation of our scheduling strategy.

Property 1

A task that cannot be finished before its user expected finish time is still assigned to a compute node for execution.

(13) $\forall t_i \in T, p_j \in P: ft_{ij} \le eft_i \text{or} ft_{ij} > eft_i.$

Property 2

A running task cannot be preempted, namely, a running tasks cannot be interrupted during its execution and a task can be run only after the running task is completed.

$$(14)\quad \begin{aligned} &\forall t_i \in T: \sum_{j=1}^{|P|} x_{ij} = 1, \\ &\forall t_i, t_k \in T, x_{ij} = 1, [st_{kj}, ft_{kj}] \cap [st_{ij}, ft_{ij}] \neq \emptyset: x_{kj} = 0. \end{aligned}$$

Property 3

For a new task, the lowest *supply voltage* is firstly attempted. If the lowest supply voltage is unable to meet user expected finish time, the supply voltage is increased step by step until the user's expected finish time is met or the supply voltage reaches the highest level.

$$(15)\ \forall t_i \in T, p_j \in P, (\exists \min\{v_{ij}\}: ft_{ij} \leq eft_i) \text{or} (v_{ij} = V_{jk}: ft_{ij} > eft_i).$$

Property 4

If a new task cannot be finished within its user expected time even with the highest supply voltage, the supply voltages of tasks that await in the local queue for execution will be adjusted to approach the user's expected finish time.

Property 4 implies that the finish times and execution times of tasks waiting in a local queue can be modified, and thus the start times of the following tasks also.

Property 5

The scheduling event is triggered as a new task arrives, i.e., *immediate* mode is employed.

Now we analyze the available time at_{ij} of task t_i on compute node p_j, which is defined as the arrival time of t_i on p_j. at_{ij} can be approximated as follows:

$$(16)\ at_{ij} = tst_{ij} + tt_{ij} = tst_{ij} + \frac{l_i}{tr_j},$$

where tst_{ij} is the transmission start time of task t_i to compute node p_j.

The start time st_{ij} of task t_i on compute node p_j is in one of three options:

$$(17)\ st_{ij} = \begin{cases} at_{ij} & \text{if} \sum_{i=1}^{|T|} w_{ij} \cdot x_{ij} = 0 \text{and} r_j = 0, \\ at_{ij} + rt_j & \text{if} \sum_{i=1}^{|T|} w_{ij} \cdot x_{ij} = 0, \\ at_{ij} + rt_j + \sum_{o_{kj} < o_{ij}, w_{kj} = 1} et_{kj} & \text{else.} \end{cases}$$

where $r_j = 0$ denotes no task is running on compute node p_j, and rt_j represents the remaining execution time of a running task on p_j. $\sum_{i=1}^{|T|} w_{ij} \cdot x_{ij} = 0$ denotes that there are no tasks in the local queue of p_j.

The finish time of task t_i on compute node p_j is equal to the sum of the start time st_{ij} and t_i's execution time on p_j:

$$(18)\ ft_{ij} = st_{ij} + et_{ij}.$$

The 3E scheduling strategy employs the earliest expected finish time first policy placing new and waiting tasks in a local queue. Thereby, we get the following property.

Property 6

The start times of tasks waiting in a local queue can be modified if a new incoming task requires increasing the supply voltages of some tasks in the queue.

Assume that t_i is the new task placed in the local queue of compute node p_j, and t_k is the task whose start time needs to be recalculated.

Case 1: if $o_{ij} = min\{o_{mj}|w_{mj} = 1\}$ and $o_{kj} = o_{ij} + 1$, then $st'_{kj} = st_{kj} + et_{ij}$. Fig. 2 illustrates an example of Case 1.

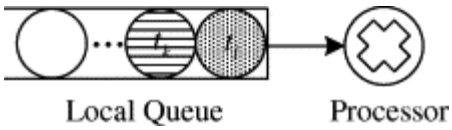

Fig. 2. An example of Case 1.

Case 2: if $o_{ij} = min\{o_{mj}|w_{mj} = 1\}$ and $o_{kj} > o_{ij} + 1$, then $st'_{kj} = st_{kj} + et_{ij} - \sum_{o_{mj}<o_{kj}, o_{mj}\neq o_{ij}} (et_{mj} - et'_{mj})$. Fig. 3 illustrates an example of Case 2.

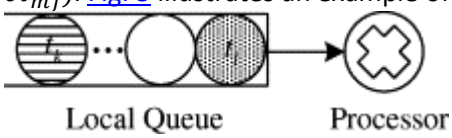

Fig. 3. An example of Case 2.

Case 3: if $o_{kj} = min\{o_{mj}|w_{mj} = 1\}$, then $st'_{kj} = st_{kj}$. Fig. 4 illustrates an example of Case 3.

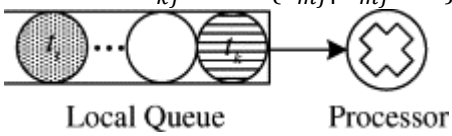

Fig. 4. An example of Case 3.

Case 4: if $o_{ij} \neq min\{o_{mj}|w_{mj} = 1\}$, $o_{kj} \neq min\{o_{mj}|w_{mj} = 1\}$, and $o_{kj} < o_{ij}$, then $st'_{kj} = st_{kj} - \sum_{o_{mj}<o_{kj}} (et_{mj} - et'_{mj})$. Fig. 5 illustrates an example of Case 4.

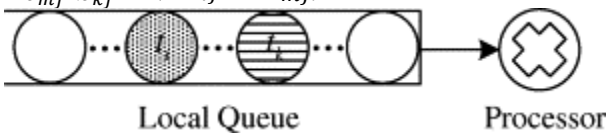

Fig. 5. An example of Case 4.

Case 5: if $o_{ij} \neq min\{o_{mj}|w_{mj} = 1\}$, $o_{kj} \neq min\{o_{mj}|w_{mj} = 1\}$, and $o_{kj} > o_{ij}$, then $st'_{kj} = st_{kj} - \sum_{o_{mj}<o_{kj}} (et_{mj} - et'_{mj} + et_{ij})$. Fig. 6 illustrates an example of Case 4.

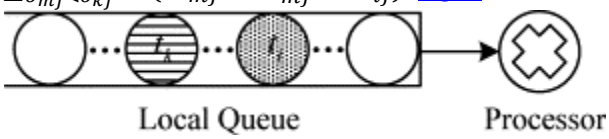

Fig. 6. An example of Case 5.

Hence, the new finish time ft'_{kj} of a task t_k in the local queue of a compute node p_j is:

(19) $$ft'_{kj} = st'_{kj} + et'_{kj} = st'_{kj} + \frac{l_k}{s(v'_{kj})}.$$

The 3E strategy uses heuristic algorithm. It performs the following operations when a new task arrives. First, it computes the start and finish times for the task on each compute node at the lowest supply voltage.

Next, if none of the compute nodes meets the user expectation, it increases the supply voltage gradually until the new task's finish time is earlier or equal to its user expected finish time. Third, if the highest-level supply voltage still cannot meet the user expected finish time for the new task, the 3E strategy will examine if it can increase the supply voltages of the existing tasks in the local queue. If there exists an allocation where both the adjusted existing tasks and the new task can meet the expected finish times, it sets the allocation. The 3E chooses the compute node with the smallest sum of node and transmission energy consumption to save energy.

The elasticity of our 3E lies in that it can flexibly adjust scheduling objectives based on the system workload. When the system is heavily loaded, the 3E strives to guarantee user expectations by increasing the supply voltages of new tasks and tasks waiting in local queues. In contrast, when the system is lightly loaded, the 3E is able to aggressively reduce energy consumption while maintaining user expectations.

The pseudocode of 3E scheduling strategy is shown in Algorithm 1.

Algorithm 1

Pseudocode of 3E scheduling strategy

1:	**for** each new task t_i **do**
2:	*mSelectedNode* ← *NULL*; *nMSelectedNode* ← *NULL*; *furtherAdjust* ← *TRUE*; *energyCons*← ∞; *meetExpectation* ← *FALSE*;
3:	**for** each compute node p_j in a heterogeneous computing system **do**
4:	Calculate the transmission energy consumption ec_{ij}^{tran} using Eq. (6);
5:	Calculate the start time st_{ij} using Eq. (17);
6:	vij←Vj1;
7:	**if** $st_{ij} + et_{ij} > eft_i$ **then**
8:	**adjustPhase1()**;
9:	**if** *furtherAdjust* = = *TRUE* **then**
10:	**adjustPhase2()**;
11:	**end if**
12:	**if** *meetExpectation* = = *FALSE* **then**
13:	**noMeetExpectation()**;
14:	**end if**
15:	**else**
16:	**noNeedAdjust()**;
17:	**end if**
18:	**end for**
19:	**if***mSelectedNode* ≠ *NULL* **then**
20:	Allocate t_i to *mSelectedNode* and update scheduling information;
21:	**else**
22:	Allocate t_i to *nMSelectedNode* and update scheduling information;
23:	**end if**
24:	**end for**

First, the 3E computes the transmission energy consumption and start time (see Lines 4 and 5, Algorithm 1). Second, it examines if a new task's expected finish time can be met with the lowest supply voltage (see Lines 6 and 7, Algorithm 1). If the initial test is not passed, the 3E calls **Function adjustPhase1()**.

Algorithm 2

Pseudocode of **Function adjustPhase1()**

1:	**while** $v_{ij} \leq V_{jk}$ **do**
2:	Increase one supply voltage level: $v'_{ij} \leftarrow v_{ij} + +$

3:	**if** $st_{ij} + et'_{ij} \leq eft_i$ **then**
4:	*furtherAdjust* ← *FALSE*;
5:	*meetExpectation* ← *TRUE*;
6:	Calculate the node energy consumption ec_{ij}^{node} using Eq. (1);
7:	**if** $ec_{ij}^{tran} + ec_{ij}^{node} \leq energyCons$ **then**
8:	$energyCons \leftarrow ec_{ij}^{tran} + ec_{ij}^{node}$;
9:	*mSelectedNode* ← *j*;
10:	break;
11:	**end if**
12:	**end if**
13:	**end while**

In **Function adjustPhase1()**, the 3E increases the supply voltage gradually for this new task until its user expectation is satisfied (lines 2–5, Algorithm 2). Next, the 3E selects the compute node with the least energy consumption (lines 6–11, Algorithm 2). After the while loop, if the variable *furtherAdjust* is equal to *TRUE* meaning the highest supply voltage cannot meet the user's expectation, the 3E further adjusts the supply voltage of waiting tasks in the local queue of this compute node, thus, the **Function adjustPhase2()** is called.

Algorithm 3

Pseudocode of **Function adjustPhase2()**

1:	**for** each task t_m in the local queue of p_j **do**
2:	**while** $v_{mj} \leq V_{jk}$ **do**
3:	Increase one supply voltage level: $v'_{mj} = v_{mj} + +$;
4:	Calculate t_m's new start time st'_{mj} and execution time et'_{mj};
5:	**if** $st'_{mj} + et'_{mj} \leq eft_m \text{and} st'_{ij} + et_{ij} \leq eft_i$ **then**
6:	*meetExpectation* ← *TRUE*;
7:	Calculate the new node energy consumption $pec_j = ec_{ij}^{node} + \sum_{w_{mj}=1} ec_{kj}^{'node}$;
8:	**if** $ec_{ij}^{tran} + pec_j \leq energyCons$ **then**
9:	*mSelectedNode* ← *j*;
10:	**end if**
11:	break;
12:	**else**
13:	$v'_{mj} = v_{mj} - -$;
14:	break;
15:	**end if**
16:	**end while**
17:	**end for**

For a task in the local queue, the 3E first increases the supply voltage step by step until it reaches the highest (lines 2 and 3, Algorithm 3). If the increased supply voltage cannot meet the expected finish times for the new task or the waiting task, the supply voltage is degraded to its pervious value (lines 12 and 13, Algorithm 3). Otherwise, if the new task can be finished within its user expectation (line 6, Algorithm 3), the 3E finds the compute node with the least energy consumption (lines 7–10, Algorithm 3).

If neither of **Function adjustPhase1()** and **Function adjustPhase2()** can set the variable *meetExpectation*to be *FALSE*, which means none of the supply voltage adjustments is able to meet the new task's expected finish time, the **Function noMeetExpectation()** is called.

Algorithm 4

Pseudocode of **Function noMeetExpectation()**

1:	**if** $ec_{ij}^{tran} + ec_{ij}^{node}(v_{ij} = V_{jk}) \leq energyCons$ **then**
2:	*nMSelectedNode* ← *j*;
3:	**end if**

In **Function noMeetExpectation()**, the 3E selects the compute node with the least energy consumption employing the highest supply voltage for the new task (see lines 1–3, Algorithm 4).
If the lowest supply voltage meets its user's expectation, the 3E calls the **Function noNeedAdjust()**.

Algorithm 5

Pseudocode of **Function noNeedAdjust()**

1:	**if** $ec_{ij}^{tran} + et_{ij}^{node}(v_{ij} = V_{j1}) \leq energyCons$ **then**
2:	*mSelectedNode* ← *j*;
3:	**end if**

The node with the least energy consumption is chosen in **Function noNeedAdjust()** to save energy (lines 1–3, Algorithm 5).

Let us go back to see Algorithm 1. A value of *mSelectedNode* that is not *NULL* indicates some compute nodes can execute the new task within the user expectation using our elastic voltage adjustment policy. In this case, the 3E allocates the new task to the selected node with the least energy consumption. Otherwise, no compute nodes can meet the user expected finish time for the new task. In this case, the 3E also selects a node with the least energy consumption (lines 19–23, Algorithm 1).

The time complexity of 3E depends on the number of compute nodes in a heterogeneous computing system, the number of tasks, and the number of supply voltage levels.

Theorem 1

The time complexity of scheduling a new task with 3E is $O(|P||Q||K|)$, where $|P|$ is the number of compute nodes in a heterogeneous computing system, $|Q|$ is the number of tasks in a local queue, and $|K|$ is the number of supply voltage levels.

Proof

The time complexity of Function *adjustPhase*1() is $O(|K|)$. Function *adjustPhase*2() consumes $O(|Q||K|)$ time. The time complexity of Functions *noMeetExpectation*() and *noNeedAdjust*() are $O(1)$. Other lines only consume $O(1)$. Thus, the time complexity of 3E is calculated as follows: $O(|P|)(O(|K|) + O(|Q||K|) + O(1) + O(1)) = O(|P||Q||K|)$.□

5. Performance evaluation

In this section, we evaluate the effectiveness of the proposed 3E scheduling strategy. We quantitatively compare 3E with three other algorithms:

- Greedy *energy-efficient* (GEE). GEE strives to guarantee user expected and reduce energy consumptions by adjusting the supply voltage of a newly arrived task. GEE does not adjust the supply voltage of tasks waiting in local queues of compute nodes.
- *Highest voltage energy-efficient (HVEE)*. HVEE offers the highest supply voltage for each new task and selects the compute node with the least energy consumption to execute the new task.
- *Lowest voltage energy-efficient (LVEE)*. LVEE provides the lowest supply voltage for each new task and selects the compute node with the least energy consumption to execute the new task.

We use three performance metrics to evaluate the algorithms.

- *Satisfaction rate*, the ratio of the number of tasks whose finish time meet their users' expectations to the total number of tasks×100%.
- *Total energy consumption*, the total energy consumption including the node energy consumption and transmission energy consumption.
- *Makespan*, the latest task finish time in the task set.

We use normalized total energy consumption in our study, a common practice used in literature (Kim et al., 2008, Laszewski et al., 2009) and (Liu et al., 2010).

5.1. Simulation method and parameters

Before presenting our experimental results, we present the simulation model as follows: Table 2summarizes the configuration parameters of the simulated heterogeneous computing systems used in our experiments. The parameters of nodes and links in the heterogeneous computing systems are chosen to resemble real-world processors.

Table 2. Parameters for simulation studies.

Parameter	**Value(fixed)–(varied)**
Number of compute nodes	(32)–(8, 16, 32, 56, 64, 96, 128)
Number of tasks	(2048)
minSpeed (kbps)	([250, 450])–([300, 400]), ([250, 450]), ([200, 500])
maxSpeed (kbps)	([900, 1100])–([950, 1050]), ([900, 1100]), ([850, 1150])
bandWidth (kbps)	([1250, 1400])
intervalTime (s)	(2.0)–(2.0, 4.0, 6.0, 8.0, 10.0, 12.0, 14.0, 16.0)
taskSize (kb)	([500, 1000])–([0, 500], [500, 1000], [1000, 1500])
finishTimeBase (s)	(2.0)–(2.0, 4.0, 6.0, 8.0, 10.0, 12.0, 14.0, 16.0)
nodeECR (W)	([14, 100])
transmissionECR (W)	([10, 20])

The voltage levels of all the compute nodes are from 0.9 V to 1.5 V with an increment of 0.1 V. Corresponding to the lowest and the highest voltages, the lowest and highest power consumptions are 14 W and 100 W respectively, and the slowest and fastest processing speeds are 350 kbps and 1000 kbps in average, respectively. The energy consumptions and performances for other voltages are determined by the energy and performance models.

The parameters and the values used in our simulations are summarized in Table 1.

1) The heterogeneity of the simulated distributed computing system is reflected by the nodes' processing speeds and network bandwidths. The minimum processing speed *minSpeed* corresponding to the lowest voltage Parameter is normally distributed across the compute nodes, and so are the maximum processing speed *maxSpeed* corresponding to the highest voltage level and the transmission rates *bandWidth* from the scheduler to different compute nodes.
2) Parameter *nodeECR* represents a range of node energy consumption rate from the minimal voltage to the maximal one. Again, *transmissionECR* is a range of transmission energy consumption rate from the worst link to the best one. Node energy consumption rate and transmission energy consumption rate are normally distributed in *nodeECR* and *transmissionECR*, respectively. This parameters are similar to that in Xie and Qin (2008). Additionally, in our experiments, we set the node energy consumption rate

being idle is a tenth of the node energy consumption rate when active; and the transmission energy consumption rates being active and idle are equal.

3) We study three task sizes: small, median and large in our simulations, where small size is less than 500 kb, median size is within 500–1000 kb, and large size is within 1000–1500 kb.
4) Parameter *finishTimeBase* determines whether a task's expected finish time is loose or tight. The expected finish time eft_i of task t_i in Eq. (20) is designed similar as that in Qin and Jiang (2006),

$$(20)\ eft_i = a_i + (1 + finishTimeBase) \times e_i^{max},$$

where e_i^{max} is the longest execution time that can be computed as follows:

$$(21)\ e_i^{max} = max\{e_{ij}(V_{j1})\}.$$

5) Parameter *intervalTime* represents the arrival interval between two consecutive tasks. The arrival rate of tasks is Poisson distribution.

5.2. Scalability

Scalability is an important measure of scheduling strategy for large scale systems. In our experiments, we examine how the four strategies perform when the count of compute nodes increases from 8 to 128. Fig. 7, Fig. 8, Fig. 9 show satisfaction rate, total energy consumption, and makespan with each strategy.

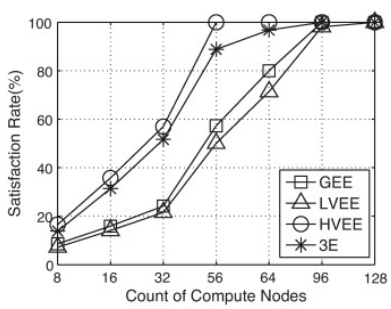

Fig. 7. Performance impact of the count of compute nodes on satisfaction rate.

Fig. 7 shows that satisfaction rate improves with the increase of compute node count for all strategies. This is because more nodes provide more computing resources. HVEE always yields higher satisfaction rate than other strategies as it sets the voltage to the highest level all the time and thus results in less execution times for tasks and earlier start times. Hence, the likelihood of missing user expected finish time is reduced. In contrast, LVEE offers the worst satisfaction rate because of employing the lowest voltage level all the time. The 3E strategy outperforms GEE as it utilizes the information of tasks in the local queues in addition to newly arrived tasks and adjusts their supply voltage levels when the system is under heavy workload.

Fig. 8 shows how the total energy consumption varies with the count of compute nodes. HVEE achieves the lowest energy efficiency while LVEE achieves the highest. These results indicate that both baseline schemes have no elasticity, lacking the capacities to make trade-offs between satisfaction rate and total energy consumption and to respond to the dynamics of system workload. GEE exhibits the similar trend as 3E, with a lower total energy consumption at the expense of smaller satisfaction rate. This is explained by the fact that 3E aggressively enhances the satisfaction rate when the system is heavily loaded. However, the difference between 3E and GEE becomes negligible when the number of compute nodes is larger than 96. Interestingly, the total energy consumption with 3E decreases when the count of compute nodes increases from 8 to 64. This is because 3E strives to reduce energy consumption while keeping high satisfaction rate when the system workload becomes lighter. Additionally, when the count of compute nodes is more than 64, the total energy consumption with 3E goes up as most of the compute nodes are sitting idle, leading to increasing idle energy consumption even if the compute nodes are scheduled at the lowest voltage levels.

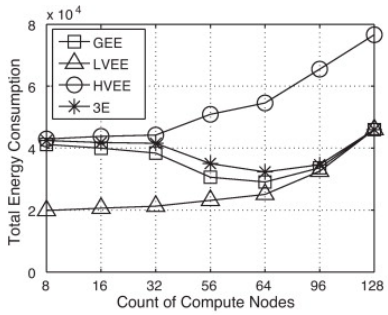

Fig. 8. Performance impact on the count of compute nodes in terms of total energy consumption.

Fig. 9 shows that LVEE delivers the worst makespan as a result of deploying the lowest voltage level and the least energy consumption for tasks. On the contrary, HVEE delivers the best makespan at the expense of higher energy consumption. With these two strategies, energy consumption and makespan do not compensate for each other. The results are the consequences of the lack of elasticity in the scheduling strategies in heterogeneous computing systems. 3E delivers better makespan than GEE when the compute node count is less than 96. To guarantee higher satisfaction rate when the system is under heavy workload, 3E boosts the supply voltage levels for some tasks in local queues leading to shorter execution times, whereas GEE lacks the capability to shorten the execution times of those waiting tasks. Consequently, tasks scheduled by 3E have shorter execution times with higher throughput compared with GEE. The two strategies exhibit identical makespan when the number of compute nodes is larger than 96 because the system workload is light enough for both to schedule tasks with the lowest voltage level.

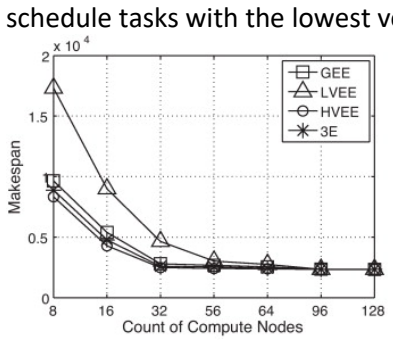

Fig. 9. Performance impact on the count of compute nodes in terms of makespan.

5.3. Arrival rate

To examine the performance sensitivities of the four strategies to the arrival rate of tasks, in this set of experiments, we vary the parameter *intervalTime* from 2 to 16 with increment of 2. Fig. 10, Fig. 11, Fig. 12 plot the performances of GEE, LVEE, HVEE, and 3E.

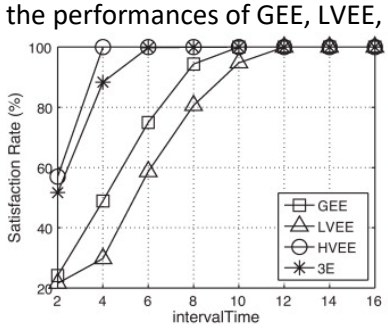

Fig. 10. Performance impact of the arrival rate on satisfaction rate.

The first observation drawn from Fig. 10 is that, for all strategies, the satisfaction rate is improved with the increase of *intervalTime*. This is because the smaller *intervalTime* means more frequent tasks arrivals, more tasks waiting in the local queues, and heavier system workload. Consequently, a newly arrived task has to wait

until the tasks with higher orders are finished. The long-time waiting increases the possibility of missing user expected finish time. With larger *intervalTime*, the number of waiting tasks in local queues becomes smaller, the system workload becomes lighter, and a newly arrived task can be started earlier. Again, we observe that HVEE and LVEE generate the highest and the lowest satisfaction rates, respectively, while 3E performs better than GEE. We attribute the results to the fact that HVEE and LVEE consistently maintain the highest voltage level and the lowest voltage level for each new task without flexibility, and 3E is able to adjust the voltage levels of waiting tasks to improve the satisfaction rate when the system is under heavy workload.

The results reported in Fig. 11 demonstrate that most energy is consumed with HVEE and least energy is consumed with LVEE. The 3E strategy has unique features. For instance, when the value of parameter *intervalTime* varies from 2 to 8, the system workload becomes lighter, 3E dynamically degrades the supply voltage levels of tasks to reduce energy consumption with the constraint that the user expected finish times are met. The total energy consumption by a 3E-enabled heterogeneous computing system increases when *intervalTime* is larger than 8. This is because the light workload produces more idle time and thus increasing energy consumption. This phenomenon is more obvious when the node count is large enough. An interesting observation from Fig. 11 is that GEE performs similarly as 3E and is slightly more energy-efficient when the *intervalTime* is larger than 8. However, this is at the expense of satisfaction rate under heavy system workload.

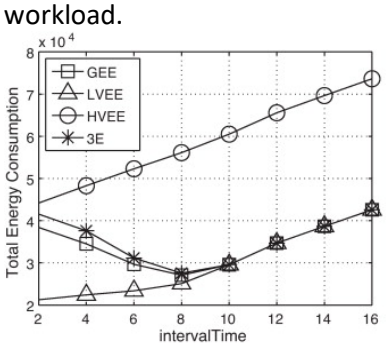

Fig. 11. Performance impact of the arrival rate on the total energy consumption.

In Fig. 12, we observe that when *intervalTime* varies from 2 to 16, the makespan under all the four schemes increase. This is because tasks arrive less frequently if the value of *intervalTime* increases, thus, tasks start later and finish later compared with those with smaller *intervalTime*. Another important observation is that HVEE generates the best makespan, while LVEE is the opposite. 3E outperforms GEE in terms of makespan when the *intervalTime* is less than 8. As the system workload is heavy, 3E trades low energy efficiency for high satisfaction, leading to shorter execution times of tasks.

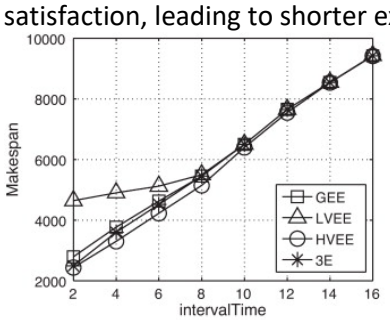

Fig. 12. Performance impact of the arrival rate on makespan.

5.4. Expected finish time

This subsection discusses the performance impact of user expected finish times with GEE, LVEE, HVEE, and 3E. We vary the parameter *finishTimeBase* from 2 to 16. Fig. 13, Fig. 14, Fig. 15 plot the performances of the four policies.

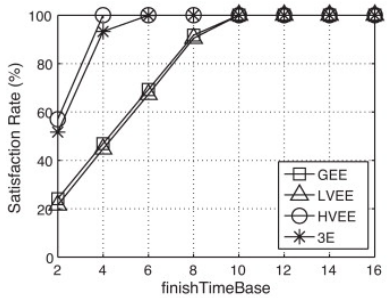

Fig. 13. Performance impact of the *finishTimeBase* on satisfaction rate.

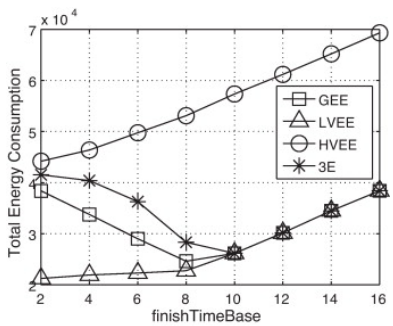

Fig. 14. Performance impact of *finishTimeBase* on the total energy consumption.

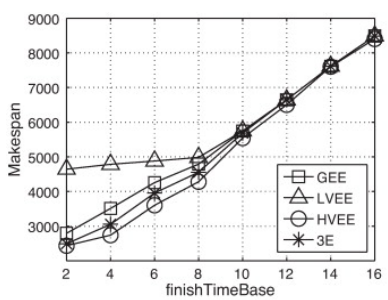

Fig. 15. Performance impact of *finishTimeBase* on makespan.

We observed from Fig. 13 that with the increase of *finishTimeBase* and looser constraint for user expected finish time for tasks, the satisfaction rate with each strategy increases accordingly. As the timing requirement becomes loose, and tasks can finish late yet still meet user expected finish times. Additionally, Fig. 13 demonstrates that HVEE and LVEE have the highest and the lowest satisfaction rate, respectively. Again, the satisfaction rate under 3E is better than that under GEE. These results are consistent with what are observed from Figs. 7 and 10).

Increasing *finishTimeBase* reflects the system workload becomes lighter. We observe from Fig. 14 that when the value of *finishTimeBase* is less than 10, the energy consumption under 3E gradually decreases as a result of good elasticity. We also observe that when the *finishTimeBase* is more than 10, the result is consistent with previous simulations. Although GEE has a similar trend with 3E, it yields lower satisfaction rate than 3E when the system is heavily loaded. In addition, both HVEE and LVEE are unable to reduce the energy consumption no matter what the system workload is.

Fig. 15 shows that LVEE has longer makespan than the other schemes because it always uses the lowest voltage level. HVEE is opposite of LVEE. 3E making better makespan than GEE is because the execution times of some tasks with 3E are shortened to guarantee high satisfaction rate.

5.5. Task size

In this set of experiments, we evaluate the performance impact of task size. We test three configurations of task size, as described in Table 1. Fig. 16, Fig. 17, Fig. 18 depict the performances of the four schemes under small, median, and large task sizes.

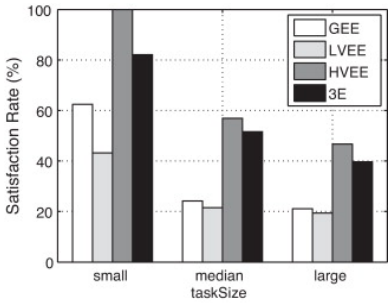

Fig. 16. Performance impact of the *taskSize* on satisfaction rate.

The results in Fig. 16 reveal that when the task size is small, all the strategies are able to provide higher satisfaction rate because of the shortened tasks' execution times. For median and large task sizes, the execution time are longer and the satisfaction rates are smaller. HVEE always offers the highest satisfaction while LVEE offers the lowest because of the static nature and lack of ability of adjusting voltage levels according to the system workload. 3E has higher satisfaction rate than GEE as 3E strives to meet the user's expectations at the cost of energy efficiency when the system is under heavy workload.

Fig. 17 shows that when the *taskSize* varies from small granularity to large one, the total energy consumption of a heterogeneous computing system under all the tested methods goes up. This is because the larger size tasks require longer execution times and thus more energy consumption. HVEE has the highest energy consumption and LVEE has the lowest energy dissipation, as exhibited in the previous experiments. Interestingly, when the task size is median or large, the energy consumption with 3E is slightly more than that with GEE because 3E weights more on the high satisfaction rate when the system is heavy loaded. However, when the task size is small, 3E and GEE have basically identical energy efficiency.

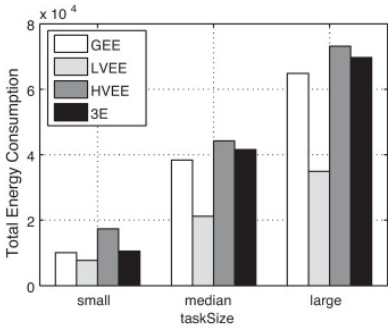

Fig. 17. Performance impact of the *taskSize* on total energy consumption.

The makespans of the four strategies shown in Fig. 18 indicate that the elasticity of 3E is good. These results are consistent with the ones plotted in Figs. 12 and 15.

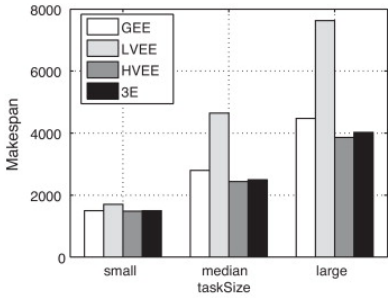

Fig. 18. Performance impact of the *taskSize* on makespan.

5.6. Compute node heterogeneity levels

The experiments in these subsection focus on the impact of compute node heterogeneity on system performance. To be specific, we evaluate three node heterogeneity degrees: small heterogeneity, middle heterogeneity, and large heterogeneity. Fig. 19, Fig. 20, Fig. 21 depict the performances of GEE, LVEE, HVEE, and 3E.

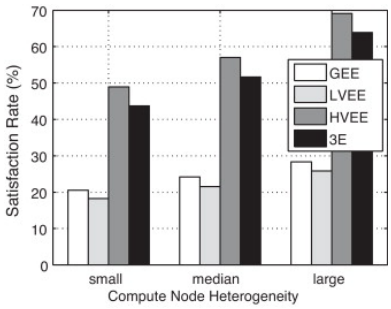

Fig. 19. Performance impact of the compute node heterogeneity on satisfaction rate.

In Fig. 19, we observe that the satisfaction rate of all strategies are boosted with the increase of compute node heterogeneity. This is mainly because more tasks are allocated to nodes with larger processing capability and less tasks are allocated to nodes with smaller processing capability. 3E outperforms others except for HVEE because 3E judiciously adjusts the supply voltages of queuing tasks in local queues. HVEE holds the highest satisfaction rate at the expense of consuming the most energy.

We observe from Fig. 20 that HVEE and LVEE consistently provide the highest and the lowest energy consumption, respectively. The reason is same as the explanations in earlier discussion. Fig. 15 also shows that 3E is no more energy-efficient than GEE. 3E gives priority to satisfaction rate although more energy is consumed when the system is under heavy workload.

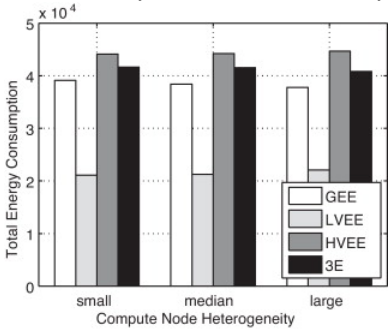

Fig. 20. The impact of compute node heterogeneity on total energy consumption.

Fig. 21 shows that the makespan of each strategy slightly increases when the compute node heterogeneity becomes large. This is because the system workload decreases a little with longer execution time that is leveraged for energy conservation. Again, 3E has better makespan than GEE due to shorter tasks execution times and better satisfaction rate under heavy workload.

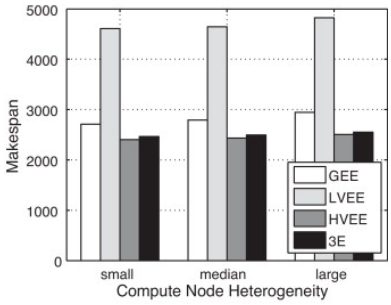

Fig. 21. The impact of computational node heterogeneity on satisfaction rate.

6. Conclusions and future work

In this paper, we have addressed the issue of scheduling and allocating independent tasks on heterogeneous computing systems to make trade-offs between users' expectations and energy efficiency. The proposed energy-efficient elastic (3E) scheduling strategy can efficiently improve the flexibility of heterogeneous computing systems by adaptively adjusting supply voltages of both new tasks and queued tasks according to the system workload. We have quantitatively evaluated the effectiveness of 3E strategy in extensive simulation studies, and the experimental results reveal that 3E outperforms other existing and baseline strategies due to its elasticity, and is a feasible scheduling strategy in dynamic environments.

Our future studies will focus on two avenues. First, we would like to extend 3E scheduling strategy to deal with heterogeneous storage systems and second, we intend to modify 3E scheme to handle parallel tasks on heterogeneous computing systems.

Acknowledgement

This research was supported by the National Natural Science Foundation of China under Grant No. 61104180.

References

Goller and Leberl, 2009 A. Goller, F. Leberl **Radar image processing with clusters of computers** IEEE Aerospace and Electronics Systems Magazine, 24 (January (1)) (2009), pp. 18-22

Zheng et al., 2006 K. Zheng, J. Wang, L. Huang, G. Decarreau **Open wireless software radio on common PC** Proc. 17th Ann. IEEE Int'l Symp. Personal, Indoor and Mobile Radio Communication (PIMRC'06), September (2006), pp. 1-5

Donoho, 2004 G. Donoho **Building a web service to provide real-time stock quotes** MCAD Net (February) (2004)

Xie and Qin, 2008 T. Xie, X. Qin **An energy-delay tunable task allocation strategy for collaborative applications in networked embedded systems** IEEE Transactions on Computers, 57 (March (3)) (2008), pp. 329-343

Bianchini and Rajamony, 2004 R. Bianchini, R. Rajamony **Power and energy management for server systems** Computer, 37 (November (11)) (2004), pp. 68-74

Feng, 2003 W. Feng **Making a case for efficient supercomputing** ACM Queue, 1 (7) (2003), pp. 54-64

http, in pressa http://www.transmeta.com

http, in pressb http://www.athlon.com

Chen et al., 2006 J. Chen, C. Yang, T. Kuo **Slack reclamation for real-time task scheduling over dynamic voltage scaling multiprocessors** Proc. IEEE Int'l Conf. Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC'06), June (2006), pp. 358-367

Braun et al., 2001 T.D. Braun, H.J. Siegel, N. Beck, *et al.* **A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems** Journal of Parallel and Distributed Computing, 61 (June (6)) (2001), pp. 810-837

Zhu and Lu, 2008 X. Zhu, P. Lu **Study of scheduling for processing real-time communication signals on heterogeneous clusters** Proc. 9th Int'l Symp. Parallel Architectures, Algorithms, and Networks (I-SPAN'08), May (2008), pp. 121-126

Fujimoto and Hagihara, 2006 N. Fujimoto, K. Hagihara **A 2-approximation algorithm for scheduling independent tasks onto a uniform parallel machine and its extension to a computational Grid** Proc. the IEEE Int'l Conf. Cluster Computing (CLUSTER'06) (Sept. 2006), pp. 1-7

Wang et al., 2010 H. Wang, J. Li, W. Huang, D. Qiu **Area census-oriented electronic reconnaissance satellites scheduling technique under uncertain space-frequency domain environments** Proc. Int'l Conf. Electronics and Information Engineering (ICEIE'10), August (2010), pp. 443-448

Doˇgan and Özgüner, 2006 A. Doˇgan, F. Özgüner **Scheduling of a meta-task with QoS requirements in heterogeneous computing systems** Journal of Parallel and Distributed Computing, 66 (February (2)) (2006), pp. 181-196

Mehta et al., 2007 A.M. Mehta, J. Smith, H.J. Siegel, A.A. Maciejewski, A. Jayaseelan, B. Ye **Dynamic resource allocation heuristics that manage tradeoff between makespan and robustness** Journal of Supercomputing, 42 (October (1)) (2007), pp. 33-58

Kang and He, 2009 Q. Kang, H. He **A novel discrete differential evolution algoritnm for task scheduling in heterogeneous computing systems** Proc. IEEE Int'l Conf. Systems, Man, and Cybernetics (SMC'09), October (2009), pp. 5006-5011

Wang et al., 2007 J. Wang, J. Li, Y. Tan **Study on heuristic algorithm for dynamic scheduling problem of earth observing satellites** Proc. 8th ACIS Int'l Conf. Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'07), July–August (2007), pp. 9-14

Coffman, 1976 E.G. Coffman **Computer and Job-Shop Scheduling Theory** John Wiley & Sons (1976)

Kim et al., 2008 J. Kim, H.J. Siegel, A.A. Maciejewski, R. Eigenmann **Dynamic resource management in energy constrained heterogeneous computing systems using voltage scaling** IEEE Transactions on Parallel and Distributed Systems, 19 (November (11)) (2008), pp. 1445-1457

Karatza, 2009 H.D. Karatza **Performance of gang scheduling strategies in a parallel system** Simulation Modelling Practice and Theory, 17 (February (2)) (2009), pp. 430-441

Aydin et al., 2004 H. Aydin, R. Melhem, D. Mossé, P. Mejía-Alvarez **Power-aware scheduling for periodic real-time tasks** IEEE Transactions on Computers, 53 (May (5)) (2004), pp. 584-600

Mishra et al., 2003 R. Mishra, N. Rastogi, D. Zhu, D. Mossé, R. Melhem **Energy aware scheduling for distributed real-time systems** Proc. Int'l Parallel and Distributed Processing Symp. (IPDPS'03), April (2003), pp. 21-29

Yu and Prasanna, 2002 Y. Yu, V.K. Prasanna **Power-aware resource allocation for independent tasks in heterogeneous real-time systems** Proc. 9th Int'l Conf. Parallel and Distributed Systems (ICPADS'02), December (2002), pp. 341-348

Zhu et al., 2003 D. Zhu, R. Melhem, B.R. Childers **Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems** IEEE Transactions on Parallel and Distributed Systems, 14 (July (7)) (2003), pp. 686-700

Ge et al., 2005 R. Ge, X. Feng, K.W. Cameron **Performance-constrained distributed DVS scheduling for scientific applications on power-aware clusters** Proc. ACM/IEEE Conference on Supercomputing (SC'05), November (2005), pp. 34-44

Tavares et al., 2008 E. Tavares, B. Silva, P. Maciel **An environment for measuring and scheduling time-critical embebbed systems with energy constraints** Proc. 6th IEEE Int'l Conf. Software Engineering and Formal Methods (SEFM'08), November (2008), pp. 291-300

Nélis et al., 2008 V. Nélis, J. Goossens, R. Devillers, D. Milojevic, N. Navet **Power-aware real-time scheduling upon identical multiprocessor platforms** Proc. IEEE Int'l Conf. Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC'08), June (2008), pp. 209-216

Hu et al., 2008 L. Hu, H. Jin, X. Liao, X. Xiong, H. Liu **Magnet: a novel scheduling policy for power reduction in cluster with virtual machines** Proc. IEEE Int'l Conf. Cluster Computing (CLUSTER'08), September (2008), pp. 13-22

Laszewski et al., 2009 G. Laszewski, L. Wang, A.J. Younge, X. He **Power-Aware Scheduling of Virtual Machines in DVFS-Enabled Clusters** Proc. IEEE Int'l Conf. Cluster Computing (CLUSTER'09), August (2009), pp. 1-10

Liu et al., 2010 W. Liu, H. Li, F. Shi **Energy-efficient task clustering scheduling on homogeneous clusters** Proc. 11th Int'l Conf. Parallel and Distributed Computing, Applications and Technologies (PDCAT'10), December (2010), pp. 381-385

Zong et al., 2011 Z. Zong, A. Manzanares, X. Ruan, X. Qin **EAD and PEBD: two energy-aware duplication scheduling algorithms for parallel tasks on homogeneous clusters** IEEE Transactions on Computers, 60 (March (3)) (2011), pp. 360-374

Hamano et al., 2009 T. Hamano, T. Endo, S. Matsuoka **Power-aware dynamic task scheduling for heterogeneous accelerated clusters** Proc. 4th Workshop on High-Performance, Power-Aware Computing (HPPAC'09), May(2009), pp. 1-8

Zong et al., 2007 Z. Zong, X. Qin, X. Ruan, K. Bellam **Energy-efficient scheduling for parallel applications running on heterogeneous clusters** Proc. Int'l Conf. Parallel Processing (ICPP'07), September (2007), pp. 19-26

Shekar and Izadi, 2010 V. Shekar, B. Izadi **Energy aware scheduling for DAG structured applications on heterogeneous and DVS enabled processors** Proc. Int'l Conf. Green Computing (GREENCOMP'10), August (2010), pp. 495-502

Zikos and Karatza, 2011 S. Zikos, H.D. Karatza **Performance and energy aware cluster-level scheduling of compute-intensive jobs with unknown service times** Simulation Modelling Practice and Theory, 19 (January (1)) (2011), pp. 239-250

Yan et al., 2005 L. Yan, J. Luo, N.K. Jha **Joint dynamic voltage scaling and adaptive body biasing for heterogeneous distributed real-time embedded systems** IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 24(July (7)) (2005), pp. 1030-1041

Chen et al., 2007 J. Chen, C. Yang, T. Kuo, C. Shih **Energy-efficient real-time task scheduling in multiprocessor DVS systems** Proc. 12th Asia and South Pacific Design Automation Conf. (ASP-DAC'07), January(2007), pp. 342-349

Hung et al., 2006 C. Hung, J. Chen, T. Kuo **Energy-efficient real-time task scheduling for a DVS system with a non-DVS processing element** Proc. 27th IEEE Int'l Real-Time Systems Symp. (RTSS'06), December (2006), pp. 303-312

Terzopoulos and Karatza, 2011 G. Terzopoulos, H. Karatza **Performance evaluation of a real-time grid system using power-saving capable processor** Journal of Supercomputing (2011), 10.1007/s11227-011-0689-y

Zhu et al., 2011 X. Zhu, X. Qin, M. Qiu **QoS-aware fault-tolerant scheduling for real-time tasks on heterogeneous clusters** IEEE Transactions on Computers, 60 (June (6)) (2011), pp. 800-812

Zhu and Lu, 2009 X. Zhu, P. Lu **Multi-dimensional scheduling for real-time tasks on heterogeneous clusters** Journal of Computer Science and Technology, 24 (March (3)) (2009), pp. 434-446

Zhu and Lu, 2009 X. Zhu, P. Lu **A two-phase scheduling strategy for real-time applications with security requirements on heterogeneous clusters** Computers & Electrical Engineering, 35 (November (6)) (2009), pp. 980-993

Zhu et al., 2011 J. Zhu, X. Zhu, J. Jiang **Improving adaptivity and fairness of processing real-time tasks with QoS requirements on clusters through dynamic scheduling** Information Processing Letters, 111 (June (12)) (2011), pp. 609-613

Qin and Jiang, 2006 X. Qin, H. Jiang **A novel fault-tolerant scheduling algorithm for precedence constrained tasks in real-time heterogeneous systems** Journal of Parallel Computing, 32 (August (5)) (2006), pp. 331-356