EDF-hv: An Energy-Efficient Semi-Partitioned Approach for Hard Real-Time Systems

Jesse Patterson
Electrical and Computer Engineering
Utah State University
4120 Old Main Hill
Logan, Utah 84322
jesse.patterson@aggiemail.usu.edu

Thidapat Chantem
Electrical and Computer Engineering
Virginia Tech
900 N. Glebe Road
Arlington, VA 22203
tchantem@vt.edu

ABSTRACT

The shift from uniprocessor to multi-core architectures has made it more difficult to design predictable hard real-time systems (HRTS) and achieving high processor utilization remains a major challenge. As energy efficiency becomes an important design metric in HRTS, most systems use dynamic voltage and frequency scaling (DVFS) to reduce dynamic power consumption when the system is underloaded. However, for many multi-core systems, DVFS is implemented using voltage and frequency islands (VFI), implying that individual cores cannot independently select their voltage and frequency (v/f) pairs, which results in less energy savings when existing energy-aware task assignment and scheduling techniques are used. In this work, we present an analysis of the increase in energy consumption in the presence of VFI. Further, we propose a semi-partitioned approach called EDF-hv to reduce the energy consumption of HRTS on multi-core systems with VFI. Simulation results revealed that when workload imbalance among the cores is sufficiently high, EDF-hv can reduce system energy consumption by 15.9% on average.

Keywords

Energy efficiency; hard real-time systems; multi-core systems; dynamic voltage and frequency scaling; voltage and frequency islands; semi-partitioning

1. INTRODUCTION

Most modern multi-core processors employ some types of power management (PM) schemes to improve energy efficiency. Typically, PM is broadly broken into static (or leakage) and dynamic components, where power-gating is used to reduce static power and dynamic voltage and frequency scaling (DVFS) is used to reduce dynamic power [16, 11]. However, due to shrinking technology nodes, increased clock speeds, and an increase in the number of processing cores, providing an independent power supply and clock to each core in the multi-core to implement PM in hardware has become increasingly difficult to realize. To overcome this challenge, most modern multi-core processors are built using voltage and frequency islands (VFI) [8, 12, 17], which distribute a single voltage (i.e., power) and frequency (i.e., clock) to a group of cores on the multi-core. Although VFI simplifies implementation, it also restricts the effectiveness of the underlying PM schemes as each core cannot set its own voltage or frequency independently of the VFI, i.e., each core must operate at the voltage and frequency (v/f) pair of the VFI [20].

To make matters worse, for hard real-time systems (HRTS) where deadlines cannot be violated, each VFI must accommodate the needs of the core with the highest demand; which can lead to a less than optimal PM solution. For example, consider the system shown in Figure 1. The system has sixteen cores grouped onto four VFI (i.e., each VFI has four cores). Suppose the workload can be scheduled in such a way that deadlines can be guaranteed by using the full capacity of six cores (i.e., six core must operate at maximum capacity to execute the workload on time). Since static power now dominates total power consumption [20], in general, the optimal static PM solution is to only have six operating cores and power-gate the remaining ten. Yet, due to the restrictions imposed by VFI, the system cannot attain this optimal solution as only four or eight cores can be power-gated while having at least six operating cores available. Thus, the limitations of VFI prevent the actual implementation of the optimal static power solution.

In addition, the presence of VFI also imposes limitations on DVFS for managing dynamic power. As mentioned, for HRTS, the VFI must accommodate the needs of the core with the highest demand. This means that if one core requires a v/f pair above that of the other cores on the same VFI, then all the cores on the VFI must operate at the increased v/f pair. Hence, the system not only pays the increased dynamic cost for the core that needs the increased v/f pair, but also the increased dynamic cost for all other cores on the same VFI. This increased dynamic power cost due to cores operating above their optimal v/f pair is referred to herein as the *VFI cost.*

One possible solution to improve the dynamic power cost is to distribute the workload evenly across the operating cores, a principle referred to herein as *load-balancing.* If all the cores have the same workload, then typically all the cores will need approximately the same v/f pair (subject to pro-

RTNS '16, October 19 - 21, 2016, Brest, France
© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-4787-7/16/10…$15.00
DOI: http://dx.doi.org/10.1145/2997465.2997491

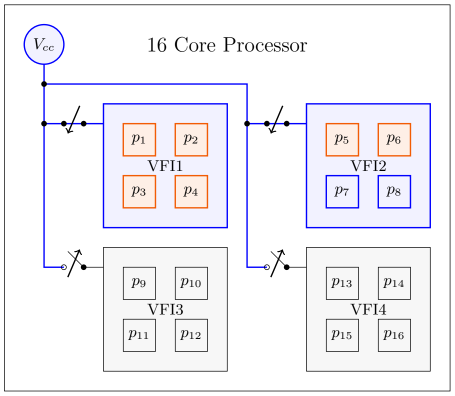


Figure 1: Example of the limitations imposed by VFI on static power. When only six cores are needed, two VFI must be switched-on and the other two VFI can be power-gated (i.e., switched-off). However, this leaves the system with eight operating cores consuming energy.

cess variation) and thus the VFI cost is minimized. Consider again the system in Figure 1. Since a minimum of eight cores must be turned on for the workload to meet their deadlines, even though only six cores are needed, using all eight cores is more energy efficient. When the workload is scheduled on only six of the eight operating cores, then at least two cores on each VFI will need to operate at full capacity; thus both VFI must operate at the maximum v/f pair. Conversely, load-balancing the workload on all eight cores allows the v/f pair of both VFI to be reduced by 25% and still meet deadlines. Therefore, load-balancing the workload on all eight cores has the potential to reduce dynamic power cost by up to 58% (assuming a CMOS circuit where the dynamic power is proportional to the cube of the clock frequency).

Unfortunately, achieving a load-balance that is suitable for HRTS is not always possible. Typically, to be able to guarantee deadlines, partitioned scheduling is used. In partitioned scheduling, the workload is partitioned into shares such that each share is assigned to its own core and then each core only executes the share of the workload it was assigned. Although in some cases it is possible for the workload to be broken into equal shares, the more typical case is that each core is assigned different sized share and thus the workload is imbalanced. As such, we propose Earliest Deadline First for HRTS with VFI (EDF-hv), which utilizes a semi-partitioned approach to achieve a load-balance and thereby improves energy efficiency in the presence of VFI. The main contributions of this work are as follows.

- To the best of our knowledge, this is the first work to quantify the energy implications of VFI in HRTS. Herein, we define the concepts of imbalance and VFI cost and derive the best- and worst-case bounds on said VFI cost.
- Simulation results pertaining to VFI cost, approximating the average behavior, are shown with its correlation to system imbalance.
- A scheduling algorithm, namely EDF-hv, which is suitable for HRTS, is presented to improve the energy efficiency of the system in the presence of VFI.
- Finally, simulation results that allow the performance of EDF-hv to be assessed are presented and a classification of the types of systems that will benefit from EDF-hv are described.

The rest of this paper is organized as follows. In Section 2, the related work is presented. Section 3 provides the system model. The imbalance and VFI cost are described in Section 4. To mitigate the effects of VFI cost, we introduce EDF-hv in Section 5. Section 6 presents simulation results pertaining to the increase in energy efficiency as obtained by EDF-hv. Section 7 concludes this paper and presents future research directions.

2. RELATED WORK

The work in [6] categorizes multi-core real-time scheduling algorithms as global, partitioned, or a hybrid of the two. These scheduling algorithms are classified based on the amount of task migration that is allowed among the cores. Partitioned scheduling prohibits migration and, conversely, global scheduling permits unrestricted migration. Further, [6] calculates how much of the system's total capacity the task set can require while still guaranteeing a feasible schedule. Partitioned scheduling minimizes scheduling overhead, as there is no task migration, and once tasks have been assigned to cores, deadlines can be guaranteed using the Earliest Deadline First (EDF) [15] scheduling algorithm. However, if the task set will use more than approximately 50% of the total capacity of a system, the task set is not guaranteed to be schedulable [6]. On the other hand, global scheduling algorithms, such as proportionate fairness (p-fair) [7], can schedule up to 100% of the systems capacity; however, the scheduling overhead is so large that these algorithms are not practical in reality.

In an attempt to combine the best of partitioned and global scheduling, a hybrid approach to scheduling that allows restricted task migration has been developed [3, 1, 2, 14, 10] and is typically referred to as semi-partitioning. In semi-partitioning, limited task migration is permitted for specific tasks while other tasks are only allowed to execute on their assigned core. In this work, we leverage the semi-partitioning heuristics proposed by Anderson et. al. in [1, 2]. The work in [1] presents EDF-fm, a semi-partitioning algorithm that guarantees feasibility up to 100% of the system capacity with the restriction that no single task can have a utilization greater than 50% of a single core. In [2], EDF-fm was extended to EDF-os, which removes the per task utilization requirements, i.e., tasks can have a utilization up to 100% of a single core. Neither of these semi-partitioning heuristics can guarantee deadlines; however, both have a bound on how late the tasks can be, referred to as bounded tardiness. Further, in both EDF-fm and EDF-os, to reduce migration overhead, a migrating task can only migrate from one core to another after a job/iteration of the task has fully completed, a property referred to as *boundary-limited.*

EDF-os [2] is designed for periodic tasks on a homogeneous multi-core system. As a semi-partitioning algorithm, task are either fixed (i.e., the task is assigned to execute on only one core) or migrating (i.e., the task is assigned to execute a defined share of jobs on a subset of the cores). In EDF-os, tasks are partitioned in two steps. In the first step, as many tasks as possible are assigned to one of the cores as fixed tasks using the worst-fit decreasing (WFD) heuristic

[9]. When a task is encountered that cannot be assigned to a core using WFD, the second part of the partition begins. In the second step, using a next-fit-like heuristic, the remaining tasks are assigned a share on cores that still have available capacity until the complete utilization of the task is allotted. Under EDF-os, a migrating task can execute on more than two cores; however, each core is limited to a maximum of two migrating tasks. After the tasks have been partitioned, the migrating tasks will execute the defined share of jobs on each of its cores and migration is determined with a p-fair algorithm [7]. Fixed tasks are prioritized using EDF while migrating tasks have fixed priority over fixed tasks. Additionally, on cores that have two migrating tasks, one migrating task has a fixed priority over the other. Due to the method in which tasks are assigned to the cores, Anderson et al. were able to derive tardiness bounds for the tasks [2].

3. SYSTEM MODEL

We consider a periodic task set, $\eta = \{\tau_1, \tau_2, ..., \tau_\mathrm{N}\}$, of N independent tasks to be scheduled on a homogeneous multi-core system with M identical processing cores, $P = \{p_1, p_1, ..., p_\mathrm{M}\}$, on a set of I VFI, $\Gamma = \{\gamma_1, \gamma_2, ..., \gamma_\mathrm{I}\}$, where N, M, and I are positive integers. The system must have at least one VFI and, since the system is homogeneous, each VFI, γ_l, must have the same number of cores on it. For ease of discussion, P_l is the set of cores on γ_l and M_l is the number of cores in P_l. Further, since this work considers multi-core systems with multiple cores per VFI, $\mathrm{M} \geq \mathrm{M}_l > 1$.

Each VFI can adjust its voltage independently of the other VFI to operate at a set of K discrete frequencies, $\lambda = \{f_1, f_2, ..., f_\mathrm{K}\}$ such that $f_1 < f_2 < ... < f_\mathrm{K}$, where f_1 is the minimum/slowest frequency and f_K is the maximum/fastest frequency. For the sake of clarity, scheduling is described in terms of quanta of execution, where each quantum is the number of clock cycles sufficient to yield a reasonable unit of execution for scheduling (While the proper selection of a quantum size is an important aspect of scheduling, it is out of the scope of this work.). As such, each frequency, f_k, represents the number of quanta per second.

When a VFI is operating at frequency f_k, each core on the VFI has a corresponding total power consumption of w_k. As such, there is a set of K power consumption levels $W = \{w_1, w_2, ..., w_\mathrm{K}\}$, referred to herein as power states, that correspond with the set of frequencies in λ and $0 < w_1 < w_2 < ... < w_\mathrm{K}$. Note that we implicitly assume that a core is not permitted to operate at a frequency where leakage power dominates dynamic power. This assumption is realistic for many modern processors or can be applied by analysis [13].

Each task, τ_i, is defined by a worst-case execution time (WCET), C_i, a period, T_i, a relative deadline, D_i, a utilization, u_i, and a profile of execution, x_i. The WCET is the absolute maximum number of quanta required for the task to complete and includes scheduling overhead due to context switches, etc. In this way, the WCET is a fixed amount of work with a variable execution time based on the frequency of the core that it is being executed on. The period is the time, in seconds, for how often the task must execute. The deadlines of each task for this work are implicit, i.e., $T_i = D_i$; the task must complete before the start of the next period. The utilization, u_i, is the share of a core's utilization that the task must be allotted in order to meet its deadline and can be calculated by:

$$u_i = \frac{C_i}{f_\mathrm{K} \cdot T_i}. \tag{1}$$

Note that the utilization is calculated assuming the the maximum frequency, f_K.

The utilization of each task is restricted to $0 < u_i \leq 1$, i.e., a task must require a positive non-zero utilization, but no single task can require more that the total capacity of a single core. The profile of execution follows the probabilistic model proposed in [21] such that x_i, is a set of X_i potential execution times, c_α, with their probability of occurrence, ρ_α. As such, $x_i = \{(\rho_1, c_1), (\rho_2, c_2), ..., (\rho_{\mathrm{X}_i}, c_{\mathrm{X}_i})\}$, $\mathrm{X}_i > 0$, and $c_1 < c_2 < ... < c_{\mathrm{X}_i} = C_i$. Also,

$$\sum_{\alpha=1}^{\mathrm{X}_i} \rho_\alpha = 1. \tag{2}$$

Additionally, tasks in the task set are sorted by utilization such that:

$$u_1 \geq u_2 \geq ... \geq u_\mathrm{N}. \tag{3}$$

Since the task set, η, will be scheduled on the set of cores, P, the system utilization, U, is the sum of the utilization of the tasks and can be calculated as:

$$\mathrm{U} = \sum_{i=1}^{\mathrm{N}} u_i. \tag{4}$$

In order to guarantee deadlines with EDF, the cores cannot be overloaded. Hence, $\mathrm{U} \leq \mathrm{M}$ is a necessary condition for schedulability.

During partitioning, the tasks are partitioned to the cores such that the task set, η, is divided into task subsets where each subset η_j is assigned to core p_j. However, following the EDF-os [2] semi-partitioning heuristic, each task is assigned a share, $s_{i,j}$, on each core such that:

$$\sum_{j=1}^{\mathrm{M}} s_{i,j} = u_i. \tag{5}$$

Note that shares can be zero and some tasks may have only one non-zero share. Such tasks are referred to as fixed tasks. Conversely, tasks that have a non-zero share on more than one core are considered migrating tasks. As mentioned, migrations are boundary limited and determined by p-fair [7]. Thus, all jobs of a fixed task will execute on a single core, i.e., the fixed task do not migrate, while the jobs of a migrating task will migrate between cores with a non-zero share such that the number of jobs executed on each core is equal to the share assigned to that core. Therefore, the average utilization of each core U_j, is:

$$U_j = \sum_{i=1}^{\mathrm{N}} s_{i,j}. \tag{6}$$

For ease of discussion, η_j is the subset of tasks with a non-zero share on p_j, and N_j is the number of tasks in η_j.

4. IMBALANCE AND VFI COST

We now consider the imbalance and the VFI cost. As has been mentioned previously, imbalance occurs in pure partitioning heuristics when some cores in a multi-core system are assigned a greater workload than others. VFI cost is the increase in energy consumption that occurs when one or more cores on a VFI operate at a v/f pair above that which is necessary to guarantee deadlines because another core on the VFI needs the increased v/f pair to guarantee deadlines. In this section, we will formally define the imbalance and the VFI cost as well as present a best- and worst-case analysis of the VFI cost. We will conclude this section by showing simulation results approximating the average-case behavior of VFI cost and demonstrate the correlation between imbalance and VFI cost.

4.1 Imbalance

To achieve an ideal load-balance on the system, each core should be assigned the same utilization. The load-balanced utilization, Φ, that each core needs to be assigned can be calculated by:

$$\Phi = \frac{\mathrm{U}}{\mathrm{M}}. \tag{7}$$

Unfortunately, since the utilization of a task can be any value which is greater than zero but less than or equal to one, there is no guarantee that the tasks can be partitioned in such a way as to achieve a load-balance. Further, while the WFD heuristic does approach a load-balanced partition, and thus is often used for energy-efficiency, it is not guaranteed to find a load-balanced solution, even if one exists. Formally, the imbalance, ϕ, is the ratio by which the utilization of the cores differs from the ideal load-balance:

$$\phi = \frac{\sum_{j=1}^{\mathrm{M}} \frac{|\Phi - U_j|}{\Phi}}{\mathrm{M}} = \frac{\sum_{j=1}^{\mathrm{M}} |\Phi - U_j|}{\mathrm{U}}. \tag{8}$$

4.2 VFI Cost

The VFI cost is the ratio of energy consumed as a result of one or more cores operating at a v/f pair that is higher than is necessary to the total energy. (Recall the VFI must operate at the highest v/f pair required by any of the cores on it.) More formally, for a period of T quanta, $[0, \mathrm{T})$, on a given VFI, γ_l, for each quantum, t, each core in P_l will calculate an optimal operating frequency using some DVFS-based scheduling algorithm. Next, γ_l will set its frequency to the maximum of the frequencies calculated by the cores in P_l. Let the optimal frequency, f_o, calculated by each core at quantum t be denoted by $f_o(j,t)$, with the corresponding power state given by $w_o(j,t)$. In addition, let the frequency set by γ_l be $f_{\mathrm{VFI}}(l,t)$, with the corresponding power state given by $w_{\mathrm{VFI}}(l,t)$. The VFI cost, Ψ, for a single VFI for a single quantum, denoted as $\Psi_l(t)$, can be calculated as:

$$\Psi_l(t) = \frac{\sum_{\forall p_j \in P_l} [w_{\mathrm{VFI}}(l,t) - w_o(j,t)]}{\mathrm{M}_l \cdot w_{\mathrm{VFI}}(l,t)}. \tag{9}$$

Since the VFI cost of the system is simply the average of all VFI in the system over all of the quanta, the total system VFI cost can be calculated by:

$$\Psi = (\mathrm{I} \cdot \mathrm{T})^{-1} \sum_{l=1}^{\mathrm{I}} \sum_{t=0}^{\mathrm{T}-1} \frac{\sum_{\forall p_j \in P_l} [w_{\mathrm{VFI}}(l,t) - w_o(j,t)]}{\mathrm{M}_l \cdot w_{\mathrm{VFI}}(l,t)}. \tag{10}$$

Now, since the value of $w_{\mathrm{VFI}}(l,t)$ and $w_o(j,t)$ for any value of t are in the power state set W and $w_{\mathrm{VFI}}(l,t) \geq w_o(j,t)$, then: $w_{\mathrm{VFI}}(l,t) > w_{\mathrm{VFI}}(l,t) - w_o(j,t)$. Thus,

$$\mathrm{M}_l \cdot w_{\mathrm{VFI}}(l,t) > \sum_{\forall p_j \in P_l} [w_{\mathrm{VFI}}(l,t) - w_o(j,t)]. \tag{11}$$

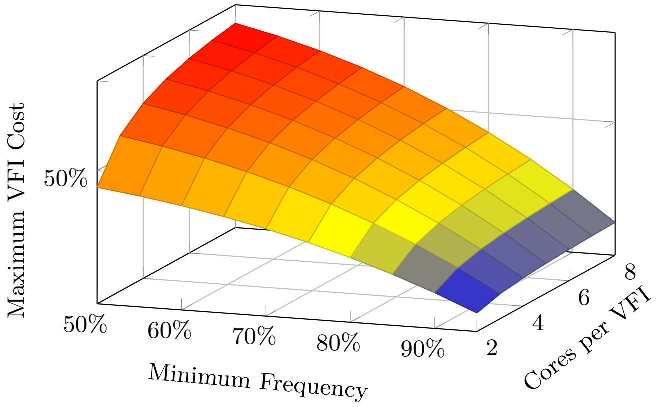


Figure 2: The maximum VFI cost (Equation 13) as a function of the number of cores (M) by the minimum frequency (f_1) that the VFI can reduce to (as a percentage of the maximum frequency, f_{K}) assuming a pure CMOS circuit (i.e., $w_k \propto f_k^3$).

As a result, the VFI cost is strictly less than one. Additionally, from Equation 10, it is possible to determine worst- and best-case VFI cost, as described next.

4.2.1 Worst-Case VFI Cost

The maximum VFI cost will occur when the difference between the set VFI power state $w_{\mathrm{VFI}}(l,t)$ and the cores power state $w_o(j,t)$ is largest for all of the cores on the VFI for all quanta. The maximum possible value of $w_{\mathrm{VFI}}(l,t)$ is w_{K}, and the minimum possible value of $w_o(j,t)$ is w_1, however, $w_{\mathrm{VFI}}(l,t)$ can only be w_{K} if the value of $w_o(j,t) = w_{\mathrm{K}}$ for at least one core on the VFI. Thus, the maximum VFI cost for a single VFI at a single quantum t, denoted as $\Psi_{l_{max}}(t)$, will occur when $w_o(j,t) = w_{\mathrm{K}}$ for only one core and $w_o(j,t) = w_1$ for all other cores. Therefore, the maximum VFI cost for a single quantum is:

$$\Psi_{l_{max}}(t) = \frac{\sum_{j=1}^{\mathrm{M}_l - 1}[w_{\mathrm{K}} - w_1]}{\mathrm{M}_l \cdot w_{\mathrm{K}}} = \frac{(\mathrm{M}_l - 1)(w_{\mathrm{K}} - w_1)}{\mathrm{M}_l \cdot w_{\mathrm{K}}}$$

$$\Psi_{l_{max}}(t) = 1 - \frac{w_{\mathrm{K}} + (\mathrm{M}_l - 1)w_1}{\mathrm{M}_l \cdot w_{\mathrm{K}}}. \tag{12}$$

It follows then that the maximum VFI cost of the system, Ψ_{max}, will be achieved when all of the VFI experience the maximum VFI cost for each quantum in the considered time interval, i.e. [0,T). Therefore, substituting Equation 12 into Equation 10 produces:

$$\Psi_{max} = (\mathrm{I} \cdot \mathrm{T})^{-1} \sum_{l=1}^{\mathrm{I}} \sum_{t=0}^{\mathrm{T}-1} \left(1 - \frac{w_{\mathrm{K}} + (\mathrm{M}_l - 1)w_1}{\mathrm{M}_l \cdot w_{\mathrm{K}}}\right)$$

$$\Psi_{max} = (\mathrm{I} \cdot \mathrm{T})^{-1} \left(1 - \frac{w_{\mathrm{K}} + (\mathrm{M}_l - 1)w_1}{\mathrm{M}_l \cdot w_{\mathrm{K}}}\right) \sum_{l=1}^{\mathrm{I}} \sum_{t=0}^{\mathrm{T}-1} 1$$

$$\Psi_{max} = 1 - \frac{w_{\mathrm{K}} + (\mathrm{M}_l - 1)w_1}{\mathrm{M}_l w_{\mathrm{K}}}. \tag{13}$$

Thus, the maximum VFI cost of the system is the same as the maximum VFI cost of a single quantum. The maximum VFI cost is shown in Figure 2 for two to eight cores per VFI, assuming a pure CMOS circuit, i.e., $w_k \propto f_k^3$, for systems that can reduce the frequency from 95% of the maximum to 50% of the maximum.

4.2.2 *Best-Case VFI Cost*

Conversely, to achieve the minimum VFI cost, Ψ_{min}, the difference between $w_{\text{VFI}}(l,t)$ and $w_o(j,t)$ must be minimized for all of the cores on each VFI for all quanta. This will occur when $w_{\text{VFI}}(l,t) = w_o(j,t)$. Thus, from Equation 10, when $w_{\text{VFI}}(l,t) = w_o(j,t)$, the VFI cost is zero. Therefore, it is possible for there to be no VFI cost regardless of whether the system is operating at its maximum, minimum, or any other power state.

4.3 Simulation Results

While we have described the best- and worst-case VFI cost, the best- and worst-case scenarios rarely occur in practice. Hence, we performed simulations of randomly generated task sets for several different utilizations to assess the average VFI cost. We will explain our simulation setup first and then show the simulation results of the average-case behavior and demonstrate the correlation between VFI cost and imbalance.

4.3.1 *Simulation Setup*

To minimize biasing in the randomly generated task sets, a slightly modified version of the UUniFast algorithm in [5] was used. Since the UUniFast algorithm assumes a single core system, it needed to be modified so that the target utilization can exceed 100% while restricting the individual task utilization to 100%. Due to this modification, the UUniFast algorithm occasionally failed to produce a valid task set. When this situation arose, the task set was discarded, and a new task set was randomly generated. In addition, not every valid task set can be partitioned unless the system utilization is less than approximately 50% [6]. If a task set was encountered that could not be partitioned, the task set was also discarded and another task set was generated randomly.

The maximum allowable period for a task was set to 100 quanta with a minimum allowable period of 5 quanta. Additionally, each task was required to have a minimum WCET of one quantum. For implementation purposes, the system utilization was allowed to deviate from the desired utilization by up to 1%. An execution profile was then generated for each task with a random number of probability and execution time entries (see Section 3). The probabilities for each execution profile were generated using the original UUniFast algorithm. The potential execution times were similarly generated using a modified version of UUniFast where, for each iteration, the actual execution time was taken as some percentage of the WCET.

In addition to generating tasks, we also generated the available VFI frequencies and corresponding power states. Since the VFI cost is a ratio between the power states, the values of the frequencies and power states themselves are not as important as the ratio between the frequencies and power states. For this set of simulations, cores in a VFI could reduce their frequency to up to 70% of the maximum frequency at 5% granularity. The corresponding power was calculated as $w_k \propto f_k^3$ (we assume leakage current is constant and thus the power states reflect the dynamic power cost).

For a given number of tasks, number of cores per VFI, and system utilization, 20 task sets were randomly generated. Since each task has a random execution profile, the task sets were simulated three times to obtain meaningful data. Thus, the VFI cost for the given number of tasks, number of cores per VFI, and system utilization is the average of the VFI cost of all 60 simulations, i.e., 20 task sets each simulated 3 times.

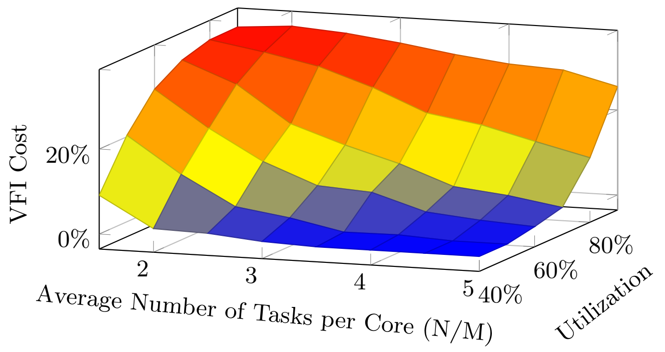
Simulations were run for systems configured with two, four, and eight cores per VFI. Each set of simulations only considered one VFI per system, since, as shown in Equation 10, the VFI cost of a system is simply the average of the VFI cost of each VFI. Thus, since the simulation is already the average of multiple simulations, the results of M cores on a single VFI are representative of kM cores on k VFIs. Further, to be able to compare the results of the VFI with two, four, and eight cores per VFI, values of N were selected such that the average number of tasks per core, i.e, $\frac{\text{N}}{\text{M}}$, varied from 1.5 to 5 (at intervals of 0.5). Finally, task sets with total utilization between 40% and 90% (at intervals of 10%) were considered. The minimum utilization of 40% was selected since each core can reduce its frequency to 70%, implying that, for a total utilization at or below 30%, the VFI cost is negligible. Conversely, the maximum utilization was set at 90% since partitioning task set with a 100% total utilization is only feasible in rare cases. Once partitioned, tasks are scheduled with EDF and the frequency for each core is calculated using the Look-Ahead Dynamic Voltage Scaling (LADVS) algorithm [19].

4.3.2 *Results*

The combined average VFI cost and imbalance for VFI configurations of two, four, and eight cores per VFI are shown in Figure 3 (the individual results for each configuration are omitted due to space constraints but are available in [18]). Since these graphs are produced from a set of randomly generated data, they do not represent exact values, but can be used to establish general trends. As shown in the graphs, as the utilization increases, the VFI cost also increases. In addition, as the average number of tasks per core increases, both the imbalance and VFI cost are reduced.

Another trend shown in Figure 3 is that the VFI cost increases as the number of cores per VFI increases. This behavior follows the mathematical model in Equation 12. The mathematical limit for two, four, and eight cores per VFI are 33.3%, 50.0%, and 58.3%, respectively. The maximum simulated VFI cost for each configuration are 23.9%, 37.3%, and 48.6%, respectively. While the simulated maximum VFI cost for all three VFI configurations occurred at a high utilization with a low average number of tasks per core, the simulated results attained an average of 76.6% of the mathematical limit with the configurations with more cores per VFI being closer to the mathematical limit in all three cases. Further, the maximum simulated VFI cost for all three configurations occurred when the imbalance was at its highest for the given utilization.

As shown in Figure 3, as the average number of tasks per core increases, the imbalance decreases. For the lower average number of tasks per core, as the utilization increases, the imbalance decreases. This is expected since as the utilization increases, the probability that the task set is not able to be partitioned also increases. Since any randomly generated task set that could not be partitioned was not considered in



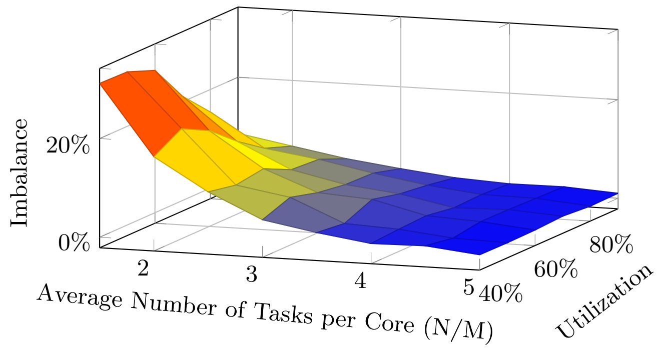


Figure 3: Combined average VFI cost (top) and imbalance (bottom) (i.e., combined average of two, four, and eight cores per VFI), as a function of the average number of tasks per core (i.e., N/M) and system utilization.

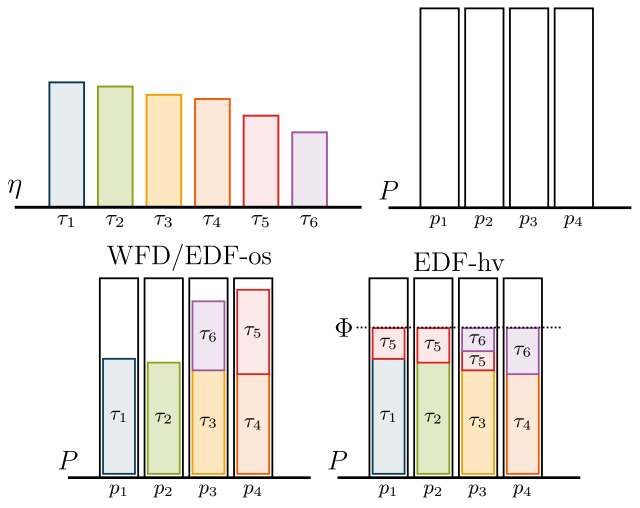


Figure 4: Partitioning the task set, η (top left), where each task is represented by its utilization, onto the set of cores, P (top right), using the WFD (or EDF-os; bottom left) and EDF-hv (bottom right). EDF-hv restricts the available capacity of each core to Φ.

this work, the amount of imbalance decreases as necessary to be feasible for partitioning.

The final important trend shown in Figure 3 is that although the VFI cost is clearly dominated by system utilization, there is a strong correlation between the VFI cost and imbalance. The average correlation of the reduction in imbalance to VFI cost for the different system utilizations across all three VFI configurations was nearly one to one. As such, although the VFI cost is primarily influenced by the utilization, our data suggests that it is also correlated to imbalance and thus can be partially mitigated by load-balancing the system.

5. EDF-HV

EDF-hv modifies the partitioning process of EDF-os to load-balance a system instead of guaranteeing feasibility up to 100% of the system's capacity. With this modification, EDF-hv then extends the scheduling procedure of EDF-os to guarantee deadlines instead of providing tardiness bounds. In this section, we first present the partitioning process of EDF-hv. We then describe the modifications to the task scheduling process. We end this section with some important properties of EDF-hv and provide a proof that EDF-hv can guarantee hard real-time deadlines.

5.1 Partitioning for a Load-Balance

EDF-hv follows the partitioning process of EDF-os with two main modifications in order to achieve a load-balance. First, we restrict the available capacity of each core to be no more than Φ. (Note that this assumption is acceptable for under-loaded systems where there are energy saving opportunities. For fully-loaded or near fully-loaded systems, it is unlikely that applying a PM scheme will yield significant energy savings. Thus those systems are not the focus of this work.) Second, since no task can have a utilization greater than that of a core's capacity in order to guarantee feasibility, it is also necessary to restrict the utilization of each task such that $u_i \leq \Phi$ (considerations for task sets with tasks that have a $u_i > \Phi$, referred to as oversized tasks, are presented in Appendix A of [18]). An example comparing the partition of EDF-hv to WFD and EDF-os is shown in Figure 4. In this example, there are six tasks to be partitioned onto four cores. With WFD, one task is assigned to each core, until all of the cores have one task. Next, the remaining two tasks are assigned to the cores with the lowest utilization. Since all of the tasks can fit onto the cores, the second step of the partitioning process is not initiated, and, thus, the partition is the same for EDF-os as it is for WFD (recall that EDF-os is designed to increase feasibility, not to improve the load-balance). In contrast, with EDF-hv, the capacity of each core is restricted to Φ such that, after the initial four tasks have been assigned, the remaining two tasks will no longer fit into the remaining available capacity of each core and thus, the remaining two tasks are assigned as migrating tasks with shares on multiple cores.

Since EDF-hv follows the partitioning process of EDF-os, EDF-hv can now guarantee that the resultant partition to be load-balanced; as stated by the following theorem.

THEOREM 1. *Given a task set η where $u_i \leq \Phi$, $i = 1, ..., \mathrm{N}$, and a set of* M *cores, EDF-hv partitions the tasks among the cores in such a way as to achieve a load-balance.*

PROOF. As previously mentioned, Anderson et al. showed that the partitioning process of EDF-os can guarantee feasibility with a system utilization bound of 100% while allowing individual task utilization of up to 100% of a core's capacity [2]. As such, for a system where U = M, EDF-os guarantees that the system can be partitioned. Interestingly, in this case, EDF-os also load-balances the system since each core must be assigned 100% of its utilization, otherwise, $\sum_{j=1}^{\mathrm{M}} U_j < \mathrm{U} = \mathrm{M}$, which cannot be true. The partitioning process of EDF-hv is identical to that of EDF-os, with the exception of the two modifications. These modifications effectively induce the case where the task set utilization is equal to the available system capacity, which EDF-os has already been shown to guarantee feasibility. Therefore, EDF-hv is guaranteed to be load-balanced. □

CAVEAT 1. *Restricting the utilization of each task to $u_i \leq \Phi$ yields a circular definition that can cause an invalid state. Consider Equation 7 and Equation 4:*

$$\Phi = \frac{\mathrm{U}}{\mathrm{M}} = \frac{\sum_{i=1}^{\mathrm{N}} u_i}{\mathrm{M}}.$$

Thus, if $u_i \leq \Phi$, then:

$$u_i \leq \frac{\sum_{i=1}^{\mathrm{N}} u_i}{\mathrm{M}}.$$

Considering the case of τ_1:

$$u_1 \leq \frac{\sum_{i=1}^{\mathrm{N}} u_i}{\mathrm{M}},$$

which yields:

$$\mathrm{M} \cdot u_1 \leq u_1 + u_2 + ... + u_{\mathrm{N}}. \qquad (14)$$

Recall that tasks are sorted in a non-increasing order of utilization, i.e., the first task has the largest utilization, and thus, Equation 14 yields that M *times the largest utilization must be less than or equal to the sum of* N *tasks. As such, Equation 14 is only true if* $\mathrm{N} > \mathrm{M}$*, except in the case that* $\mathrm{N} = \mathrm{M}$ *and* $u_1 = u_2 = ... = u_{\mathrm{N}}$*. Since assuming all tasks have the same utilization is not very practical, EDF-hv therefore requires that* $\mathrm{N} > \mathrm{M}$.

LEMMA 1. *Each core, p_j, has at least one fixed task.*

PROOF. In EDF-hv, $\mathrm{N} > \mathrm{M}$, $\Phi = \frac{\mathrm{U}}{\mathrm{M}}$, and tasks are first partitioned using WFD. Under the WFD heuristic, a task will be assigned to an empty core before a core will be assigned a second task. Further, since all tasks have a utilization, $u_i \leq \Phi$, it is guaranteed that at least one task can fit on each core. Finally, since there are more tasks than cores (i.e., $\mathrm{N} > \mathrm{M}$), each core must be assigned at least one fixed task. □

LEMMA 2. *Each core, p_j, has a maximum of two migrating tasks.*

PROOF. Following the proof for Property 3 of EDF-os in [2], it can be shown by induction that during the partitioning process of EDF-hv, when assigning a migrating task to a core, there can be at most one migrating task already assigned. □

5.2 Scheduling

The reason that tardiness bounds can be derived in EDF-os, but deadlines cannot be guaranteed is that cores with migrating tasks can be overloaded when migrating tasks are executing. As an example, consider the periodic task set for a multi-core system with two cores, as shown in Table 1. For simplicity, task periods are given in quanta to avoid having to consider the frequency of the cores. The resultant EDF-os partition is shown on the right side of Table 1. In this system, τ_1 and τ_2 are fixed tasks of p_1 and p_2, respectively, while τ_3 is a migrating task on both p_1 and p_2. Based on the shares of τ_3 on p_1 and p_2, of every eleven jobs of τ_3, five will execute on processor p_1, while six will execute on p_2. Although the assigned utilization of p_1 and p_2 is 100% each, p_1 and p_2 are overloaded when they have to execute both their fixed and migrating tasks. Since all three tasks have the same period, it is easy to see that during periods where p_1 has only its fixed task, τ_1, to execute, it only needs 15 of the 20 quanta in that period, i.e., the utilization during this period is only $\frac{15}{20} = 75\%$. However, during periods where p_1 has its fixed task, τ_1, and its migrating task, τ_3, to execute, it must execute 15 + 11 = 26 cycles of 20, i.e., the utilization during this period is $\frac{15+11}{20} = \frac{26}{20} = 130\%$. Therefore, p_1 is overloaded (it is trivial to show that p_2 is similarly overloaded). As such, in EDF-os, since the cores can be overloaded when they have both their fixed and migrating tasks, deadlines cannot be guaranteed. However, since the cores are not fully utilized when they do not have their migrating tasks, the cores can "catch-up", thus, the tardiness of deadlines can be bounded.

Task Set				Partition		
Task	C_i	T_i (Quanta)	u_i	Task	p_1	p_2
τ_1	15	20	75%	τ_1	75%	0%
τ_2	14	20	70%	τ_2	0%	70%
τ_3	11	20	55%	τ_3	25%	30%

Table 1: Example periodic task set (left) with EDF-os partition onto a system with two cores (right).

Therefore, for EDF-hv to guarantee deadlines, the cores must not be overloaded. Anderson et. al. in [2] propose that EDF-os can be suitable for HRTS by increasing the capacity of the system, e.g. increasing the number of cores, using cores with higher clock speed, etc., until the tardiness bounds are zero. However, this approach is not ideal as the tardiness bounds in EDF-os are not dependent on the task utilizations due to the prioritization of the migrating tasks. For example, let us reconsider the task set in Table 1. For p_1 to not be overloaded, during a period where both τ_1 and τ_3 are on p_1, since τ_3 is a migrating task, it must complete τ_3 first, and then τ_1 can execute. As such, the capacity of p_1 must increase to $\frac{11+15}{20} = 130\%$. However, if the WCET and period of τ_3 are increase to 22 and 40, respectively, even though the utilization of the system remains constant, for periods when both τ_1 and τ_3 are on p_1, the capacity must increase to $\frac{22+15}{20} = 185\%$. As such, prioritizing migrating tasks over fixed task yields that guaranteeing deadlines is no longer solely a function of the utilization.

COROLLARY 1. *In the presence of migrating tasks, EDF-os cannot guarantee a tardiness bound of zero by only considering task utilizations.*

As a result, in EDF-hv, all tasks are scheduled with EDF, i.e., migrating tasks are not prioritized over fixed tasks. Clearly, by changing the prioritization of the migrating and fixed tasks, the tardiness bounds in [2] for EDF-os are no longer valid. However, as will be proved next, tardiness bounds are not necessary for EDF-hv since deadlines can be guaranteed.

5.3 Guaranteeing Deadlines

It is well-known that, even with all tasks being scheduled with EDF, task deadlines cannot be guaranteed when a core is overloaded. In the previous section, it was shown that EDF-hv can load-balance the task set on the cores. However, this is insufficient to guarantee deadlines, because the utilization considered during the partitioning process, shown in Equation 6, is the average utilization of each core. As has been discussed, the actual core utilization fluctuates depending on whether the migrating task(s) are present. Thus, to guaranteed deadlines, the core must not be overloaded even when experiencing the maximum utilization.

To determine the maximum utilization that a core can experience, recall that a fixed task has a non-zero share equal to the task utilization on only one core, and migrating tasks have more than one non-zero share less than the task's utilization on the cores. However, during execution, since migrations are boundary limited, each core will have to be able to execute the complete utilization of each task assigned to it. As such, for cores with only fixed tasks, the maximum utilization is simply the sum of the shares assigned to that core as given by Equation 6. For cores with one migrating task, the maximum utilization is the sum of the shares of the fixed tasks, plus the utilization of the migrating task. Finally, for cores with two migrating tasks, the maximum utilization is, in the best-case, the sum of the shares of the fixed tasks plus the utilization of the first migrating task (since the first migrating task must have a utilization greater than or equal to the second migrating task by Equation 3), or, in the worst-case, the sum of the shares of the fixed tasks plus the utilization of both migrating tasks. The best-case for cores with two migrating tasks occurs when the migrating tasks have migration patterns that ensure only one migrating task at a time will be present on the core; otherwise, the worst-case will occur. Unfortunately, unless the two migrating tasks have harmonic periods and both tasks have harmonic shares on all of the cores with non-zero shares, a time-demand analysis based calculation is required to determine which case will occur. Thus, to simplify calculation, the worst-case is used. Therefore, the maximum utilization that a core can experience is:

$$U_{j_{max}} = \sum_{i=1}^{\mathrm{N}_j} u_i,\ \forall \tau_i \in \eta_j. \tag{15}$$

For cores with only fixed tasks or cores with only one migrating task, Equation 15 is the maximum utilization that a core can experience, however, for cores with two migrating tasks, Equation 15 is an upper bound on the maximum utilization. Further, the maximum utilization may only occur rarely (e.g., if a core with a single migrating task is assigned only a hundredth share of the migrating task's utilization, the maximum utilization will only occur one out of a hundred iterations of the task), yet it is the upper limit on the utilization that the core can experience. As such, even if the maximum utilization only occurs rarely, to guarantee deadlines under all conditions, the maximum utilization must be less than or equal to one.

THEOREM 2. *If $U_{j_{max}} \leq 1$, all task deadlines are guaranteed under EDF-hv.*

PROOF. EDF is known to be able to guarantee deadlines on a core with a periodic task set with implicit deadlines as long as the utilization of that processor is less than or equal to one [15]. EDF-hv also assumes a periodic task set but allowing tasks to migrate at job boundaries complicates the behavior of the task set. While cores with only fixed tasks have a periodic task set with implicit deadlines, meaning deadlines can be guaranteed by the properties of EDF, for a core with both fixed and migrating tasks, some share of the time the migrating tasks will not be executing jobs on the core. However, when the migrating tasks are executing on the cores, the jobs are released at period boundaries and have implicit deadlines. In fact, the task will still have all the properties of a periodic task with implicit deadlines, with the exception that for some periods, the execution time will be zero. Thus the total utilization of a core with migrating tasks will always be less than or equal to the maximum utilization. Therefore, since the maximum utilization of all fixed and migrating tasks is less than or equal to one, deadlines can be guaranteed by EDF. □

Therefore, in EDF-hv, guaranteeing deadlines is dependent on the maximum utilization that any of the cores in the system can experience, which is appropriate for run-time use.

6. SIMULATION RESULTS

EDF-hv was compared to WFD for systems with two, four, and eight cores per VFI to assess its performance. Although it may seem that EDF-hv should be compared to EDF-os, since the former is heavily based on the latter, a fair comparison cannot be made, as EDF-os is intended for soft real-time systems. We describe our setup and explain the results next.

6.1 Simulation Setup

The simulation setup for EDF-hv is similar to that for VFI cost with four exceptions. First, to establish a baseline for EDF-hv, task execution profiles were not considered, i.e., each task was assumed to require its WCET. Second, instead of simulating 20 task sets at each system utilization, 100 task sets were simulated at each utilization. Since task execution profiles were not considered, each task set was simulated using EDF-hv and WFD. The performance of EDF-hv was then assessed by comparing the ratio of the total dynamic energy consumption of EDF-hv to WFD (e.g. a performance of 80% means that EDF-hv required only 80% of the energy that WFD required to execute the same task set). Third, the frequency profile was set such that the VFI could reduce their frequency to 60% of the maximum frequency at 1% granularity. This frequency profile follows the work in [4], which showed that the Intel Atom N2600 dual core processor has 22 v/f pairs when reducing the voltage to approximately 62% of the maximum. Since the number of v/f pairs can vary significantly between hardware, for this work we simulate a multi-core with a high number of v/f pairs to reduce the effects of the frequency profile on the results. Fourth, task sets with system utilizations from 45% to 85% were considered at 10% intervals. The maximum total utilization was reduced to 85%, as only task sets that could meet Theorem 2 were considered, which was not typical of task sets with utilization above 85%. Note that considerations for frequency profiles with a restricted number of v/f pairs, system utilizations at 5% intervals, and systems with eight cores per VFI were omitted due to space constraints,

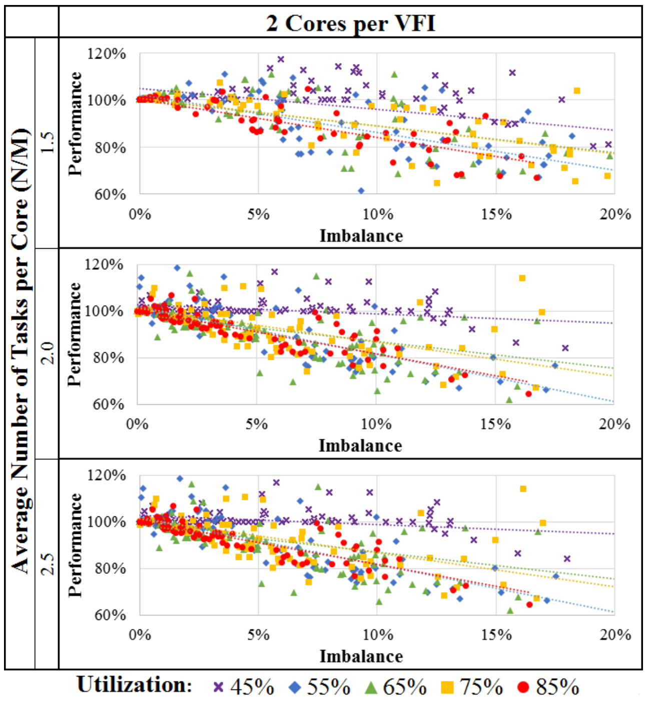


Figure 5: Performance of EDF-hv versus the imbalance of WFD on a two-core systems with one VFI for task sets with 1.5, 2.0, and 2.5 average number of tasks per core (i.e., N/M).

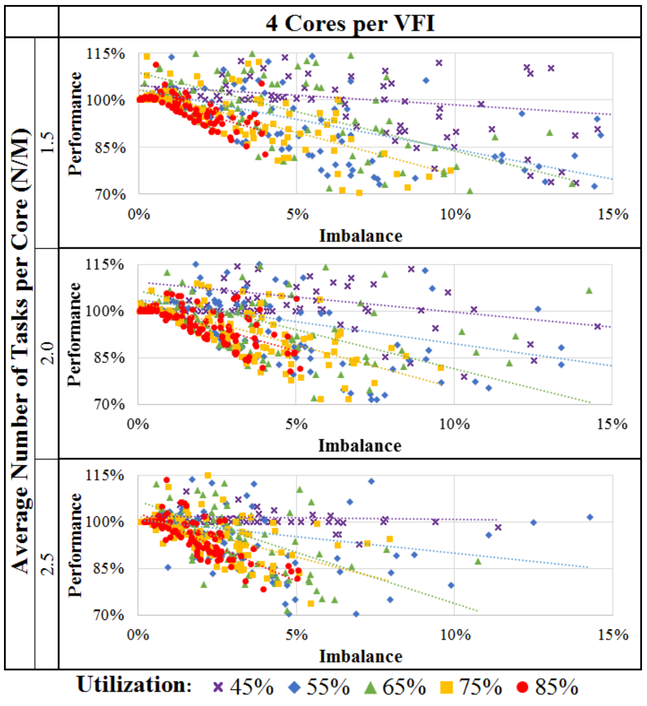


Figure 6: Performance of EDF-hv versus the imbalance of WFD on a four-core systems with one VFI for task sets with 1.5, 2.0, and 2.5 average number of tasks per core (i.e., N/M).

but are available in [18].

6.2 Performance of EDF-hv

The performance of EDF-hv as compared to WFD are shown in Figure 5 and Figure 6. Figure 5 shows the performance for a system with two cores per VFI and Figure 6 shows the performance for a system with four cores per VFI. Both Figure 5 and Figure 6 are comprised of three graphs with each graph expressing the results for a given average number of tasks per core. Further, each graph shows a set of scatter plots for the considered system utilizations where the individual points in the scatter plot represents a task set. The performance is shown versus the imbalance because the primary way that EDF-hv conserves energy is by load-balancing the system and thus, it is expected that EDF-hv will typically only improve the energy consumption of task sets with some imbalance. Trend lines for each system utilization are shown for reference.

For utilizations less than 55%, EDF-hv performed worse on average than WFD, which was expected since the utilizations of these systems is lower than the VFI could reduce their frequency. However, for utilizations 55% and above, EDF-hv consumed 6.3% less energy than WFD. Further, as shown in Figure 5 and Figure 6, there is a negative correlation between the amount of imbalance in a task set and the performance of EDF-hv. Additionally, as the average number of tasks per core increased, so did the negative correlation between imbalance and performance. This demonstrates that for systems with imbalance, EDF-hv typically performed better that WFD.

On average EDF-hv consumed 4.8% less energy than WFD, however, for systems with a utilization of 55% or greater and an imbalance of 5% or greater, the average energy consumption was reduced by 15.9%. As such, as shown in Figure 7, while applying EDF-hv generally can improve overall energy efficiency, it is much better to only apply EDF-hv to systems with some imbalance and a system utilization approximately greater than or equal to the percentage of the maximum frequency that the VFI can reduce its frequency to (i.e. $U \gtrsim 1 - \frac{f_1}{f_K}$). Fortunately, calculating the imbalance of the system and the system utilization have a computational complexity of N (i.e., O(N)), so determining systems that are appropriate for EDF-hv can be accomplished quickly.

7. CONCLUSION AND FUTURE WORK

The best- and worst-case VFI cost have been derived, and both the mathematical model and simulation results suggest that the VFI cost can be a significant contributor to the overall energy consumption. The proposed algorithm, EDF-hv, can help to mitigate VFI cost in HRTS with multiple cores per VFI. Our results also show that the energy savings from EDF-hv can be significant, especially for systems with significant imbalance and high utilizations.

Several aspects of this work can be further explored to improve energy efficiency. First, a task scheduling algorithm as well as a DVFS algorithm that is explicitly designed to accommodate migrating tasks can be developed. Second, further exploration into the relationship between VFI cost and imbalance may provide insight into ways to improve EDF-hv. A third area of research is to remove the boundary limited property of EDF-hv. This would increase the scheduling

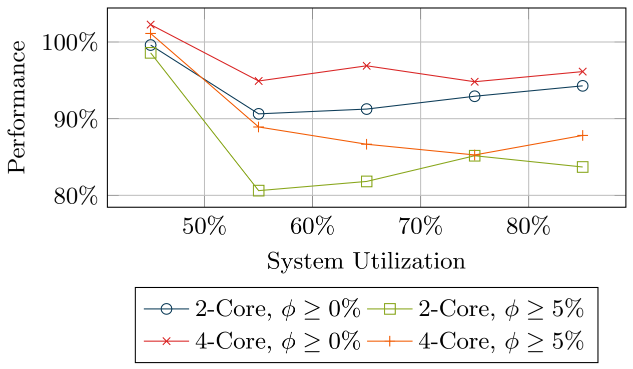


Figure 7: Performance of EDF-hv compared to WFD with two-cores and four-cores per VFI when applied generally (i.e. $\phi \geq 0\%$) and when applied only to systems with at least 5% of imbalance.

overhead, which would increase energy consumption, however, removing the boundary limited property could yield energy saving to justify the increased overhead. Finally, in addition to saving energy, load-balancing with EDF-hv may have some advantages for slowing down aging.

8. ACKNOWLEDGMENTS

We gratefully acknowledge funding and other support from the Naval Air Warfare Center Weapons Division for this work. This research was also supported in part by NSF under grant number CNS-1618979.

9. REFERENCES

[1] J. H. Anderson, V. Bud, and U. C. Devi. An EDF-based restricted-migration scheduling algorithm for multiprocessor soft real-time systems. *Real-Time Systems*, 38:85–131, 2008.

[2] J. H. Anderson, J. P. Erickson, U. C. Devi, and B. N. Casses. Optimal semi-partitioned scheduling in soft real-time systems. In *International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 1–10, Aug. 2014.

[3] B. Andersson and E. Tovar. Multiprocessor scheduling with few preemptions. In *International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)*, pages 322–334, Aug. 2006.

[4] S. Arunachalam. An online wear state monitoring methodology for off-the-shelf embedded processors. Master's thesis, Utah State University, Logan, UT, 2015.

[5] E. Bini and G. C. Buttazzo. Biasing effects in schedulability measures. In *Proceedings of Euromicro Conference on Real-Time Systems*, pages 196–203, June 2004.

[6] J. Carpenter, S. Funk, P. Holman, A. Srinivasan, J. H. Anderson, and S. Baruah. A categorization of real-time multiprocessor scheduling problems and algorithms. *Handbook on scheduling algorithms, methods, and models*, pages 30.1–30.19, 2004.

[7] J. Carpenter, S. Funk, P. Holman, A. Srinivasan, J. H. Anderson, and S. Baruah. Fair multiprocessor scheduling. *Handbook on scheduling algorithms, methods, and models*, pages 31.1–31.21, 2004.

[8] P. Choudhary and D. Marculescu. Power management of voltage/frequency island-based systems using hardware-based methods. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 17:427–438, 2009.

[9] R. I. Davis and A. Burns. A survey of hard real-time scheduling for multiprocessor systems. *ACM Comp. Surv.*, 43:35:1–35:44, 2011.

[10] F. Dorin, P. M. Yomsi, J. Goossens, and P. Richard. Semi-partitioned hard real-time scheduling with restricted migrations upon identical multiprocessor platforms. *Computing Research Repository (CoRR)*, 1006.2637, 2010.

[11] P.-E. Gaillardon, E. Beigne, S. Lesecq, and G. D. Micheli. A survey on low-power techniques with emerging technologies: from devices to systems. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 12:12.1–12.26, 2015.

[12] W. Jang, D. Ding, and D. Z. Pan. A voltage-frequency island aware energy optimization framework for networks-on-chip. In *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 264–269, Nov. 2008.

[13] R. Jejurikar, C. Pereira, and R. Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proceedings of the 41st Annual Design Automation Conference*, pages 275–280, 2004.

[14] S. Kato, N. Yamasaki, and Y. Ishikawa. Semi-partitioned scheduling of sporadic task systems on multiprocessors. In *21st Euromicro Conference on Real-Time Systems*, pages 249–258, July 2009.

[15] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *JAMC*, 20:46–61, 1973.

[16] S. Mittal. A survey of techniques for improving energy efficiency in embedded computing systems. *International Journal of Computer Aided Engineering and Technology*, 6:440–459, 2014.

[17] U. Y. Ogras, R. Marculescu, D. Marculescu, and E. G. Jung. Design and management of voltage-frequency island partitioned networks-on-chip. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 17, 2009.

[18] J. Patterson. An energy-efficient semi-partitioned approach for hard real-time systems with voltage and frequency islands. Master's thesis, Utah State University, Logan, UT, 2016.

[19] P. Pillai and K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, pages 89–102, Dec. 2001.

[20] E. L. Sueur and G. Heiser. Dynamic voltage and frequency scaling: The laws of diminishing returns. In *Proceedings of the 2010 international conference on Power aware computing and systems*, pages 1–8, Oct. 2010.

[21] T. Zitterell and C. Scholl. A probabilistic and energy-efficient scheduling approach for online application in real-time systems. In *Proceedings of the 47th Design Automation Conference*, pages 42–47, June 2010.