A Study of Utilization Bound and Run-Time Overhead for Cluster Scheduling in Multiprocessor Real-Time Systems*

Xuan Qi, Dakai Zhu
Department of Computer Science
University of Texas at San Antonio
San Antonio, TX, 78249
{xqi,dzhu}@cs.utsa.edu

Hakan Aydin
Department of Computer Science
George Mason University
Fairfax, VA 22030
aydin@cs.gmu.edu

Abstract

Cluster scheduling, where processors are grouped into clusters and the tasks that are allocated to one cluster are scheduled by a global scheduler, has attracted attention in multiprocessor real-time systems research recently. In this paper, by adopting optimal global schedulers within each cluster, first we investigate the worst-case utilization bound for cluster scheduling. Specifically, for a system with m homogeneous clusters where each cluster has k processors, we show that the worst-case achievable system utilization is $\frac{\lfloor k/\alpha \rfloor \cdot m+1}{\lfloor k/\alpha \rfloor +1} \cdot k$, where α is the maximum utilization for the periodic tasks considered. By focusing on an efficient optimal global scheduler, namely the boundary-fair (Bfair) *algorithm, we propose a* period-aware *partitioning heuristic aiming at reducing the scheduling overhead. Simulation results show that the percentage of task sets that can be scheduled is significantly improved under cluster scheduling even for small-size clusters (e.g., $k = 2$). Moreover, the proposed period-aware partitioning heuristic markedly reduces the scheduling overhead of cluster scheduling with Bfair.*

1 Introduction

The problem of scheduling periodic real-time tasks upon multiprocessor platforms has been studied for decades [12, 13] and there is a reviving interest due to the emergence of multicore processors [2, 5, 11, 15, 16, 18, 19]. In this line of research, the main objective is to find efficient scheduling algorithms that can effectively utilize the available processors, with low run-time overhead. In general, there are two main approaches to the scheduling problem in multiprocessor real-time systems: *partitioned* and *global* scheduling [12, 13].

*This work was supported in part by NSF awards CNS-0720651, CNS-0855247, CNS-0720647 and CAREER Award CNS-0546244.

In partitioned scheduling, each task is permanently assigned to a specific processor. Although well-established uniprocessor scheduling algorithms (e.g., EDF and RMS [20]) can be employed on each processor, finding a feasible partition of tasks to processors has been shown to be NP-hard [12, 13]. In global scheduling, on the other hand, all ready tasks are put into a shared queue and each idle processor fetches the next highest-priority task for execution from the global queue. Despite its flexibility that allows tasks to migrate and execute on different processors, it has been shown that simple global scheduling policies (e.g., global-EDF and global-RMS) could fail to schedule task sets with extremely low system utilization [13].

To fully utilize the system, several *optimal global scheduling algorithms* (which can schedule *any* periodic task set whose total utilization does not exceed the total computing capacity in a feasible manner) have been proposed, with varying complexity. Assuming a continuous time domain, the *T-L Plane* based algorithms were studied in [10, 14]. More recently, a generalized deadline-partitioned fair (DP-Fair) scheduling model was investigated in [19]. However, those algorithms can incur high scheduling overhead as the time allocation to tasks can be arbitrarily small. As a well-known quantum-based optimal global scheduling algorithm, the *proportional fair (Pfair)* scheduler enforces proportional progress (i.e., *fairness*) for each task at every time unit [6]. Several sophisticated variations of Pfair algorithm have also been studied, such as PD [7] and PD^2 [1]. However, by making scheduling decisions at *every* quantum time unit, these algorithms could also lead to high scheduling overhead. Observing that a periodic real-time task can only miss its deadline at its period boundary, we have developed an optimal *boundary fair (Bfair)* scheduling algorithm. By making scheduling decisions and ensuring fairness for tasks *only* at their period boundaries, Bfair can significantly reduce the scheduling overhead [24, 25],

1533-2306/10 $26.00 © 2010 IEEE
DOI 10.1109/RTCSA.2010.15

while fully utilizing all the processors.

Recently, as a hierarchical approach, *cluster scheduling* has been investigated. In this approach, processors are grouped into clusters and tasks are partitioned among different clusters. For tasks that are allocated to a cluster, different global scheduling policies (e.g., global-EDF) can be adopted [4, 9, 23]. Note that, cluster scheduling is a *general* approach, which will reduce to partitioned scheduling when there is only one processor in each cluster. For the case of a single cluster containing all the processors, it will reduce to global scheduling.

As multicore processors will be widely employed in modern real-time systems and processing cores on a chip may be organized into a few groups (i.e., *clusters*), where cores sharing on-chip caches (and even voltage supply through voltage-island techniques) belong to one group [17, 22], additional analysis of cluster scheduling is warranted. Moreover, although the schedulability test and system utilization bounds have been studied for partitioned [21] and global [3, 5] scheduling approaches, such results are not readily available for cluster scheduling.

In this paper, adopting optimal global schedulers to schedule tasks within each cluster, we study the worst-case utilization bound for cluster scheduling. Specifically, for a system with m homogeneous clusters where each cluster has k processors, by extending the worst-case utilization bound for the partitioned-EDF [21], we show that the worst-case achievable system utilization for cluster scheduling is $\frac{\lfloor k/\alpha \rfloor \cdot m+1}{\lfloor k/\alpha \rfloor +1} \cdot k$, where α is the maximum utilization for the periodic tasks under consideration.

Moreover, by using the Bfair algorithm which was shown to be an efficient optimal global scheduler [24, 25], we propose a *period-aware* partitioning heuristic that exploits the harmonicity of tasks' periods to further reduce scheduling overhead. The effects of *cluster size* (defined as the number of processors in a cluster) and the proposed period-aware partitioning heuristic on the performance (i.e., success ratio of schedulable task sets and scheduling overhead) of cluster scheduling with Bfair are evaluated through extensive simulations.

To the best of our knowledge, this is the first work that adopts optimal global schedulers in cluster scheduling and analyzes its corresponding worst-case utilization bound. The main contributions of this work are summarized as follows:

- First, for cluster scheduling where optimal global schedulers are adopted within each cluster, a worst-case utilization bound is derived;
- Second, for cluster scheduling with Bfair, an efficient period-aware partitioning heuristic is proposed;
- Third, the effects of cluster size and the proposed period-aware partitioning heuristic on the performance of cluster scheduling are thoroughly evaluated.

The remainder of this paper is organized as follows. Section 2 presents task and system models and states the problem to be addressed. The optimal global scheduler Bfair is reviewed and the period-aware partitioning heuristic is presented in Section 3. The worst-case utilization bound for cluster scheduling is analyzed in Section 4. Simulation results are presented and discussed in Section 5 and Section 6 concludes the paper.

2 System Models

2.1 Task Model

In this work, we consider a set Γ of n periodic real-time tasks: $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$. Preemption is allowed and there is no dependency between tasks. Each task τ_i is defined by a tuple (c_i, p_i), where c_i is the task's worst-case execution time (WCET) and p_i is its period (which is also the relative deadline of the task). We assume that both c_i and p_i are integers, denoting the number of time quanta. Moreover, it is assumed that tasks are synchronous and the first task instance of each task arrives at time 0. The j's task instance of task τ_i arrives at time $(j-1) \cdot p_i$ and needs to finish its execution by its deadline at time $j \cdot p_i$ ($j \geq 1$).

The *utilization* (or *weight*) of a task τ_i is defined as $u_i = \frac{c_i}{p_i}$ and the *system utilization* is defined as the summation of all tasks' utilization $U = \sum_{i=1}^{n} u_i$. With the assumption that a task cannot execute in parallel on more than one processors, we have $u_i \leq 1$.

2.2 System Model

We consider a shared memory multiprocessor (or multicore) system with m identical processors. For scheduling and other considerations (such as power management in multicore processors [17, 22]), the processors are grouped into m_k homogeneous clusters with each cluster having k processors. That is, we assume that $m = m_k \cdot k$. The system utilization is assumed to be $U \leq m$ (otherwise, the task set will not be schedulable).

Note that it is possible for the processors to be grouped into heterogeneous clusters where each cluster has a different number of processors [22]. However, exploring the full implications of heterogeneous clusters on the worst-case achievable utilization is beyond the scope of this paper and is left for our future work.

2.3 Problem Description

The schedulability analysis for a virtual cluster scheduling mechanism has been addressed in [23], by adopting global-EDF within each cluster. However, it is well-known that global-EDF can lead to very low system utilization [13]. In this paper, by focusing on optimal global scheduling algorithms (such as Bfair [24]) that can fully utilize all processors within each cluster, we investigate **the maximum achievable system utilization under cluster scheduling**. More specifically, we will derive a utilization bound U^{bound} that can be used as a schedulability test in cluster scheduling. That is, any task set with system utilization $U \leq U^{bound}$ should be guaranteed to be schedulable under cluster scheduling.

3 Cluster Scheduling: Bfair and PA-FF

There are two main steps involved in cluster scheduling, which adopts a hierarchical approach. First, tasks are partitioned among different clusters (of processors) following a given strategy. Then, the tasks within each cluster are scheduled by a global scheduler [4, 9, 23]. It can be seen that, to ensure the schedulability of the tasks within each cluster, the number and/or utilization of the tasks that can be allocated to one cluster depend not only on the number of processors in the cluster but also on the global scheduling algorithm deployed within each cluster.

Note that, to obtain the maximum system utilization under the worst-case scenario, ideally optimal global schedulers that can fully utilize all processors within each cluster should be deployed in the second step of cluster scheduling. Therefore, before discussing the partitioning strategies, we first review an efficient optimal global scheduler: the *boundary-fair (Bfair)* scheduling algorithm [24], which will be adopted in this work.

3.1 An Optimal Global Scheduler: Bfair

Several optimal global scheduling algorithms for multiprocessor real-time systems have been studied, including the T-L plane based algorithms [10, 14], the well-known proportional-fair (Pfair) algorithm [6] and its variants [1, 7]. Although all these global schedulers can achieve full system utilization, the T-L plane-based algorithms may require scheduling decisions at *any* time instant [10, 14]. Moreover, by making scheduling decision at every time unit, Pfair algorithms [1, 7, 6] could incur quite high scheduling overhead, especially for systems with small time quantum. Therefore, in this work, we focus on the boundary-fair (Bfair) scheduling algorithm, which makes scheduling decisions and ensures fairness to tasks only at tasks' period boundaries [24].

More specifically, for a subset Γ_s of periodic real-time tasks to be executed on a cluster of k processors, we can define the period boundary time points as $B = \{b_0, \ldots, b_f\}$, where $b_0 = 0$ and $b_j < b_{j+1}$ $(j = 0, \ldots, f-1)$. For every b_j, there exist $\tau_i \in \Gamma_s$ and an integer a, such that $b_j = a \cdot p_i$. Moreover, due to the periodicity of the settings, we consider only the schedule up to the time point of LCM (least common multiple) of all tasks' periods. That is, $b_f = LCM\{p_i | \tau_i \in \Gamma_s\}$. At a boundary time point $b_j (\in B)$, Bfair allocates processors to tasks for the time units between b_j and b_{j+1}.

Define the *allocation error* for task $\tau_i (\in \Gamma_s)$ at a boundary time $b_j (\in B)$ as $\delta(i,j) = X_i(b_j) - b_j \cdot u_i$, where $X_i(b_j)$ represents the total number of time units that are allocated to task τ_i from time 0 to time b_j under Bfair scheduling. Bfair ensures that, for any task $\tau_i (\in \Gamma_s)$ at any boundary time $b_j (\in B)$, $|\delta(i,j)| < 1$ (i.e., the allocation error is within one time unit). By ensuring this property, we have proved that all tasks can meet their deadlines under Bfair if $U_s = \sum_{\tau_i \in \Gamma_s} u_i \leq k$ (i.e., the system utilization is no greater than the number of available processors) [24].

As an example, consider a task set with 6 periodic tasks to be executed on a cluster of two processors: $\Gamma = \{\tau_1(2,5), \tau_2(3,15), \tau_3(3,15), \tau_4(2,6), \tau_5(20,30), \tau_6(6,30)\}$. Here, we have $U = \sum_{i=1}^{6} u_i = 2$ and $LCM = 30$. Figure 1a shows the schedule generated by the Bfair algorithm [24], where the dotted lines in the figure are the period boundaries of the tasks. The numbers in the rectangles denote that the corresponding tasks will be executed on the specific processor during those time units. For comparison, the schedule obtained from the Pfair algorithm [6] is shown in Figure 1b.

From these schedules, we can see that there are only 10 scheduling points for Bfair, while the number of scheduling points for Pfair is 30 (the number of quanta within LCM). Our recent results showed that, compared to Pfair, Bfair can reduce the number of scheduling points by up to 94% [25]. Moreover, by making CPU time allocation in groups, the execution of tasks can be aggregated under Bfair scheduling and the number of context switches can be significantly reduced as well. Note that, between two consecutive scheduling points of Bfair schedule (e.g., time 0 and time 5), processors can execute different tasks which results in additional context switch points (e.g., time 2, 3 and 4 for the processor at the top and time 3 for the processor at the bottom). In this example, there are 45 context switches in the Bfair schedule and the number is 52 for the Pfair schedule within one LCM. Our recent study showed that, compared to Pfair, Bfair can reduce the number of context switches by up to 82% [25]. For the same reasoning, the number of task migrations under Bfair can also be reduced by up to 85% [25]. Such reduction in context switches and task migra-

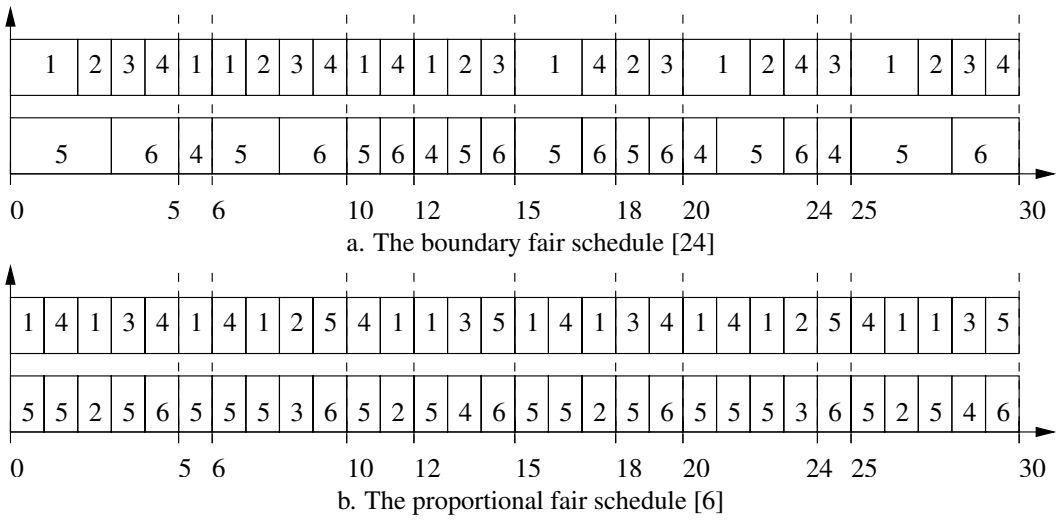


Figure 1. The schedules for the example task set $\Gamma = \{ \tau_1(2,5)$**,** $\tau_2(3,15)$**,** $\tau_3(3,15)$**,** $\tau_4(2,6)$**,** $\tau_5(20,30)$**,** $\tau_6(6,30)\}$ **(the dotted lines are period boundaries).**

tions is critical to reduce the run-time overhead in real-time systems.

Considering its optimality (in terms of achieving full system utilization) and low overhead (in terms of scheduler overhead, the number of context switches and task migrations at run-time), in this paper, we will adopt Bfair as the global scheduler within each cluster in cluster scheduling. The detailed steps of the Bfair algorithm is not shown due to space limitations and can be found in [24]. However, we would like to point out that, the worst-case utilization bound developed for cluster scheduling in this work is not specific to Bfair. *Any optimal global scheduler (e.g. Pfair variants [1, 7, 6] or T-L plane algorithms [10, 14]) that can fully utilize all processors within each cluster will lead to the same utilization bound.*

3.2 Period-Aware First-Fit (PA-FF)

When optimal global schedulers are adopted within each cluster, we can allocate as many tasks as possible to a cluster, provided that the summation of these tasks' utilization is no more than k, the number of processors within the cluster. That is, the partitioning phase of cluster scheduling is essentially a problem of efficiently packing n items into m_k bins, where the size of each item is no more than 1 and each bin has the size of k ($m = k \cdot m_k$). However, unlike the *bin-packing* problem where the goal is to minimize the number of bins to be used or to maximize the number of items to be packed, we are interested in developing efficient partitioning heuristics to pack *all* items in the bins and deriving the corresponding worst-case utilization bound.

A number of heuristics have been studied for the partitioned scheduling problem with different objectives. In this work, with the objective of maximizing the worst-case utilization bound and minimizing the scheduling overhead, we propose a variant of the *first-fit (FF)* heuristic, namely the *period-aware first-fit (PA-FF)* partitioning scheme.

Note that the scheduling points of Bfair within each cluster are all the period boundaries of the tasks allocated to the cluster. If we can allocate tasks with *harmonic* periods to a cluster, all the period boundaries will be determined by the tasks with the smallest period and tasks with larger periods will not increase the number of scheduling points for Bfair within that cluster. That is, the *harmonicity* of tasks' periods can be exploited to reduce the scheduling overhead in cluster scheduling with Bfair, which is different from earlier work on exploiting harmonic periods of tasks to improve the schedulability under rate-monotonic scheduling [8]. Moreover, for tasks with non-harmonic periods, by separating tasks with small periods from the ones with large periods and allocating them to different clusters, the effects of small period tasks on the number of scheduling points will be constrained within their clusters and the number of scheduling points for clusters with large period tasks can be significantly reduced.

Following these guidelines, the detailed steps of the *PA-FF* partitioning scheme are shown in Algorithm 1. First, the task with the smallest period are put in a harmonic task set ($HarmonicSet$), which are followed by tasks with harmonic periods in the increasing order of tasks' periods (lines 4 to 17). Then, all tasks in the har-

Algorithm 1 Period-Aware First-Fit (PA-FF) Heuristic

1: Find $p_{max} = \max_i^n p_i$;
2: Initialize an ordered task list: $LIST = \emptyset$;
3: **while** (Task set $\Gamma \neq \emptyset$) **do**
4: Find a task τ_x with $p_x = \min\{p_i | \tau_i \in \Gamma\}$;
5: $HarmonicSet = \{\tau_x\}$; $\Gamma - = \{\tau_x\}$;
6: $LCM = p_x$; $j = 1$;
7: **while** ($LCM \cdot j \leq p_{max}$) **do**
8: **if** ($\exists \tau_i \in \Gamma$ with $p_i = LCM \cdot j$) **then**
9: **for all** (τ_i with $p_i = LCM \cdot j$) **do**
10: $HarmonicSet + = \{\tau_i\}$; $\Gamma - = \{\tau_i\}$;
11: **end for**
12: $LCM = LCM \cdot j$;
13: $j = 1$;
14: **else**
15: $j = j + 1$;
16: **end if**
17: **end while**
18: Move all tasks in $HarmonicSet$ in increasing period order to $LIST$;
19: **end while**
20: Allocate tasks in the order as in $LIST$ to clusters with First-Fit heuristic;

monic tast set will be added to an ordered task list in the order of increasing tasks' periods (line 18). The above process is repeated until all tasks are put in the ordered task list (lines 3 to 19). Then, in the same order as in the ordered task list, the tasks are allocated to clusters with the first-fit heuristic (line 20). The superiority of the new PA-FF scheme over the simple first-fit heuristic on reducing the scheduling overhead (e.g., the number of context switches and task migrations) of cluster scheduling with Bfair is evaluated and illustrated in Section 5.

Note that, the periods of tasks are not determining factors for system utilization. In what follows, we can see that the worst-case utilization bound for cluster scheduling does not depend on the particular PA-FF heuristic. More specifically, we will show that, for any *reasonable* partitioning heuristic [21] that has the objective of maximizing system utilization (such as *first-fit (FF)*, *best-fit(BF)*, *first-fit decreasing (FFD)* and *best fit decreasing (BFD)*), the same worst-case utilization bound for cluster scheduling holds. Incidentally, the same bound holds also for the PA-FF scheme, which is a variant of *FF*.

4 Utilization Bound for Cluster Scheduling

When optimal global schedulers are deployed within each cluster, any subset of tasks allocated to a cluster can be scheduled provided that the summation of these tasks' utilization is no more than the number of processors in that cluster. That is, the worst-case utilization bound for cluster scheduling with an optimal global scheduler is essentially determined by its partitioning scheme. In this work, by focusing on *reasonable* partitioning heuristics (e.g., FF, BF, FFD and BFD [21]), we will develop the corresponding worst-case utilization bound, which is formally defined as follows.

Definition 1 *For a system with m_k clusters where each cluster has k processors, the* worst-case achievable system utilization bound *for cluster scheduling with any reasonable (RA) partitioning scheme (e.g., FF, BF, FFD or BFD) and an optimal (OPT) global scheduler is defined as a real number U_{RA-OPT}^{bound}, such that:*

- *Any periodic task set with system utilization $U \leq U_{RA-OPT}^{bound}$ can be scheduled by cluster scheduling with a reasonable partitioning scheme on m_k clusters, where each cluster has k processors and adopts an optimal global scheduler;*
- *For any system utilization $U' > U_{RA-OPT}^{bound}$, it is always possible to find a task set with system utilization as U', and the task set cannot be scheduled by cluster scheduling on m_k clusters each with k processors.*

Note that, when each of the clusters has only one processor (i.e., $k = 1$), the problem will reduce to finding the utilization bound for the partitioned scheduling, which has been derived by Lopez *et. al* [21]. Specifically, for any reasonable partitioning heuristic, the utilization bound for the partitioned-EDF is given as [21]:

$$U_{partition-EDF}(m, \beta) = \frac{\beta \cdot m + 1}{\beta + 1} \tag{1}$$

where m is the number of processors in the system; and $\beta = \lfloor \frac{1}{\alpha} \rfloor$ denotes the maximum number of tasks that can fit in one processor if all tasks have the maximum task utilization $\alpha (\leq 1)$. When $\alpha = 1$, the bound reduces to $(m + 1)/2$. That is, for systems with large number of processors, partitioned-EDF can only achieve around 50% system utilization in the worst-case scenario.

For the cases where each cluster has more processors, by extending the result in Equation (1), we can have the following theorem regarding the worst-case utilization bound for cluster scheduling that adopts reasonable partitioning heuristics and optimal global schedulers.

Theorem 1 *For a real-time system with m processors that are grouped into m_k clusters with each cluster having k processors ($m = k \cdot m_k$), if cluster scheduling adopts a reasonable partitioning heuristic (such as FF, BF, FFD or BFD) and an optimal global scheduler within each cluster, the worst-case utilization bound for cluster scheduling is:*

$$U_{RA-OPT}^{bound}(m_k, k) = \frac{k \cdot m_k + 1}{k + 1} \cdot k \tag{2}$$

Proof We will prove the theorem by transforming the problem of cluster scheduling to that of partitioned scheduling with EDF.

Note that, for any problem of scheduling a set of tasks $\Gamma = \{\tau_i | i = 1, \ldots, n\}$ on m_k clusters each with k processors under cluster scheduling, we can construct a corresponding partitioned scheduling problem with a set of tasks $\Gamma' = \{\tau'_i | i = 1, \ldots, n\}$ and m_k processors each of unit capacity, where the utilization of task τ'_i is $\frac{1}{k}$ of that of the task τ_i. That is, $u'_i = \frac{u_i}{k}$ for $i = 1, \ldots, n$.

We can see that, for *any* partitioning heuristic, if it can successfully partition the tasks in task set Γ' on m_k processors without exceeding each processor's capacity, the same heuristic will be able to allocate the tasks in task set Γ on m_k clusters without exceeding each cluster's capacity, which is k.

Note that, for every task $\tau_i \in \Gamma$, we have $u_i \leq 1 (i = 1, \ldots, n)$. Therefore, the maximum task utilization for tasks in Γ' will be $\alpha' = \max\{u'_i | i = 1, \ldots, n\} = \max\{\frac{u_i}{k} | i = 1, \ldots, n\} \leq \frac{1}{k}$. From Equation (1), we know that, for any of the reasonable partitioning heuristics (e.g., FF, BF, FFD and BFD), the task set Γ' can be successfully allocated on m_k processors if the system utilization of Γ' is $U' = \sum_i^n u'_i \leq \frac{\lfloor (1/\alpha') \rfloor \cdot m_k + 1}{\lfloor (1/\alpha') \rfloor + 1} = \frac{k \cdot m_k + 1}{k+1}$.

Hence, for the original task set Γ, any reasonable partitioning heuristic can allocate all tasks on m_k clusters each with k processors if the system utilization $U = \sum_i^n u_i = k \sum_i^n u'_i = k \cdot U' \leq \frac{k \cdot m_k + 1}{k+1} k$, which concludes the proof. ■

Following the same reasoning, if the maximum task utilization of the tasks in task set Γ is $\alpha (\leq 1)$, the utilization bound can be generalized as:

$$U_{RA-OPT}^{bound}(m_k, k, \beta) = \frac{\beta \cdot m_k + 1}{\beta + 1} \cdot k \qquad (3)$$

where $\beta = \lfloor k/\alpha \rfloor$ denotes the maximum number of tasks that can fit in one cluster of k processors if all tasks have utilization as α.

Note that, if the cluster size is one (i.e., $k = 1$ and $m_k = m$), the bounds given in the above two equations are reduced to the worst-case utilization bounds for partitioned-EDF studied by Lopez *et al.* in [21]. Moreover, if all processors belong to one cluster (i.e., $m_k = 1$ and $k = m$), the utilization bound will be m, the number of processors in the system, which coincides with the assumption that the optimal global scheduler can schedule any task set with system utilization not exceeding the number of processors.

From the above equations, we can also see that, for a system with a given number of processors, the organization of the clusters (i.e., size and number of clusters) has a direct effect on the worst-case utilization bound. As a concrete example, Figure 2 shows the bound for a 64-processor system for different organizations of the clusters. The Y-axis depicts the normalized utilization bound (that is, $U_{RA-OPT}^{bound}/64$).

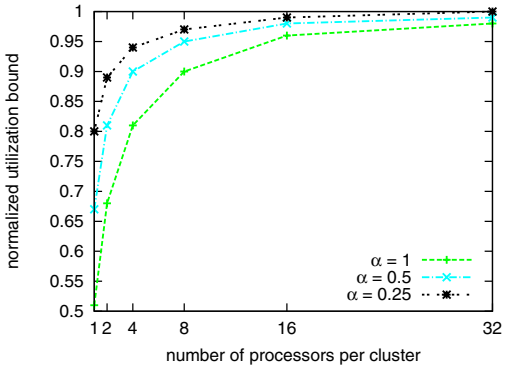


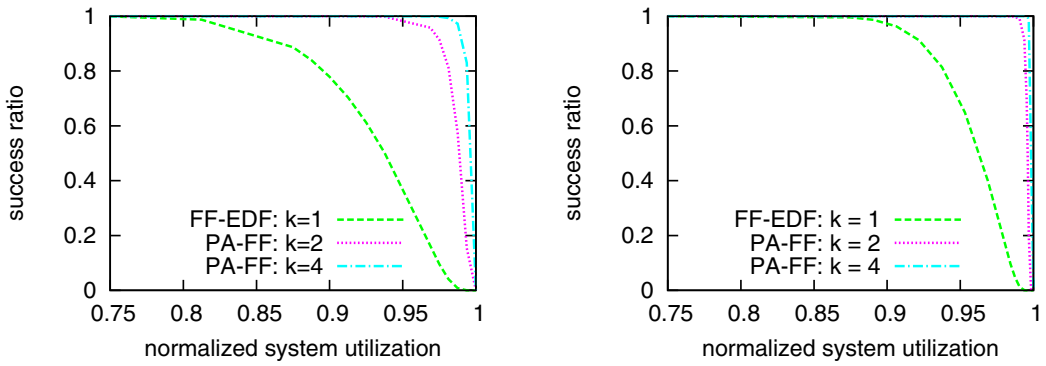
Figure 2. Utilization bounds for a 64-processor system.

As expected, for a given value of the maximum task utilization α, the utilization bound increases when the number of processors in each cluster increases and fewer clusters are organized. For instance, if the processors are organized as 4 clusters each having 16 processors, the normalized utilization bound is 0.95, while the bound is only 0.8 for the organization of 16 clusters each with 4 processors. The reason comes from the fact that, with larger size clusters, the utilization waste due to fragmentation of partitioning becomes less significant. Moreover, the effect of α on the utilization bound is more prominent for smaller clusters.

However, as shown in Section 5, it could be beneficial to organize the processors as smaller clusters in terms of reducing scheduling overhead. That is, for smaller clusters, there will be relatively fewer number of tasks and the number of scheduling points (i.e., period boundaries) can be significantly reduced within each cluster, leading to less scheduling overhead.

5 Evaluations and Discussions

Note that the utilization bounds given in Equations (2) and (3) correspond to the worst-case scenario. Thus, task sets with system utilization larger than the derived bounds could still be schedulable under cluster scheduling. In this section, focusing on the *success ratio* of schedulable task sets and *scheduling overhead*, we will empirically evaluate the effects of cluster size and the proposed period-aware first-fit (PA-FF) partitioning heuristic on the performance of cluster scheduling with Bfair through extensive simulations. For comparison, the



(a) 16-processor systems, $m = 16$

(b) 64-processor systems, $m = 64$

Figure 3. Success ratios for cluster scheduling and partitioned FF-EDF.

simple first-fit (FF) heuristic is also considered.

In the simulations, synthetic task sets are generated from a set of parameters: task set system utilization U_{tot}, the maximum task utilization α, the minimum task period P_{min} and the maximum task period P_{max}. In this paper, we consider systems with 16 (i.e., $m = 16$) and 64 processors (i.e., $m = 64$), respectively. Moreover, we define the *normalized system utilization* as $\frac{U_{tot}}{m}$, which will vary from 0.75 to 1.0. The maximum task utilization α has the range of $[0.2, 1.0]$. Unless specified otherwise, when generating the periods of tasks, we considered $P_{min} = 10$ and $P_{max} = 100$.

For a given setting, the period and utilization of a task are first generated within the range of $[P_{min}, P_{max}]$ and $(0, \alpha]$, respectively, following the uniform distribution. Then, to ensure the integer property of the task's worst-case execution time (WCET), its utilization is adjusted so as not to exceed α. More tasks are generated repeatedly provided that the summation of their utilization is no greater than U_{tot}, the target system utilization. When the difference between U_{tot} and the summation utilization of generated tasks is less than α, the last task takes its utilization as the difference (i.e., the system utilization of the task set is exactly U_{tot}).

5.1 Success Ratio

For task sets with different system utilization, we first evaluate the number of task sets that can be successfully scheduled under cluster scheduling with PA-FF and Bfair. For this purpose, we define the *success ratio* as the number of task sets that are schedulable under a given scheduling algorithm over the total number of the generated task sets. With the maximum task utilization being fixed as $\alpha = 1.0$, $1,000,000$ task sets are generated for each setting following the above steps.

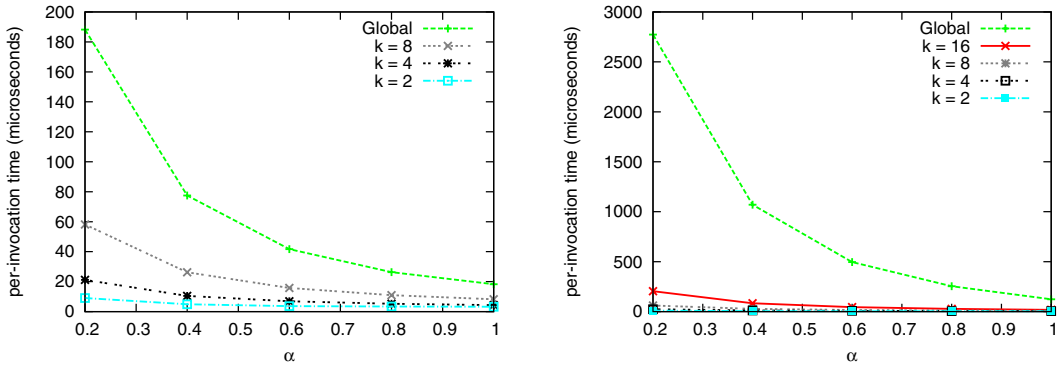
Figures 3(a) and 3(b) show the success ratio of the generated task sets for systems with 16 and 64 processors, respectively. Here, the X-axis shows the normalized system utilization (i.e., $\frac{U_{tot}}{m}$) of the task sets generated. For cluster scheduling, only the results for cluster size of $k = 2$ and $k = 4$ are presented. For larger clusters with more processors, the success ratio under cluster scheduling is almost 1 even for very high system utilization. For comparison, the success ratio of schedulable task sets under partitioned scheduling with first-fit heuristic and EDF (FF-EDF) [21] is also shown.

From the results, we can see that, for the system with 16 processors ($m = 16$), even with clusters of the smallest size (i.e., $k = 2$), the cluster scheduling can successfully schedule almost all task sets when the normalized system utilization does not exceed 0.94. Not surprisingly, the cluster scheduling performs better with larger clusters as the wasted utilization fragmentation becomes relatively less. For instance, for the case where each cluster has four processors (i.e., $k = 4$), the normalized system utilization limit can reach 0.98 while almost all task sets are still schedulable. In comparison, the partitioned FF-EDF can only schedule almost all task sets when the normalized system utilization does not exceed 0.81.

When the maximum task utilization becomes smaller (e.g., $\alpha = 0.5$), better success ratios are achieved by both schemes. In that case, cluster scheduling still outperforms partitioned FF-EDF (the details are omitted due to space limitations). Moreover, for systems with more processors, more clusters can be formed and tasks have more chance to fit into one of them. As shown in Figure 3(b) for systems with 64 processors, both cluster scheduling and partitioned FF-EDF perform better and can schedule more task sets with higher normalized system utilization.

5.2 Scheduling Overhead

Although cluster scheduling with larger clusters can successfully schedule more task sets with higher system utilization, larger clusters could lead to higher scheduling



(a) 16-processor systems (b) 64-processor systems

Figure 4. Time overhead of cluster-Bfair at each scheduling point within a cluster.

overhead. With more tasks being allocated to each cluster, there will be more scheduling points for the global scheduler Bfair and the interval between adjacent period boundaries can become smaller. Therefore, in addition to the time overhead for invoking Bfair at each scheduling point (which largely depends on the number of tasks in a cluster), the generated schedule can require more context switches and task migrations due to shorter execution of tasks within each interval between period boundaries [24, 25].

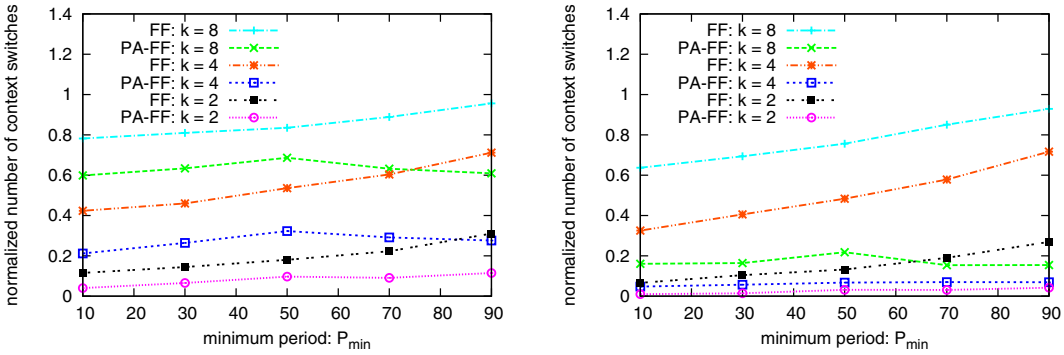
For systems where the processors are organized as different size of clusters, the scheduling overhead of cluster scheduling with Bfair is also evaluated through extensive simulations. Here, to ensure that most of the generated task sets are schedulable, we set the normalized system utilization to $\frac{U_{tot}}{m} = 0.95$. Moreover, we fix $P_{min} = 10$ and $P_{max} = 100$. The value of the maximum task utilization α is varied from 0.2 to 1.0. With fixed system utilization, varying α effectively affects the number of tasks in the task sets under consideration, where smaller values of α mean more tasks for each task set. For each data point in the following figures, 100 schedulable task sets are generated and the average result is reported.

Time Overhead: For cluster scheduling with different cluster sizes, Figures 4(a) and 4(b) first show the invocation time of the Bfair algorithm at each scheduling point for 16-processor and 64-processor systems, respectively. Here, the algorithms are implemented in C and run on a Linux machine with an Intel 2.4GHz processor. Note that, when all processors in a system form a single cluster (i.e., $k = m$), the cluster scheduling essentially becomes to be the global Bfair [24], which is labeled as "Global" and used for comparison.

As shown in [24, 25], the time overhead of Bfair at each scheduling point depends largely on the number of tasks to be scheduled together. For cluster scheduling, after partitioning tasks to clusters, the scheduling decisions of Bfair for each cluster can be made independently. Therefore, as we can see from Figure 4(a), when cluster size becomes smaller, the time overhead at a scheduling point for a cluster can be significantly reduced. The reason comes from the fact that fewer tasks are allocated to a smaller cluster. Moreover, as α becomes smaller, more tasks will be contained in each task set. That is, more tasks will be allocated to each cluster, and the time overhead of Bfair at each scheduling point generally increases. However, the effect of α is more prominent for larger clusters, especially for the global Bfair where there is only one cluster and all tasks are handled together. For systems with more processors (Figure 4(b)), there are more tasks can be scheduled and the time overhead at each scheduling point becomes larger, which is consistent with our previous results reported in [24, 25].

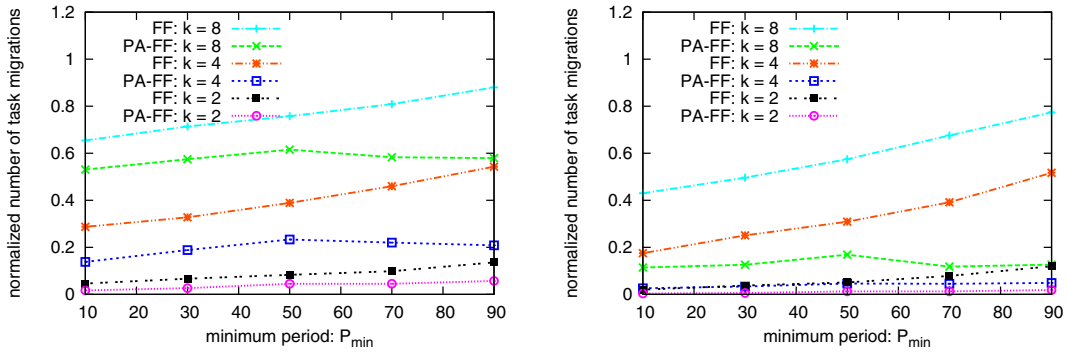
Context Switches and Task Migrations: Next, we evaluate the schedules generated by cluster scheduling with different sizes of clusters in terms of the required number of context switches and task migrations. Here, the maximum task utilization is fixed as $\alpha = 1.0$ and the normalized system utilization is set as $\frac{U_{tot}}{m} = 0.95$. With fixed $P_{max} = 100$, we vary P_{min} from 10 to 90 and evaluate the effects of tasks' periods on cluster scheduling with Bfair. The proposed period-aware first-fit (PA-FF) partitioning heuristic is evaluated against the simple first-fit (FF) partitioning heuristic where tasks are randomly ordered. For easy comparison, those of the schedule generated by the global Bfair (i.e., with cluster size $k = m$) are used as a baseline and normalized results are reported.

Figures 5(a) and 5(b) show the normalized number of context switches for the schedules generated by cluster scheduling for systems with 16 and 64 processors, respectively. From the results, we can see that, compared to that of the schedule generated by the global Bfair where



(a) 16-processor systems (b) 64-processor systems

Figure 5. Normalized number of context switches



(a) 16-processor systems (b) 64-processor systems

Figure 6. Normalized number of task migrations

all tasks are scheduled together, the number of context switches in the schedule generated by cluster scheduling decreases drastically as cluster size becomes smaller. For instance, when the cluster size is $k = 2$, the number of context switches can be reduced by more than 90%. The reason is that, with fewer tasks in a smaller cluster, the scheduling points that are the tasks' period boundaries become fewer, which in turn provides more opportunities for tasks to run for more consecutive time units and thus to reduce the number of context switches. Moreover, by allocating harmonic tasks together, the proposed PA-FF partitioning scheme can further reduce the number of context switches significantly when compared to that of the simple FF heuristic, especially for the ones with large size clusters and systems with more processors.

Furthermore, as P_{min} increases while P_{max} is fixed as 100, more tasks are likely to have the same period. That is, the number of scheduling points becomes less for both global Bfair and cluster scheduling for each cluster. However, the reduction in scheduling points is more significant for global Bfair algorithm, which leads to a much reduced number of context switches [24, 25]. Therefore, for the simple FF heuristic, the normalized number of context switches for the schedules generated by cluster scheduling increases slightly as P_{min} increases. However, for the proposed PA-FF partitioning scheme that allocates tasks with harmonic periods together, the normalized number of context switches stays roughly the same.

Figures 6(a) and 6(b) further show the normalized number of task migrations for the schedules generated by cluster scheduling with Bfair. Due to the similar reasons, the number of task migrations is also reduced under cluster scheduling, especially with smaller clusters and the PA-FF partitioning scheme.

6 Conclusions

In this paper, with an optimal global scheduler being adopted within each cluster, we studied the worst-case utilization bound for cluster scheduling (where processors are grouped into clusters and tasks allocated to

one cluster are scheduled by a global scheduler). For a system where processors are grouped into m homogeneous clusters with each cluster having k processors, we proved that the worst-case achievable system utilization is $\frac{\lfloor k/\alpha \rfloor \cdot m+1}{\lfloor k/\alpha \rfloor+1} \cdot k$, where α is the maximum utilization of the periodic tasks considered. Focusing on an efficient optimal global scheduler, the *boundary-fair (Bfair)* scheduling, we also proposed a *period-aware* partitioning heuristic that attempts to allocate tasks with harmonic periods together in order to to reduce scheduling overhead.

The simulation results showed that, compared to that of the partitioned scheduling, the success ratio of schedulable task sets can be significantly improved under cluster scheduling even with small size clusters (e.g., $k = 2$). Moreover, when compared to global Bfair scheduling, cluster scheduling can drastically reduce the scheduling overhead (such as execution time of the scheduler, and the number of context switches and task migrations for the generated schedules), especially for the cases with the proposed period-aware partitioning heuristic and small size clusters (i.e., $k = 2$ or $k = 4$).

References

[1] J.H. Anderson and A. Srinivasan. Early-release fair scheduling. In *Proc. of the* 12^{th} *Euromicro Conference on Real-Time Systems*, Jun. 2000.

[2] B. Andersson, K. Bletsas, and S. Baruah. Scheduling arbitrary-deadline sporadic task systems on on multiprocessors. In *Proc. of the 2008 IEEE Real-Time Systems Symposium*, pages 385–394, 2008.

[3] T.P. Baker. Multiprocessor edf and deadline monotonic schedulability analysis. In *Proc. of The* 24^{th} *IEEE Real-Time Systems Symposium*, pages 120–129, Dec. 2003.

[4] S. Baruah. Techniques for multiprocessor global schedulability analysis. In *Proc. of the IEEE Real-Time Systems Symposium*, pages 119–128, 2007.

[5] S. Baruah and T. Baker. Schedulability analysis of global edf. *Real-Time Syst.*, 38(3):223–235, 2008.

[6] S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varel. Proportionate progress: A notion of fairness in resource allocation. *Algorithmica*, 15(6):600–625, 1996.

[7] S. K. Baruah, J. Gehrke, and C. G. Plaxton. Fast scheduling of periodic tasks on multiple resources. In *Proc. of The International Parallel Processing Symposium*, Apr. 1995.

[8] A. Burchard, J. Liebeherr, Y. Oh, and S. H. Son. New strategies for assigning real-time tasks to multiprocessor systems. *IEEE Trans. on Computers*, 44(12):1429–1442, 1996.

[9] J. M. Calandrino, J. H. Anderson, and D. P. Baumberger. A hybrid real-time scheduling approach for large-scale multicore platforms. In *Proc. of the Euromicro Conference on Real-Time Systems*, pages 247–258, 2007.

[10] H. Cho, B. Ravindran, and E.G. Jensen. An optimal real-time scheduling algorithm for multiprocessors. In *Proc. of the IEEE Real-Time Systems Symposium*, 2006.

[11] R. Davis and A. Burns. Priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 398–409, 2009.

[12] M. L. Dertouzos and A. K. Mok. Multiprocessor on-line scheduling of hard-real-time tasks. *IEEE Trans. On Software Engineering*, 15(12):1497–1505, 1989.

[13] S. K. Dhall and C. L. Liu. On a real-time scheduling problem. *Operation Research*, 26(1):127–140, 1978.

[14] K. Funaoka, S. Kato, and N. Yamasaki. Work-conserving optimal real-time scheduling on multiprocessors. In *Proc. of the Euromicro Conference on Real-Time Systems*, 2008.

[15] N. Guan, M. Stigge, W. Yi, and G. Yu. New response time bounds for fixed priority multiprocessor scheduling. In *IEEE Real-Time Systems Symposium*, 2009.

[16] N. Guan, M. Stigge, W. Yi, and G. Yu. Fixed-priority multiprocessor scheduling with liu & laylands utilization bound. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2010.

[17] S. Herbert and D. Marculescu. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *Proc. of ISLPED*, pages 38–43, 2007.

[18] S. Kato and N. Yamasaki. Real-time scheduling with task splitting on multiprocessors. In *Proc. of the IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2007.

[19] G. Levin, S. Funk, C. Sadowski, I. Pye, and S. Brandt. Dp-fair: A simple model for understanding optimal multiprocessor scheduling. In *Proc. of the Euromicro Conference on Real-Time Systems*, 2010.

[20] C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *J. ACM*, 20(1):46–61, 1973.

[21] J.M. Lopez, M. Garcia, J.L. Diaz, and D.F. Garcia. Worst-case utilization bound for edf scheduling on real-time multiprocessor systems. In *Prof. of the 12th Euromicro Conference on Real-Time Systems*, pages 25–33, 2000.

[22] X. Qi and D. Zhu. Power management for real-time embedded systems on block-partitioned multicore platforms. In *Proc. of the Int'l Conference on Embedded Software and Systems (ICESS)*, 2008.

[23] I. Shin, A. Easwaran, and I. Lee. Hierarchical scheduling framework for virtual clustering of multiprocessors. In *Proceedings of the 20th Euromicro Conference on Real-Time Systems*, Jul. 2008.

[24] D. Zhu, D. Mossé, and R. Melhem. Periodic multiple resource scheduling problem: how much fairness is necessary. In *Proc. of The* 24^{th} *IEEE Real-Time Systems Symposium (RTSS)*, pages 142–151, Dec. 2003.

[25] D. Zhu, X. Qi, D. Mossé, and R. Melhem. An optimal boundary-fair scheduling algorithm for multiprocessor real-time systems. Technical Report CS-TR-2009-005, Dept. of CS, Univ. of Texas at San Antonio, 2009.