

Cluster scheduling for real-time systems: utilization bounds and run-time overhead

Xuan Qi · Dakai Zhu · Hakan Aydin

Published online: 24 March 2011
© Springer Science+Business Media, LLC 2011

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, assuming that an optimal global scheduler is adopted within each cluster, we investigate the *worst-case utilization bounds* for cluster scheduling with different task allocation/partitioning heuristics. First, we develop a lower limit on the utilization bounds for cluster scheduling with any *reasonable task allocation scheme*. Then, the lower limit is shown to be the exact utilization bound for cluster scheduling with the *worst-fit* task allocation scheme. For other task allocation heuristics (such as *first-fit*, *best-fit*, *first-fit decreasing*, *best-fit decreasing* and *worst-fit decreasing*), higher utilization bounds are derived for systems with both *homogeneous* clusters (where each cluster has the same number of processors) and *heterogeneous* clusters (where clusters have different number of processors). In addition, focusing on an efficient optimal global scheduler, namely the *boundary-fair (Bfair)* algorithm, we propose a *period-aware* task allocation heuristic with the goal of reducing the scheduling overhead (e.g., the number of scheduling points, context switches and task migrations). Simulation results indicate that the percentage of task sets that can be scheduled is significantly improved under cluster scheduling even for small-size clusters, compared to that of the partitioned scheduling. Moreover, when comparing to the simple generic task

X. Qi · D. Zhu (✉)
Department of Computer Science, University of Texas at San Antonio, One UTSA Circle, San Antonio, TX 78249, USA
e-mail: dzhu@cs.utsa.edu

X. Qi
e-mail: xqi@cs.utsa.edu

H. Aydin
Department of Computer Science, George Mason University, 4400 University Drive, Fairfax, VA 22030, USA
e-mail: aydin@cs.gmu.edu

allocation scheme (e.g., first-fit), the proposed period-aware task allocation heuristic markedly reduces the scheduling overhead of cluster scheduling with the Bfair scheduler.

Keywords Real-time systems · Multiprocessors · Cluster scheduling · Utilization bound · Bfair algorithm · Scheduling overhead

1 Introduction

Real-time scheduling theory has been studied for decades and many well-known scheduling algorithms have been developed for various real-time tasks for both single-processor and multiprocessor systems (Baruah et al. 1996; Dertouzos and Mok 1989; Dhall and Liu 1978; Liu and Layland 1973). With the emergence of multi-core processors, there is a reviving interest in the multiprocessor real-time scheduling problem and many results have been reported recently (Andersson et al. 2008; Baruah and Baker 2008; Davis and Burns 2009a; Devi and Anderson 2005; Guan et al. 2009, 2010; Kato and Yamasaki 2007; Levin et al. 2010).

Traditionally, there have been two major approaches to the scheduling problem in multiprocessor real-time systems: *partitioned* and *global* scheduling (Dertouzos and Mok 1989; Dhall and Liu 1978). In partitioned scheduling, tasks are statically assigned to processors and a task can only run on its designated processor. Although well-established uniprocessor scheduling algorithms (e.g., earliest-deadline-first (EDF) and rate-monotonic scheduling (RMS) (Liu and Layland 1973)) can be employed on each processor, finding a feasible task allocation/partition was shown to be NP-hard (Dertouzos and Mok 1989; Dhall and Liu 1978). Based on the simple task allocation heuristics (such as first-fit, best-fit and worst-fit), the utilization bounds have been developed for partitioned scheduling with both EDF (Darera 2006; Lopez et al. 2000, 2004) and RMS (Darera 2006; Lopez et al. 2001; Oh and Baker 1998), which can be used in efficient schedulability tests. To further improve system utilization, *semi-partitioning* based scheduling has been studied recently where most tasks are statically allocated to processors and only a few tasks may migrate among processors at run time (Andersson and Tovar 2006; Guan et al. 2010; Kato et al. 2009; Kato and Yamasaki 2007, 2008).

In global scheduling, on the other hand, all tasks are put into a shared queue and each idle processor fetches the next highest-priority ready 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 very low system utilization (Dhall and Liu 1978). For task sets where the maximum task utilization is limited, some utilization bounds have also been studied for global-EDF and global-RMS scheduling algorithms (Andersson et al. 2001; Baker 2003; Goossens et al. 2003).

Several *optimal* global scheduling algorithms, which can schedule any periodic task set whose total utilization does not exceed the computing capacity of a multiprocessor system and thus achieve full system utilization, have been studied as well.

For *continuous-time* based systems, the *T-L plane* based scheduling algorithms were studied in Cho et al. (2006), Funaoka et al. (2008). More recently, a generalized deadline-partitioned fair (DP-Fair) scheduling model was investigated in Levin et al. (2010). However, those algorithms may generate arbitrarily small time allocation to tasks due to the continuous-time property and thus incur very high scheduling overhead. As the well-known *quantum-based* optimal scheduler, the *proportional fair* (Pfair) scheduling algorithm enforces proportional progress (i.e., *fairness*) for each task in every time quantum (Baruah et al. 1996). Several sophisticated variations of Pfair algorithm have also been studied, such as *PD* (Baruah et al. 1995) and *PD*² (Anderson and Srinivasan 2000a). However, by making scheduling decisions in every time quantum, 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, another optimal *boundary-fair* (Bfair) scheduling algorithm that can also achieve full system utilization was proposed (Zhu et al. 2003). By making scheduling decisions and ensuring fairness for tasks *only* at the period boundaries, Bfair can significantly reduce the scheduling overhead (Zhu et al. 2003, 2009).

Recently, as a general and 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 (Baruah 2007; Bletsas and Andersson 2009; Calandrino et al. 2007; Shin et al. 2008). 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 (Herbert and Marculescu 2007; Qi and Zhu 2008), additional analysis of cluster scheduling is warranted. Moreover, although the schedulability test and system utilization bounds have been studied for both partitioned and global scheduling approaches, such results are not readily available for cluster scheduling.

In this paper, assuming an optimal global scheduler which can fully utilize the computational capacity within each cluster, we derive the worst-case utilization bounds for cluster scheduling with different task allocation/partitioning schemes. Moreover, focusing on the Bfair algorithm that has been shown to be an efficient optimal global scheduler (Zhu et al. 2003, 2009), we propose a *period-aware* task allocation/partition heuristic that exploits the harmonicity of tasks' periods to further reduce scheduling overhead. 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 bounds. The main contributions of this work can be summarized as follows:

- A lower limit on the utilization bounds for cluster scheduling with any *reasonable task allocation scheme* (Lopez et al. 2004) is derived.
- The exact utilization bound for cluster scheduling with worst-fit task allocation is shown to coincide with this lower limit, regardless of the number of tasks in each cluster.

- For other task allocation heuristics (such as first-fit, best-fit, first-fit decreasing, best-fit decreasing and worst-fit decreasing), the utilization bounds for cluster scheduling are developed for both homogeneous clusters (which have the same number of processors) and heterogeneous clusters (where clusters have different number of processors).
- For cluster scheduling with the Bfair scheduler, an efficient *period-aware* task allocation heuristic is proposed.
- Finally, the effects of cluster size and the proposed period-aware task allocation heuristic on the performance of cluster scheduling are thoroughly evaluated through extensive simulations.

The remainder of this paper is organized as follows. The related work is reviewed in Sect. 2. Section 3 presents the system model and the problem that is addressed in this work. In Sect. 4, the Bfair scheduler is reviewed and an efficient period-aware task allocation heuristic is proposed. The worst-case utilization bounds for cluster scheduling with different task allocation heuristics are derived in Sect. 5. Simulation results are presented and discussed in Sect. 6. Section 7 concludes the paper.

2 Related work

Although rate monotonic (RM) scheduling and earliest deadline first (EDF) policies have been shown to be optimal for static and dynamic priority assignments respectively, for periodic real-time applications running on uni-processor systems (Liu and Layland 1973), neither of them is optimal in multiprocessor settings (Dhall and Liu 1978). Based on partitioned scheduling, Oh and Baker studied the rate-monotonic first-fit (RMFF) heuristic and showed that RMFF can schedule any system of periodic tasks with total utilization bounded by $m(2^{1/2} - 1)$, where m is the number of processors in the system (Oh and Baker 1998). Later, a better bound of $(m + 1)(2^{1/(m+1)} - 1)$ for RMFF was derived in Lopez et al. (2001). In Andersson and Jonsson (2003), Andersson and Jonsson proved that the system utilization bound can reach 50% for a partitioned RM scheduling by exploiting the harmonicity of tasks' periods. For the partitioned scheduling with earliest deadline first (EDF) first-fit heuristic, Lopez et al. showed that any task set can be successfully scheduled if the total utilization is no more than $(\beta \cdot m + 1)/(\beta + 1)$, where $\beta = \lfloor 1/\alpha \rfloor$ and α is the maximum task utilization of the tasks under consideration (Lopez et al. 2004; Lopez et al. 2000). Following similar techniques, the utilization bounds for partitioned-EDF in *uniform* multiprocessor systems (where the processors have the same functionalities but different processing speeds) were developed by Darera (2006).

For global-EDF scheduling, it has been shown that a task set is schedulable on m processors if the total utilization does not exceed $m(1 - \alpha) + \alpha$ (Goossens et al. 2003). Similarly, for global-RMS scheduling, a system utilization of $(m/2)(1 - \alpha) + \alpha$ can be guaranteed (Baker 2003). Andersson et al. also studied the scheduling algorithm RM-US, where tasks with utilization higher than some threshold θ have the highest priority (Andersson et al. 2001). For $\theta = \frac{1}{3}$, Baker showed that RM-US can guarantee a system utilization of $(m + 1)/3$ (Baker 2003).

To further improve the achievable system utilization and reduce the scheduling overhead, Andersson et al. proposed the EKG algorithm based on the concept of *portion tasks* (Andersson and Tovar 2006). In EKG, a separator is defined as $SEP = \frac{k}{k+1}$ for cases where $k < m$ (m is the number of processors) and $SEP = 1$ for the case of $k = m$. The *heavy* tasks with utilization larger than SEP are allocated to their dedicated processors following the conventional partitioned-EDF scheduling. Each *light* task (with utilization no greater than SEP) is split into two portion tasks only if necessary, where the portion tasks are assigned to adjacent processors. The worst-case system utilization bound that can be achieved by EKG is 66% when it is configured with $k = 2$ which results in a small number of preemptions. Full (i.e., 100%) system utilization can be achieved by setting $k = m$, which can result in high scheduling overhead with many preemptions as EKG may allocate arbitrarily small share of time to tasks. In the same line of research, several semi-partitioning based scheduling algorithms have been proposed very recently, with different handling mechanisms for portion tasks and different system utilization bounds (Guan et al. 2010; Kato et al. 2009; Kato and Yamasaki 2007, 2008).

The first global scheduling algorithm based on discrete quantum time model, the *proportional fair* (*Pfair*) algorithm, can achieve full system utilization and hence is optimal (Baruah et al. 1996). The basic idea in Pfair algorithm is to enforce proportional progress (i.e., *fairness*) for all tasks at each time unit, which actually imposes a more strict requirement for the problem. Any Pfair schedule for a set of periodic real-time tasks will ensure that all task instances can complete their executions before their deadlines, as long as the total utilization does not exceed the total number of processors. By separating tasks as *light* (with weights no greater than 50%) and *heavy* (with weights larger than 50%), a more efficient Pfair algorithm, *PD*, is proposed in Baruah et al. (1995). A simplified *PD* algorithm, *PD*², uses two fewer parameters than *PD* to compare the priorities of tasks (Anderson and Srinivasan 2001). However, both algorithms have the same asymptotic complexity of $O(\min\{n, m \lg n\})$, where n is the number of tasks and m is the number of processors. A variant of Pfair scheduling, *early-release* scheduling, was also proposed in Anderson and Srinivasan (2000a). By considering intra-sporadic tasks, where subtasks may be released later, the same authors proposed another polynomial-time scheduling algorithm, *EPDF*, which is optimal for systems with one or two processors (Anderson and Srinivasan 2000b).

The *supertask* approach was first proposed to support non-migratory tasks in Moir and Ramamurthy (1999): tasks bound to a specific processor are combined into a single supertask which is then scheduled as an ordinary Pfair task. When a supertask is scheduled, one of its component tasks is selected for execution using EDF policy. Unfortunately, the supertask approach cannot ensure all the non-migratory component tasks to meet their deadline even when the supertask is scheduled in a Pfair manner.¹ Based on the concept of supertask, a *re-weighting* technique has been studied, which inflates a supertask's weight to ensure that its component tasks meet their deadlines if the supertask is scheduled in a Pfair manner (Holman and Anderson 2001). While

¹ It has been shown that Pfair schedules with such mapping constraints do exist (Liu and Lee 2004), however, finding the efficient scheduling algorithm to obtain such Pfair schedules remains an open problem.

this technique ensures that the supertask's non-migratory component tasks meet their deadlines, the available system utilization is reduced.

Assuming a *continuous* time domain and arbitrarily small time allocations to tasks, several optimal T-L plane based scheduling algorithms have been studied. These algorithms can also achieve full system utilization and guarantee all the timing constraints of tasks (Cho et al. 2006, 2010; Funaoka et al. 2008). More recently, by adopting the idea of *deadline partitioning*, a general scheduling model named *DP-Fair* was studied for continuous-time systems by Levin et al. (2010). In addition to periodic tasks, the authors also showed how to extend DP-Fair to sporadic task sets with unconstrained deadlines. More detailed discussions for the existing work on multiprocessor real-time scheduling can be found in the recent survey paper (Davis and Burns 2009b).

The results reported for cluster scheduling in this paper are different from the existing work, although similar techniques have been employed to derive the utilization bounds for partitioned-EDF (Darera 2006; Lopez et al. 2000, 2004). In addition, for cluster scheduling with the Bfair scheduler, an efficient *period-aware* task allocation scheme is proposed to further reduce the scheduling overhead. The effects of cluster configuration (such as different number of processors within a cluster) and the proposed period-aware task allocation heuristic on the performance of cluster scheduling are thoroughly evaluated through extensive simulations.

3 System models and problem description

In this section, we first present the system models and state our assumptions. Then, the fundamentals of cluster scheduling are reviewed, followed by the description of the problem to be addressed in this work.

3.1 System models

In this work, we consider a shared memory multiprocessor (or multicore) system with m *identical* processors/cores (which can execute any task at the *same* processing speed). A set of n periodic real-time tasks $\Gamma = \{\tau_1, \tau_2, \dots, \tau_n\}$ are to be executed on this platform. It is assumed that there is no dependency between the 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. Here, p_i is also the relative deadline of task τ_i (i.e., we consider real-time tasks with *implicit-deadlines*). Both c_i and p_i are assumed to be integers, expressed as a multiple of the system time quantum value. Each task generates a sequence of infinite number of *task instances* (or *jobs*) and the first task instances of all tasks are assumed to arrive at time 0 (i.e., tasks are synchronous). 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 $u_i = \frac{c_i}{p_i}$. The *system utilization* of a task set Γ is defined as $U(\Gamma) = \sum_{\tau_i \in \Gamma} u_i$. With the assumption that a task cannot execute in parallel on more than one processors at any time, we have $u_i \leq 1$. Moreover, the maximum utilization of all tasks is denoted as $\alpha = \max\{u_i | \tau_i \in \Gamma\}$.

For scheduling and other considerations (such as power management in multicore processors (Herbert and Marculescu 2007; Qi and Zhu 2008)), the processors can

be grouped into clusters, where each cluster has several processors. In this work, we assume that m processors are grouped into b clusters and the number of processors in the i 'th cluster is $k_i \geq 1$ ($i = 1, \dots, b$). That is, the smallest cluster has at least one processor. Moreover, we have $m = \sum_{i=1}^b k_i$.

3.2 Cluster scheduling and problem description

There are two main steps in cluster scheduling: first, tasks are partitioned/allocated to clusters; then, within each cluster, tasks are scheduled by a global scheduler (Baruah 2007; Calandrino et al. 2007; Shin et al. 2008). It can be seen that, the schedulability of a task set under cluster scheduling depends not only on the task allocation/partitioning schemes, but also on the global scheduler adopted within each cluster, in addition to the number of clusters and processors in a system.

Shin et al. (2008) carried out the schedulability analysis for a virtual cluster scheduling mechanism, where the global-EDF scheduler is adopted within each cluster. However, it is well-known that global-EDF can lead to low system utilization (Dhall and Liu 1978). The effect of cluster sizes on the utilization of a *notional processor scheduler (NPS)* is also investigated in Bletsas and Andersson (2009). To achieve the maximum system utilization, in this work, we assume that an optimal global scheduler that can fully utilize all processors is adopted within each cluster. In particular, we consider an efficient optimal global scheduler: the *boundary-fair (Bfair)* scheduling algorithm (Zhu et al. 2003) which will be reviewed in the next section.

Note that, with an optimal global scheduler in each cluster, any subset of tasks that are allocated to a cluster can be successfully scheduled provided that the aggregate utilization of these tasks is no more than the number of processors in that cluster. That is, the maximum system utilization that can be achieved under cluster scheduling in the worst-case scenario will be determined by its task allocation/partitioning scheme. Finding an optimal task partitioning scheme has been shown to be NP-hard and many simple heuristic approaches have been proposed, such as *First-Fit (FF)*, *Best-Fit (BF)* and *Worst-Fit (WF)*, which are part of the class of *reasonable task allocation schemes*. In Lopez et al. (2000, 2004), a reasonable task allocation algorithm is defined as the one that always allocates a task τ_x to one of the processors as long as there exists a processor with current available capacity no smaller than u_x .

Problem Description In this paper, assuming an optimal global scheduler within each cluster, we investigate **the utilization bound for cluster scheduling with different reasonable task allocation heuristics**. More specifically, for a given system with m processors that are grouped into b clusters, the *worst-case achievable utilization bound* for cluster scheduling with a given reasonable task allocation scheme RA is defined as a real number $U_{bound}^{RA}()$ that may depend on m , b and the number of processors in each cluster, such that:

- Any periodic task set Γ with system utilization $U(\Gamma) \leq U_{bound}^{RA}()$ is guaranteed to be schedulable under cluster scheduling with the corresponding task allocation scheme RA ;

- For any system utilization $U' > U_{bound}^{RA}()$, it is always possible to find a task set with system utilization U' , that cannot be scheduled under cluster scheduling with the reasonable task allocation scheme *RA*.

Moreover, for systems with different cluster configurations, we also study the scheduling overhead (such as the scheduler time overhead at each scheduling point, the number of context switches and task migrations) under cluster scheduling with *Bfair*.

4 Bfair scheduler and period-aware task allocation

Before presenting the utilization bounds for cluster scheduling, we first review an optimal and efficient global scheduling algorithm: *The Boundary-fair (Bfair) scheduler*. Then, a *period-aware* task allocation/partition scheme is proposed, with the objective of minimizing the scheduling overhead by reducing the number of scheduling points, context switches, and task migrations.

4.1 An efficient optimal global scheduling algorithm: Bfair

Several optimal global scheduling algorithms that can achieve full system utilization have been studied for both *continuous-time* and *discrete-time* based multiprocessor real-time systems (Sect. 1). But the continuous-time based optimal global schedulers may either require scheduling decisions at *any* time instant (Cho et al. 2006; Funaoka et al. 2008) or yield arbitrary small CPU time allocation to individual tasks (Levin et al. 2010). Moreover, the well-known discrete-time based Pfair scheduler and its variations (Anderson and Srinivasan 2000a; Buruah et al. 1995, 1996) could incur quite high scheduling overhead by making scheduling decision at every time unit, especially for systems with small time quantum. Therefore, in this work, we adopt the discrete-time based *boundary-fair (Bfair)* scheduling algorithm, which is also optimal but makes scheduling decisions (and ensures fairness to tasks) only at tasks' period boundaries (Zhu et al. 2003).

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 points as $B = \{b_0, \dots, b_f\}$, where $b_0 = 0$ and $b_j < b_{j+1}$ ($j = 0, \dots, f - 1$). For every b_j , there exists $\tau_i \in \Gamma_s$ 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 that corresponds to the LCM (least common multiple) of the 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} (Zhu et al. 2003).

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 the Bfair scheduler. 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, it was proven that all tasks can meet their deadlines under Bfair if $U(\Gamma_s) = \sum_{\tau_i \in \Gamma_s} u_i \leq k$ (i.e., the

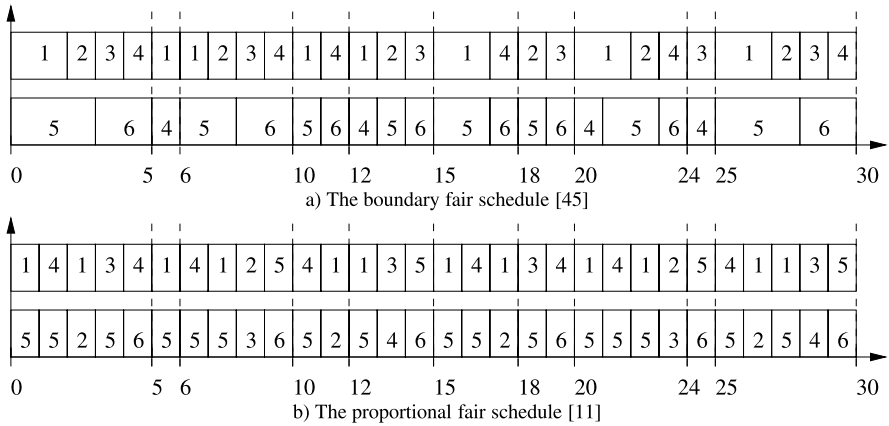


Fig. 1 The schedules for the example task set $\Gamma = \{ \tau_1(2, 5), \tau_2(3, 15), \tau_3(3, 15), \tau_4(2, 6), \tau_5(20, 30), \tau_6(6, 30) \}$ (the dotted lines denote the period boundaries)

system utilization of the subset of tasks is no greater than the number of available processors) (Zhu et al. 2003).

The detailed steps of the Bfair algorithm are omitted here and can be found in (Zhu et al. 2003). We illustrate the idea through 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 (Zhu et al. 2003), where the dotted lines in the figure are the period boundaries of the tasks. The numbers in the rectangles denote the task numbers. For comparison, the schedule obtained from the Pfair algorithm (Baruah et al. 1996) is shown in Fig. 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% (Zhu et al. 2009). Moreover, by aggregating CPU time allocation to tasks within each inter-boundary interval, Bfair can also significantly reduce the number of context switches, which occur when a processor starts executing a different task. For instance, within the first boundary interval from $t = 0$ to $t = 5$, the top processor has context switches at time 2, 3 and 4 and the bottom processor has only one context switch at time 3 in the Bfair schedule (Fig. 1a). In the above example, there are 45 context switches in the Bfair schedule and this 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% (Zhu et al. 2009). In addition, due to the same reasons, Bfair can effectively reduce the number of task migrations (which happens when a task resumes its execution on a different processor) by up to 85% (Zhu et al. 2009). Such reduction in context switches and task migrations is critical to reduce the run-time overhead in real-time systems.

Considering its optimality and low scheduling overhead, we will adopt Bfair as the global scheduler within each cluster in this work. However, we would like to point

out that the utilization bound developed for cluster scheduling in the next section is not specific to Bfair. *The use of any optimal global scheduler (e.g. Pfair variants (Anderson and Srinivasan 2000a; Buruah et al. 1995, 1996) T-L plane algorithms (Cho et al. 2006; Funaoka et al. 2008) and DP-Fair scheduler (Levin et al. 2010)) that can fully utilize all processors within each cluster will lead to the same utilization bound.*

4.2 Period-aware task allocation

From the above discussion, we can see that the scheduling points for the Bfair scheduler within each cluster are all the period boundaries of the tasks allocated to that cluster. That is, the periods of tasks allocated to a cluster have a significant effect on the scheduling overhead for cluster scheduling with the Bfair scheduler. Note that, in implementation of real systems, the system designers have often the flexibility of selecting the periods of (some subset of) tasks being harmonic (Liu 2000). In fact, there are studies that exploit the harmonicity relationships between the task periods to improve the schedulability under rate-monotonic scheduling (Andersson and Jonsson 2003; Burchard et al. 1996; Lauzac et al. 2003). Along the same lines, but this time with the objective of minimizing the scheduling overhead, we propose a *period-aware* task allocation scheme, which exploits the *harmonicity* of tasks' periods to reduce the scheduling overhead in cluster scheduling with Bfair.

Specifically, 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. 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, we propose the *period-aware first-fit (PA-FF)* task allocation scheme, as a variant of the *first-fit* heuristic. The detailed steps of the *PA-FF* scheme are shown in Algorithm 1. First, the task with the smallest period p_x is put in a task set (*HarmonicSet*), which is followed by tasks with periods harmonic with p_x (lines 4 to 17). Then, all tasks in the harmonic task set are added to an ordered task list (*LIST*) in the order of increasing periods of tasks (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 following the first-fit heuristic (line 20). The superiority of the new PA-FF task allocation scheme over the simple first-fit heuristic on reducing the scheduling overhead (e.g., the number of scheduling points, context switches and task migrations) of cluster scheduling with Bfair is evaluated and discussed in Sect. 6.

5 Utilization bounds for cluster scheduling with optimal global schedulers

As mentioned earlier, with an optimal global scheduler within each cluster, the tasks that are allocated to one cluster will be schedulable as long as their aggregate utilization is no more than the number of processors within that cluster. Therefore, the task

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

```

1: Set  $p_{\max} = \max_i^n p_i$ ;
2: Initialize an ordered task list:  $LIST = \emptyset$ ;
3: while (Task set  $\Gamma \neq \emptyset$ ) do
4:   Find a task  $\tau_x$  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 ( $\tau_i$  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 ordered list  $LIST$  to clusters with First-Fit heuristic;

```

allocation/partitioning phase of cluster scheduling becomes essentially a *bin-packing* problem of how to successfully pack n items into b bins, where the size of each item is no more than α (≤ 1) and each bin has a size of k_i ($i = 1, \dots, b$). 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 deriving the *utilization bound* that can be used in efficient schedulability tests. That is, any task set with system utilization no greater than the bound should be guaranteed to be schedulable under cluster scheduling.

It is easy to see that, for cluster scheduling with optimal global schedulers, the utilization bound will depend on its task allocation/partitioning scheme as well as the number of clusters and processors in the system under consideration. Recall that, the maximum task utilization is denoted by α (≤ 1). For a cluster with k_i processors, the *minimum* number of tasks that fit in that cluster is given by $\beta_i = \lfloor \frac{k_i}{\alpha} \rfloor$. Similarly, for a system with b clusters where the i 'th cluster has k_i processors, the minimum number of tasks that can be handled by that system can be found as $\beta_{sum} = \sum_{i=1}^b \beta_i$. That is, if the number of tasks in a task set satisfies $n \leq \beta_{sum}$, regardless of the system utilization and (reasonable) task allocation scheme, all tasks are guaranteed to be schedulable under cluster scheduling with optimal global schedulers. Therefore, in what follows, we assume that $n > \beta_{sum}$.

In partitioned scheduling, several well-known partition heuristics have been studied, such as *first-fit* (FF), *best-fit* (BF) and *worst-fit* (WF) (Lopez et al. 2000, 2004), which can also be adopted to allocate tasks to clusters in cluster scheduling. Note that, these partitioning heuristics have different features (e.g., WF typically yields better

balanced partitions, while FF and BF provide better system utilization), which will lead to different utilization bounds. Following (Lopez et al. 2000, 2004), we define a task allocation scheme for cluster scheduling to be *reasonable* if it always performs allocation of a task (to a cluster) whenever the utilization of the task does not exceed the available remaining capacity of at least one cluster. More specifically, suppose that the subset of tasks that have been allocated to the i 'th cluster ($i = 1, \dots, b$) is Γ_i and the next task to be allocated is τ_x . A *reasonable allocation (RA)* scheme will allocate the task τ_x to one cluster provided that there exists a cluster such that $u_x \leq k_i - U(\Gamma_i)$, where $U(\Gamma_i) = \sum_{\tau_j \in \Gamma_i} (u_j)$. Note that, all the above mentioned heuristics (WF, FF and BF) qualify as reasonable allocation schemes.

In what follows, following the similar reasonings as those for the utilization bounds of partitioned-EDF scheduling in uniprocessor systems (Darera 2006; Lopez et al. 2000, 2004), we first derive a *lower limit* on the utilization bounds for cluster scheduling with any reasonable task allocation scheme. Then, the exact bound for the WF allocation heuristic is presented. Finally, the bounds for other partitioning heuristics are obtained, for both *homogeneous* and *heterogeneous* clusters (that have the same and different number of processors, respectively).

5.1 Lower limit for the utilization bounds

For a system with a given cluster configuration (defined by m , b and k_i , where $i = 1, \dots, b$), the utilization bound for cluster scheduling with a *reasonable allocation* of tasks to clusters is denoted as $U_{bound}^{RA}(m, b, k_1, \dots, k_b, \alpha)$. In this section, we will derive a lower limit U_{bound}^{low} for such a utilization bound. That is, for cluster scheduling with *any* reasonable task allocation scheme, the utilization bound will satisfy $U_{bound}^{RA}(\alpha) \geq U_{bound}^{low}$.

Theorem 1 *For a real-time system with m processors that are grouped into b clusters, where the i 'th cluster has k_i processors, if the number of tasks $n > \beta_{sum}$, the utilization bound for cluster scheduling with any reasonable task allocation scheme $U_{bound}^{RA}(\alpha)$ will satisfy the following expression:*

$$U_{bound}^{RA}(m, b, k_1, \dots, k_b, \alpha) \geq U_{bound}^{low} = m - (b - 1) \cdot \alpha \quad (1)$$

Proof The proof follows the similar approach as that for Theorem 1 in Lopez et al. (2004). That is, we will prove the theorem by showing that, for any task set Γ that is not schedulable under cluster scheduling with any reasonable task allocation scheme, $U(\Gamma) > U_{bound}^{low}$ holds.

Without loss of generality, suppose that $\Gamma = \{\tau_1, \dots, \tau_n\}$ and tasks are allocated to clusters in the increasing order of tasks' indices. Let task τ_j ($j \leq n$) be the first task that cannot fit into any of the clusters. We have:

$$u_j > k_i - U(\Gamma_i), \quad i = 1, \dots, b \quad (2)$$

where Γ_i is the subset of tasks that have been allocated to the i 'th cluster and $U(\Gamma_i)$ denotes the aggregate utilization of tasks in Γ_i . Note that, the first $(j - 1)$ tasks have

been successfully allocated to the clusters. That is,

$$\sum_{x=1}^{j-1} u_x = \sum_{i=1}^b U(\Gamma_i)$$

Therefore, we have:

$$u_j + \sum_{i=1}^b U(\Gamma_i) = \sum_{x=1}^j u_x \leq \sum_{x=1}^n u_x = U(\Gamma) \tag{3}$$

From (2), we can get

$$\sum_{i=1}^b U(\Gamma_i) > \sum_{i=1}^b (k_i - u_j) = \sum_{i=1}^b k_i - \sum_{i=1}^b u_j = m - b \cdot u_j$$

Substituting the above inequality into (3), we have:

$$U(\Gamma) > u_j + m - b \cdot u_j = m - (b - 1) \cdot u_j \tag{4}$$

By definition, α is the maximum task utilization. That is, $u_j \leq \alpha$. Hence:

$$U(\Gamma) > m - (b - 1) \cdot \alpha \tag{5}$$

That is, for any task set Γ that cannot be scheduled by cluster scheduling, the above inequality holds. Consequently, any task set with system utilization $\leq m - (b - 1)\alpha$ should be schedulable under cluster scheduling, concluding the proof. \square

From Theorem 1, we can see that, regardless of whether the number of processors within each cluster is the same (*homogeneous clusters*) or not (*heterogeneous clusters*), the *lowest achievable* utilization bound for cluster scheduling with reasonable allocation schemes depends only on the maximum task utilization, the number of clusters and the total number of processors. For a system with a fixed number of processors, when the number of clusters increases, the lowest achievable utilization bound for cluster scheduling decreases.

5.2 Utilization bound for the worst-fit task allocation scheme

In this section, we show that the lower limit derived in the above section is also the upper limit on the utilization bounds for cluster scheduling with the worst-fit task allocation. That is, it is the actual utilization bound for cluster scheduling with the worst-fit task allocation. Suppose that, during task allocation, the remaining capacity on the i 'th cluster is $R_i = k_i - U(\Gamma_i)$, where k_i is the number of processors in the cluster and Γ_i represents the subset of tasks that have been allocated to the cluster. In the worst-fit task allocation, tasks τ are allocated to clusters sequentially and a task τ_j is always allocated to the cluster with the largest remaining capacity $R_x = \max_{i=1}^b \{R_i\}$, provided that $R_x \geq u_j$.

Theorem 2 For a real-time system with m processors that are grouped into b clusters, where the i 'th cluster has k_i processors, if the number of tasks $n > \beta_{sum}$, the utilization bound for cluster scheduling with worst-fit task allocation $U_{bound}^{WF}()$ is:

$$U_{bound}^{WF}(m, b, \alpha) = m - (b - 1) \cdot \alpha \quad (6)$$

Proof We will prove the theorem by showing that $m - (b - 1) \cdot \alpha$ is also the upper limit on the utilization bound for cluster scheduling with the worst-fit task allocation. That is, we will prove the existence of a set of $n (> \beta_{sum})$ tasks with the system utilization of $m - (b - 1) \cdot \alpha + \varepsilon$ (where $\varepsilon \rightarrow 0^+$) that cannot fit into b clusters under the worst-fit task allocation algorithm.

The proof adopts similar techniques as those used in the proof of Theorem 3 in Lopez et al. (2004), with three parts: (1). The task set is first constructed; (2). The task set is shown to be valid (i.e., for any task τ_i in the set, $u_i \leq \alpha$ holds); (3). Finally, it is shown that the task set cannot fit into the clusters under the worst-fit task allocation scheme.

Part 1: The task set with n tasks is constructed as follows. First, we construct b subset of tasks, where the i 'th subset Γ_i has β_i tasks and the total number of tasks is $\beta_{sum} = \sum_{i=1}^b \beta_i$. The tasks in the i 'th subset Γ_i , have the same task utilization equal to:

$$u_j = \frac{k_i - \alpha}{\beta_i} + \frac{\varepsilon}{2 \cdot \beta_{sum}}, \quad \tau_j \in \Gamma_i$$

An additional set Γ_{b+1} of $(n - \beta_{sum} - 1)$ tasks are constructed where the tasks have identical utilization as $\frac{\varepsilon}{2 \cdot (n - \beta_{sum} - 1)}$. The last task has the utilization of α . Hence, there are $\beta_{sum} + (n - \beta_{sum} - 1) + 1 = n$ tasks in the task set Γ .

Part 2: From Part 1, we can find that the system utilization of the task set Γ is $U(\Gamma) = m - (b - 1) \cdot \alpha + \varepsilon$. Since $\varepsilon \rightarrow 0^+$, for any task τ_j in the set Γ_{b+1} , $u_j \leq \alpha$. By definition, for the last task τ_n , we have $u_n \leq \alpha$.

For any task τ_j in the subset Γ_i , from the definition of $\beta_i = \lfloor \frac{k_i}{\alpha} \rfloor$, we have:

$$\beta_i > \frac{k_i}{\alpha} - 1 = \frac{k_i - \alpha}{\alpha}$$

That is, $\frac{k_i - \alpha}{\beta_i} < \alpha$. Since $\varepsilon \rightarrow 0^+$, we have:

$$u_j = \frac{k_i - \alpha}{\beta_i} + \frac{\varepsilon}{2 \cdot \beta_{sum}} < \alpha$$

Therefore, the task utilization of any task constructed in Part 1 is no more than α and the task set is valid.

Part 3: In what follows, we show that it is always possible to order the tasks in Γ , in such a way that the worst-fit technique will fail to generate a feasible allocation of the tasks on b clusters.

This special task order Θ will have the following features: In Θ , first the tasks in the b subsets Γ_i ($i = 1, \dots, b$) will appear, followed by the tasks in the subset Γ_{b+1} with identical utilizations, and finally, by the task τ_n with utilization $u_n = \alpha$. Moreover, when tasks in Γ are processed by the worst-fit technique in the order given by Θ , the tasks in the i 'th subset Γ_i will be allocated to the i 'th cluster that has k_i processors. We will show that (i) it is always possible to come up with such a task order Θ , and, (ii) when the worst-fit heuristic processes the tasks in the order Θ , the last task τ_n cannot fit in any of the clusters.

To show (i) above, we will provide a deterministic algorithm to generate the specific order for tasks in $\Gamma_1, \dots, \Gamma_b$ in Θ (because the position of tasks in Γ_{b+1} and τ_n are already fixed). Then, we will prove the statement given in (ii) above.

For (i), first note that, under the worst-fit allocation, the next task in Θ (regardless of its utilization) will be allocated to the cluster that has the largest remaining capacity. For cases where more than one clusters have the same largest remaining capacity, we assume that the task is allocated to the cluster with smaller index. Then the specific order for tasks in $\Gamma_1, \dots, \Gamma_b$ in Θ can be found through the following 3-step procedure:

- *Step 1:* The remaining capacity for the i 'th cluster is initialized as $RC_i = k_i$; $i = 1, \dots, b$. An ordered task list is initialized to be $LIST = \emptyset$.
- *Step 2:* Find the maximum remaining capacity among all clusters, that is, $RC_{\max} = \max\{RC_i | i = 1, \dots, b\}$. Suppose that $RC_x = RC_{\max}$. If there are more than one clusters that have the same maximum remaining capacity RC_{\max} , x is the smallest index of all such clusters. Remove one task from the x 'th subset and add it to the end of the ordered task list $LIST$. Recall that all tasks in the x 'th subset have the same utilization u_x . Update $RC_x = RC_x - u_x$.
- *Step 3:* If there is any subset that still contains task(s), repeat the above Step 2.

Note that, the second step in the above procedure relies on an implicit condition: *When the above procedure is applied to the tasks in $\Gamma_1, \dots, \Gamma_b$, if there are tasks that have not been added to the ordered list and the x 'th cluster has the largest remaining capacity RC_x , the subset Γ_x should contain at least one task.* Next, we prove that such a condition always holds through contradiction. That is, suppose that the x 'th cluster has the largest remaining capacity RC_x during the execution of the above procedure, Γ_x is empty and there exists another subset Γ_y ($y \neq x$) that still contains task(s), there will be a contradiction as we show in what follows.

For the tasks that are already in the ordered list $LIST$, after allocating them to the clusters under the worst-fit allocation, we know that all tasks in Γ_x will be allocated to the x 'th cluster. That is, the remaining capacity of the x 'th cluster is:

$$RC_x = k_x - U(\Gamma_x) = k_x - \beta_x \left(\frac{k_x - \alpha}{\beta_x} + \frac{\varepsilon}{2 \cdot \beta_{sum}} \right) = \alpha - \frac{\beta_x \cdot \varepsilon}{2 \cdot \beta_{sum}}$$

Note that, at least one task in the subset Γ_y is not in the ordered list yet and at most $(\beta_y - 1)$ tasks in Γ_y can be allocated to the y 'th cluster. Therefore, we have

$$RC_y \geq k_y - (\beta_y - 1) \cdot \left(\frac{k_y - \alpha}{\beta_y} + \frac{\varepsilon}{2 \cdot \beta_{sum}} \right)$$

$$\begin{aligned}
 &= k_y - \beta_y \cdot \left(\frac{k_y - \alpha}{\beta_y} + \frac{\varepsilon}{2 \cdot \beta_{sum}} \right) + \left(\frac{k_y - \alpha}{\beta_y} + \frac{\varepsilon}{2 \cdot \beta_{sum}} \right) \\
 &= \alpha - \frac{\beta_y \cdot \varepsilon}{2 \cdot \beta_{sum}} + \left(\frac{k_y - \alpha}{\beta_y} + \frac{\varepsilon}{2 \cdot \beta_{sum}} \right)
 \end{aligned}$$

Since $\varepsilon \rightarrow 0^+$, we can see that $RC_y > RC_x$, which contradicts with the assumption that the maximum remaining capacity is RC_x . Therefore, the condition always holds (and thus the second step can complete) during the execution of the above procedure.

Thus the order Θ consists of tasks in $\Gamma_1, \dots, \Gamma_b$ as yielded by the 3-step procedure above, followed by the tasks in Γ_{b+1} and τ_n . Now we show that with this specific order Θ , the worst-fit heuristic will not be able to allocate the last task τ_n to any of the clusters.

Note that, under the above procedure, all tasks in the first b subsets can be added to the ordered list. Moreover, when the tasks are allocated to clusters under the worst-fit allocation in the order Θ , all tasks in the i 'th subset Γ_i will be allocated to the i 'th cluster ($i = 1, \dots, b$). After allocating all tasks in $\Gamma_1, \dots, \Gamma_b$, the remaining capacity for the i 'th cluster will be

$$RC_i = k_i - \beta_i \left(\frac{k_i - \alpha}{\beta_i} + \frac{\varepsilon}{2 \cdot \beta_{sum}} \right) = \alpha - \frac{\beta_i \cdot \varepsilon}{2 \cdot \beta_{sum}} < \alpha$$

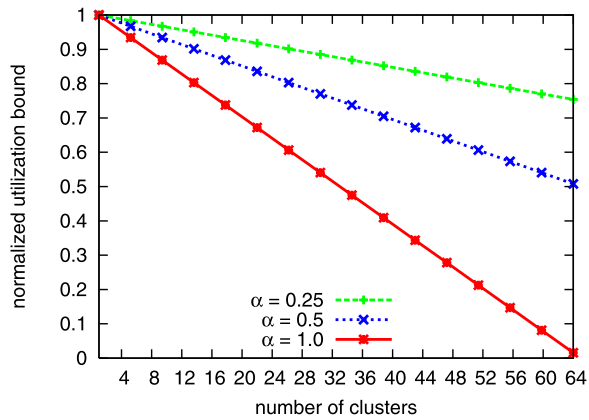
That is, after further allocating the tasks in the subset Γ_{b+1} to the clusters as in the order Θ , the remaining capacity of any cluster is less than α and the last task τ_n that has the utilization of α cannot fit into any of the clusters.

Therefore, $U_{bound}^{WF}() \leq m - (b - 1) \cdot \alpha$. Note that, from Theorem 1, we have also $U_{bound}^{WF}() \geq m - (b - 1) \cdot \alpha$. Thus, $U_{bound}^{WF}() = m - (b - 1) \cdot \alpha$, which concludes the proof. □

From Theorem 2, we can see that the utilization bound for cluster scheduling with the worst-fit task allocation scheme does *not* depend on the number of processors within each individual cluster. Instead, it depends only on the total number of processors and the number of clusters in the system as well as the maximum task utilization for the tasks under consideration.

For a system with 64 processors, Fig. 2 shows the normalized utilization bound (i.e. $\frac{U_{bound}^{WF}(64, b, \alpha)}{64}$) for the worst-fit task allocation when the processors are organized into different number of clusters. We can see that, when there are more clusters, the utilization bound becomes lower as more segmented processor capacity can be wasted in the worst-case scenario. Moreover, with increasing maximum task utilization α , the utilization bound decreases due to larger segmented processor capacity wasted in each cluster. For the case where each processor forms a cluster and $\alpha = 1$, the utilization bound for the worst-fit allocation can be extremely small (as low as 1 regardless of the number of processors in the system), which coincides with previous results on partitioned scheduling with the worst-fit task allocation scheme (Dhall and Liu 1978; Lopez et al. 2004).

Fig. 2 Normalized utilization bounds for the worst-fit allocation in a 64-processor system



5.3 Utilization bound for reasonable allocation decreasing schemes

To improve the utilization bound, instead of allocating tasks in the random order, we can allocate tasks in the decreasing order of their utilizations (Lopez et al. 2004). In what follows, for such *reasonable allocation decreasing (RAD)* schemes, which include *first-fit decreasing (FFD)*, *best-fit decreasing (BFD)* and *worst-fit decreasing (WFD)*, we will develop higher utilization bounds for cluster scheduling with optimal global schedulers. We first consider the case with *homogeneous clusters* (where all clusters have the same number of processors) in Sect. 5.3.1. The utilization bound for *heterogeneous clusters* (where clusters have different number of processors) is studied in Sect. 5.3.2.

5.3.1 The case for homogeneous clusters

In this section, we consider the case with homogeneous clusters where each cluster has the same number of processors. That is, $k_i = k$ ($i = 1, \dots, b$) and $m = b \cdot k$. Note that, when $k = 1$ (i.e., there is only one processor within each cluster), the problem can be reduced to that of finding the utilization bound for the partitioned-EDF scheduling, which has been studied by Lopez et al. (2000, 2004). Specifically, for reasonable allocation decreasing (RAD) heuristics, the utilization bound for the partitioned-EDF is given as (Lopez et al. 2000, 2004):

$$U_{bound}^{partition-EDF}(m, \alpha) = \frac{\beta \cdot m + 1}{\beta + 1} \tag{7}$$

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 when all tasks have the maximum task utilization α (≤ 1). When $\alpha = 1$, the bound reduces to $(m + 1)/2$. That is, for systems with large number of processors and $\alpha = 1$, partitioned-EDF can only achieve around 50% of the system utilization in the worst-case scenario.

For the cases where each cluster has more processors, by extending the result in (7), we can obtain the following theorem regarding the utilization bound for cluster scheduling with reasonable allocation decreasing schemes.

Theorem 3 For a real-time system with m processors that are grouped into b clusters with each cluster having k processors (i.e., $m = b \cdot k$), if an optimal global scheduler is adopted within each cluster and a reasonable allocation decreasing (e.g., FFD, BFD or WFD) heuristic is used, the worst-case utilization bound is:

$$U_{bound}^{RAD}(b, k) = \frac{k \cdot b + 1}{k + 1} \cdot k \tag{8}$$

Proof We will prove the theorem by transforming the problem of cluster scheduling to that of partitioned-EDF scheduling problem. Recall that the necessary and sufficient condition for a set of tasks to be schedulable on a uniprocessor system under EDF scheduling is the aggregate utilization of the tasks being no more than one (Liu and Layland 1973).

Therefore, for any problem of scheduling a set of tasks $\Gamma = \{\tau_i | i = 1, \dots, n\}$ on b clusters each with k processors under cluster scheduling, we can construct a corresponding partitioned-EDF scheduling problem with a set of tasks $\Gamma' = \{\tau'_i | i = 1, \dots, n\}$ and b 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 $i = 1, \dots, n$.

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

Note that, for every task $\tau_i \in \Gamma$, we have $u_i \leq 1$ ($i = 1, \dots, n$). Therefore, the maximum task utilization for tasks in Γ' will be $\alpha' = \max\{u'_i | i = 1, \dots, n\} = \max\{\frac{u_i}{k} | i = 1, \dots, n\} \leq \frac{1}{k}$. From (7), we know that, for any of the reasonable allocation decreasing heuristics, the task set Γ' can be successfully allocated on b processors if the system utilization of Γ' is $U' = \sum_i^n u'_i \leq \frac{\lfloor (1/\alpha') \rfloor \cdot b + 1}{\lfloor (1/\alpha') \rfloor + 1} \leq \frac{k \cdot b + 1}{k + 1}$.

Hence, for the original task set Γ , the same task allocation scheme can allocate all tasks on b clusters each with k processors if the system utilization $U = \sum_i^n u_i = k \sum_i^n u'_i = k \cdot U' \leq \frac{k \cdot b + 1}{k + 1} k$, which concludes the proof. \square

Following the same reasoning, if the maximum task utilization in task set Γ is α (≤ 1), the utilization bound can be generalized as:

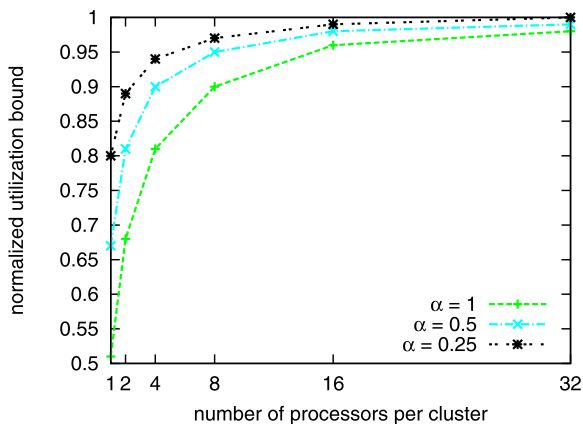
$$U_{bound}^{RAD}(b, k, \alpha) = \frac{\beta \cdot b + 1}{\beta + 1} \cdot k \tag{9}$$

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 the utilization as α . In Lopez et al. (2000, 2004), it is shown that the utilization bounds for FF and BF coincide with those for the reasonable allocation decreasing heuristics. Therefore, following the same reasoning as in proof of Theorem 3, we have

$$U_{bound}^{FF}(b, k, \alpha) = U_{bound}^{BF}(b, k, \alpha) = \frac{\beta \cdot b + 1}{\beta + 1} \cdot k \tag{10}$$

Note that, if the cluster size is one (i.e., $k = 1$ and $b = m$), the bounds given in the two equations above are reduced to the worst-case utilization bounds for partitioned-

Fig. 3 Utilization bounds for cluster scheduling with RAD schemes in a 64-processor system that is configured with homogeneous clusters



EDF scheduling studied by Lopez et al. (2000, 2004). Moreover, if all processors belong to one cluster (i.e., $b = 1$ 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 observe 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. To illustrate such effects, Fig. 3 shows the bound for a 64-processor system with different organizations of the clusters. The X -axis represents the number of processors per cluster (i.e., k) and the Y -axis depicts the normalized utilization bound (which is defined as $U_{bound}^{RAD}()/64$).

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 with 16 processors each, the normalized utilization bound is 0.95, while the bound is only 0.8 for the organization of 16 clusters each with 4 processors. This is because the smaller the clusters, the more significant is the utilization loss due to fragmentation caused during the course of partitioning algorithm. Moreover, the effect of α on the utilization bound is more prominent for smaller clusters. However, as shown in Sect. 6, it can be beneficial to organize the processors as smaller clusters to reduce scheduling overhead. That is, for smaller clusters, there will be relatively fewer number of the tasks within each cluster and the number of scheduling points (i.e., period boundaries) can be significantly reduced for cluster scheduling with the Bfair scheduler, leading to lower scheduling overhead.

5.3.2 The case for heterogeneous clusters

Due to resource and/or power management consideration, the clusters in a system may have different number of processors (Herbert and Marculescu 2007; Qi and Zhu 2008). For cluster scheduling with reasonable allocation decreasing schemes, we derive the utilization bound for systems with heterogeneous clusters in

this section. The main idea is similar to the one used in the derivation of the utilization bounds for the partitioned-EDF scheduler in systems with *non-identical uniform* processors (which have the same functionalities but different processing speeds), which have been studied by Darera (2006).

Theorem 4 *For a real-time system with m processors that are grouped into b clusters, where the i 'th cluster has k_i (≥ 1) processors and $m = \sum_{i=1}^b k_i$, if an optimal global scheduler is adopted within each cluster and a reasonable allocation decreasing (e.g., FFD, BFD or WFD) partitioning heuristic is employed, the utilization bound $U_{bound}^{RAD}()$ satisfies the following expression:*

$$U_{bound}^{RAD}(m, b, k_1, \dots, k_b, \alpha) \geq \frac{m \cdot (\beta_{sum} + 1)}{\beta_{sum} + b} \tag{11}$$

where $\beta_{sum} = \sum_{i=1}^b \beta_i$ and $\beta_i = \lfloor \frac{k_i}{\alpha} \rfloor$.

Proof The proof has the steps similar to those of Theorem 1. That is, we will show that, for any task set Γ that is not schedulable in the system that has heterogeneous clusters under cluster scheduling with any reasonable allocation decreasing scheme, the following inequality holds: $U(\Gamma) > \frac{m \cdot (\beta_{sum} + 1)}{\beta_{sum} + b}$.

Consider a set of tasks that cannot be scheduled $\Gamma = \{\tau_1, \tau_2, \dots, \tau_j, \dots, \tau_n\}$. We assume that $u_{j-1} \geq u_j$ for $1 < j \leq n$. Suppose that task τ_j ($j \leq n$) is the first task that cannot be allocated to any of the b clusters after successfully allocating the first $(j - 1)$ tasks. That is, the remaining capacity of any cluster is less than the task τ_j 's utilization u_j :

$$k_i - U(\Gamma_i) < u_j, \quad i = 1, \dots, b$$

where Γ_i presents the subset of tasks that have been allocated to the i 'th cluster. We have

$$U(\Gamma_i) > k_i - u_j, \quad i = 1, \dots, b$$

$$\sum_{i=1}^b U(\Gamma_i) > \sum_{i=1}^b k_i - b \cdot u_j = m - b \cdot u_j$$

Note that, the aggregate utilization of the first j tasks is:

$$\sum_{x=1}^j u_x = \sum_{i=1}^b U(\Gamma_i) + u_j$$

From the above two equations, we get

$$\sum_{x=1}^j u_x > m - (b - 1)u_j$$

Since tasks are allocated in the decreasing order of their utilization, we have $\sum_{x=1}^j u_x \geq j \cdot u_j$. That is,

$$\frac{\sum_{x=1}^j u_x}{j} \geq u_j.$$

Therefore, we have

$$\sum_{x=1}^j u_x > m - (b - 1) \frac{\sum_{x=1}^j u_x}{j}$$

After re-arranging the above equation, we get:

$$\sum_{x=1}^j u_x > \frac{j \cdot m}{j + b - 1}$$

Note that, with the maximum task utilization α , the minimum number of tasks that can fit into the i 'th cluster is $\beta_i = \lfloor \frac{k_i}{\alpha} \rfloor$ and the minimum number of tasks that are guaranteed to be schedulable in the system is $\beta_{sum} = \sum_{i=1}^b \beta_i$. Therefore, we have $j \geq \beta_{sum} + 1$. Combining this with the fact that the right side of the above inequality increases monotonically with increasing j , we obtain:

$$U(\Gamma) = \sum_{x=1}^n u_x \geq \sum_{x=1}^j u_x > \frac{j \cdot m}{j + b - 1} \geq \frac{m \cdot (\beta_{sum} + 1)}{\beta_{sum} + b} \tag{12}$$

Therefore, any set of tasks with utilization $\leq \frac{m \cdot (\beta_{sum} + 1)}{\beta_{sum} + b}$ is schedulable under cluster scheduling with a reasonable allocation decreasing scheme in a system with heterogeneous clusters, which concludes the proof. \square

From Theorem 4, we can see that, in addition to the maximum task utilization and the number of processors and clusters, the utilization bound of cluster scheduling in systems with heterogeneous clusters depends on the number of processors within each individual cluster (i.e., β_i). Note that, when clusters have the same number of processors (i.e., homogeneous clusters), the bound in (11) will reduce to the one given in (9).

6 Evaluations and discussions

Note that the utilization bounds derived in the preceding section correspond to the worst-case scenario. That is, it is still possible to have task sets *schedulable* with utilizations larger than the given bounds. In this section, we will empirically evaluate the performance of different task allocation heuristics as well as the effects of different cluster configurations in a system with a given number of processors, in terms of *success ratio* of synthetically generated task sets (which is defined as the ratio of the number of schedulable task sets over the total number of task sets generated).

Moreover, the performance of the proposed *period-aware first-fit (PA-FF)* task allocation heuristic on reducing the *scheduling overhead* (in terms of invocation time at each scheduling point, the number of context switches and task migrations) is also evaluated for cluster scheduling with the Bfair global scheduler through extensive simulations. For comparison, the simple first-fit decreasing (FFD) heuristic is also considered.

In the simulations, synthetic task sets are generated from a set of parameters: the system utilization of task sets U_{tot} , the maximum task utilization α , the minimum task period P_{min} and the maximum task period P_{max} . In this section, we consider systems with $m = 16$ and $m = 64$ processors, separately. 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 varies in 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 value is adjusted so as not to exceed α . Additional tasks are generated iteratively provided that their cumulative utilization does not exceed 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}). Therefore, the number of tasks in a task set depends on both its system utilization U_{tot} and the maximum task utilization α . For task sets with the same system utilization, more tasks are generally contained in the set with smaller value of α .

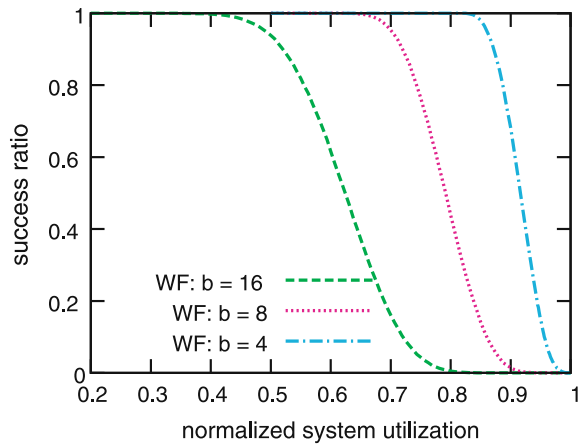
6.1 Success ratio

6.1.1 Worst-fit allocation

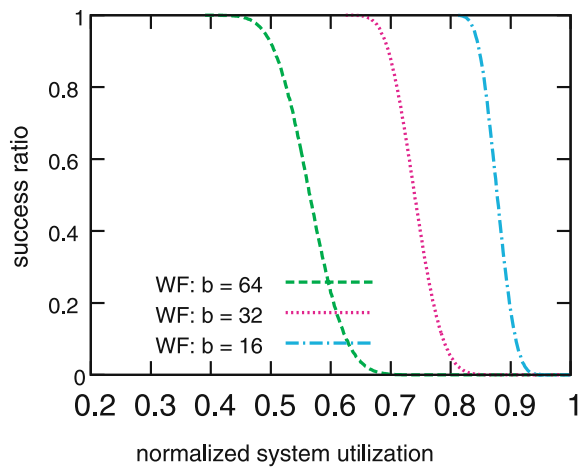
Recall from Sect. 5.2 that the utilization bound for cluster scheduling with worst-fit task allocation heuristic can be very low, especially for systems with large number of clusters. In this section, for task sets with different system utilizations, we first evaluate the number of task sets that can be successfully scheduled under cluster scheduling with Bfair and worst-fit allocation heuristic. Here, the maximum task utilization is set to $\alpha = 1.0$ and 1,000,000 task sets are generated for each setting following the aforementioned steps.

Figures 4a and 4b show the success ratio of the synthetic task sets for systems with 16 and 64 processors, respectively. Here, the X-axis represents the *normalized system utilization* (i.e., $\frac{U_{tot}}{m}$) of the task sets under consideration. From the figures, we can see that, a large number of task sets with system utilization well above the utilization bound are schedulable. For instance, with $\alpha = 1.0$, the *normalized utilization bound* for the system with 16 processors and 16 cluster is $\frac{U_{bound}^{wf}(16,16,1.0)}{16} = \frac{1}{16} = 0.0625$. However, from Fig. 4a, we can see that almost all task sets with normalized system utilization ≤ 0.4 are schedulable and more than 95% of the task sets with normalized system utilization 0.5 are schedulable. As the number of clusters decreases, the utilization bound for the worst-fit allocation increases and more task sets with higher

Fig. 4 Success ratios for cluster scheduling with WF task allocation; $\alpha = 1.0$



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



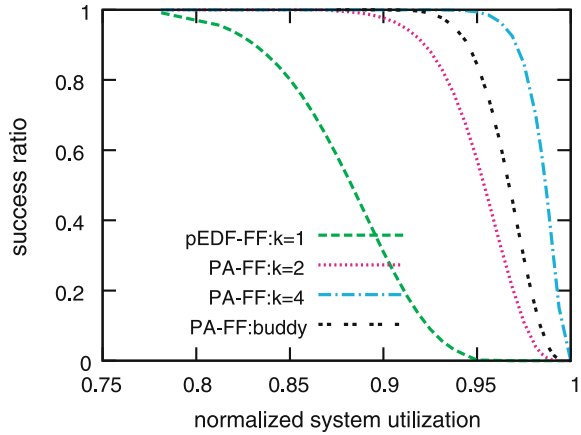
(b) 64-processor systems, $m = 64$

normalized system utilization become schedulable. For systems with 64 processors, similar results are obtained as shown in Fig. 4b. For the same number of clusters (e.g., $b = 16$), more task sets with higher normalized system utilizations are schedulable for systems with more processors, which is consistent with the utilization bound whose value increases with the number of processors when the number of clusters is fixed.

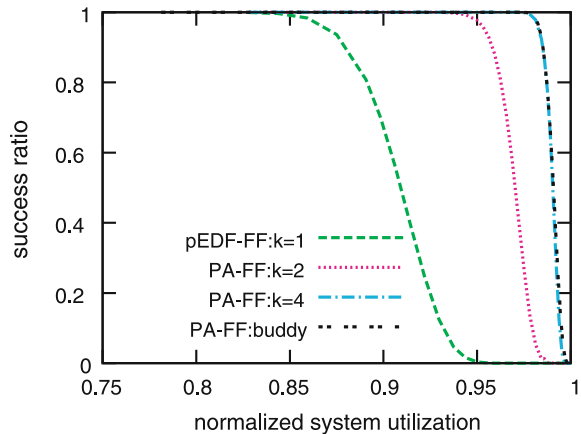
6.1.2 Period-aware first-fit allocation

Recall that the utilization bounds for reasonable allocation decreasing (RAD) heuristics (including FFD, BFD and WFD) are the same for systems with both homogeneous and heterogeneous clusters (see Sect. 5.3). Moreover, for systems with homogeneous clusters, the bounds for FF and BF heuristics are the same as that for

Fig. 5 Success ratios for cluster scheduling with PA-FF allocation; $\alpha = 1.0$



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



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

the RAD heuristics. In this section, we show the success ratio of the synthetic task sets for our newly proposed *period-aware first-fit (PA-FF)* scheme, a variant of FF heuristic. For comparison, the success ratio of the task sets under partitioned-EDF scheduling with first-fit heuristic (pEDF-FF) is also presented (Lopez et al. 2000, 2004). The results for RAD heuristics are very close to that of PA-FF for the same cluster configuration, and are not shown in the figures for clarity.

As before, we consider systems with 16 and 64 processors. The success ratios of task sets with different normalized system utilizations are shown in Figs. 5a and 5b, respectively. Here, for systems with homogeneous clusters, only the results for cluster size of $k = 2$ and $k = 4$ are presented. For larger clusters with more processors each, the success ratio of task sets under cluster scheduling with PA-FF is almost 1 even for very high normalized system utilization. Moreover, when the system is configured with heterogeneous clusters, there are many different possible cluster configurations. In this work, we consider one special *buddy* heterogeneous cluster configuration fol-

lowing the similar idea of buddy memory allocation in operating systems. Here, the first cluster has 1/2 of the processors, the second cluster has 1/2 of the remaining processors, and so on. For instance, for systems with 16 processors, there are five clusters, with 8, 4, 2, 1 and 1 processor(s) each, respectively.

First, from Fig. 5a, we can see that, the cluster scheduling can successfully schedule almost all task sets that have the normalized system utilization ≤ 0.87 even when the 16-processor system is configured as 8 homogeneous clusters of size two (i.e., $k = 2$). Not surprisingly, the cluster scheduling performs better with larger clusters (and fewer number of 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 0.95 and almost all task sets are still schedulable. In contrast, the partitioned-EDF with the first-fit heuristic (denoted as pEDF-FF) can only schedule almost all task sets when the normalized system utilization of task sets does not exceed 0.77. For the case of heterogeneous buddy cluster configuration with 5 clusters, the number of successfully scheduled task sets falls between that for homogeneous clusters of $k = 2$ (with 8 clusters) and $k = 4$ (with 4 clusters). Almost all task sets with normalized system utilization ≤ 0.92 are schedulable.

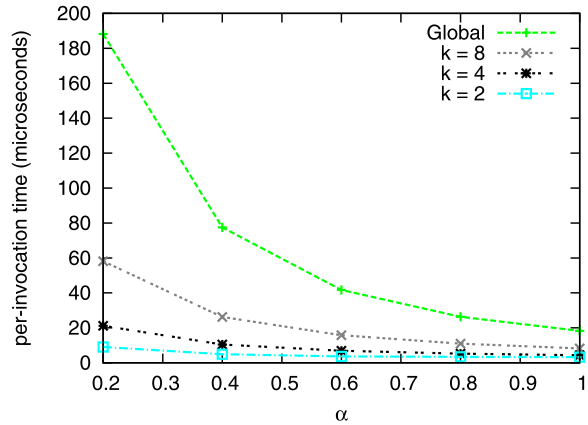
When the maximum task utilization becomes smaller (e.g., $\alpha = 0.5$), better success ratios have been obtained for both PA-FF and pEDF-FF. The results follow the similar pattern as that for $\alpha = 1.0$ and are not shown in the paper due to space limitations. However, it is worth noting that the cluster scheduling with PA-FF still outperforms partitioned-EDF with first-fit heuristic. Moreover, for systems with more processors, more clusters can be formed and tasks have more chance to fit into one of the clusters. For instance, for systems with 64 processors, Fig. 5b shows that both the cluster scheduling with PA-FF and partitioned-EDF with FF perform better and can schedule more task sets with higher normalized system utilization. Furthermore, PA-FF performs almost same for systems with 16 homogeneous clusters each having 4 processors (i.e., $k = 4$) and the ones with buddy heterogeneous clusters that have only 7 clusters.

6.2 Scheduling overhead for cluster scheduling with Bfair

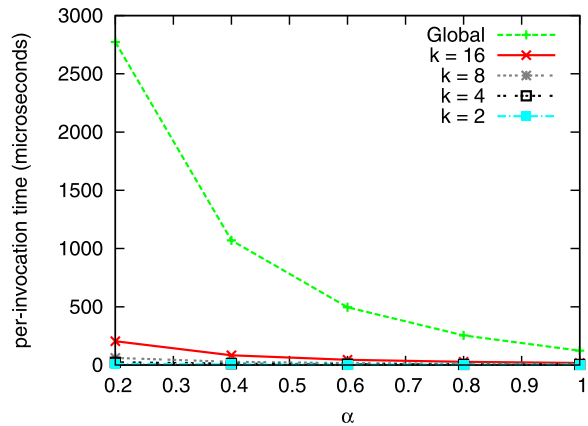
For systems with a given number of processors, although cluster scheduling can successfully schedule more task sets with higher system utilization for larger clusters each having more processors (i.e., smaller number of clusters), larger clusters could lead to higher scheduling overhead. When more tasks are allocated to each cluster, there are more scheduling points for the Bfair global scheduler and the interval between adjacent period boundaries can become smaller. Therefore, in addition to the time overhead for invoking the Bfair scheduler 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 continuous execution of tasks within each interval between period boundaries (Zhu et al. 2003, 2009).

In this section, for systems where the processors are organized as different size of homogeneous 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 $\frac{U_{tot}}{m} = 0.95$.

Fig. 6 Time overhead of the cluster-Bfair at each scheduling point within a cluster



(a) 16-processor systems



(b) 64-processor systems

Moreover, we set $P_{\min} = 10$ and $P_{\max} = 100$. The value of the maximum task utilization α is varied from 0.2 to 1.0. As mentioned earlier, for a given (normalized) system utilization, varying α effectively affects the number of tasks in the task sets under consideration, where smaller values of α lead to 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.

6.2.1 Scheduler time overhead at each scheduling point

For cluster scheduling with PA-FF, Figs. 6a and 6b first show the invocation time overhead of the Bfair scheduler at each scheduling point for 16-processor and 64-processor systems, respectively, when the systems are configured as different homogeneous clusters. Here, the Bfair algorithm is implemented in C and runs on a Linux machine with an Intel 2.4 GHz 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 scheduling (Zhu et al. 2003), which is labeled as “Global” and used for comparison.

As shown in Zhu et al. (2003, 2009), the time overhead of Bfair at each scheduling point depends largely on the number of tasks to be scheduled together. For the cluster scheduling under consideration, after tasks are allocated/partitioned to clusters, the scheduling decisions of Bfair for each cluster can be made independently. Therefore, as we can see from Fig. 6a, when the cluster size becomes smaller (i.e., smaller values of k), the time overhead at a scheduling point within a cluster can be significantly reduced. This is because 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 as shown in Fig. 6b, there are more tasks that can be scheduled and the time overhead at each scheduling point becomes larger, which is consistent with our previous results reported in (Zhu et al. 2003, 2009).

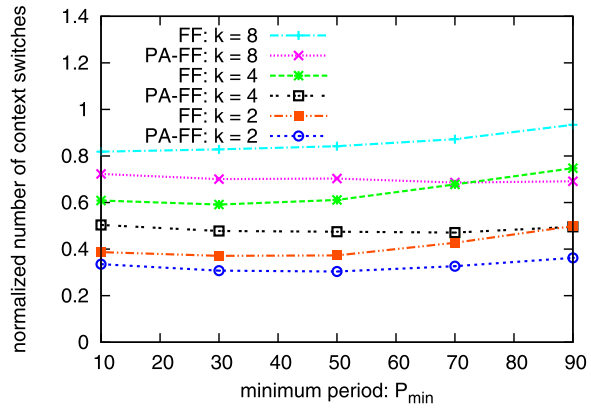
6.2.2 Number of context switches and task migrations

Next, we evaluate the schedules generated by cluster scheduling with Bfair for 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 PA-FF heuristic is evaluated against the simple first-fit (FF) allocation heuristic where tasks are randomly ordered. For easy comparison, the schedules generated by the global Bfair (i.e., with cluster size being $k = m$) are used as the baseline and normalized results are reported.

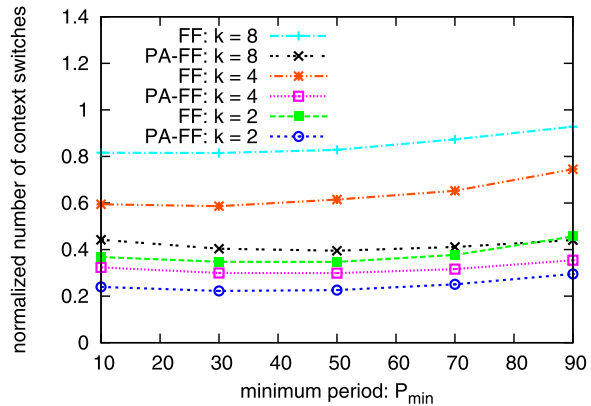
Figures 7a and 7b show the normalized number of context switches for the schedules generated by cluster scheduling with Bfair for systems with 16 and 64 processors, respectively. From the results, we can see that, compared to that of the schedules generated by the global Bfair where all tasks are scheduled together, the number of context switches in the schedules generated by cluster scheduling decreases drastically when 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 longer before the next scheduling point and thus to reduce the number of context switches. Moreover, by allocating harmonic tasks together, the proposed PA-FF allocation 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, when P_{max} is fixed as 100, more tasks are likely to have the same period as P_{min} increases. That is, the number of scheduling points becomes less for both global Bfair and cluster scheduling within each cluster. However, the reduction

Fig. 7 Normalized number of context switches



(a) 16-processor systems



(b) 64-processor systems

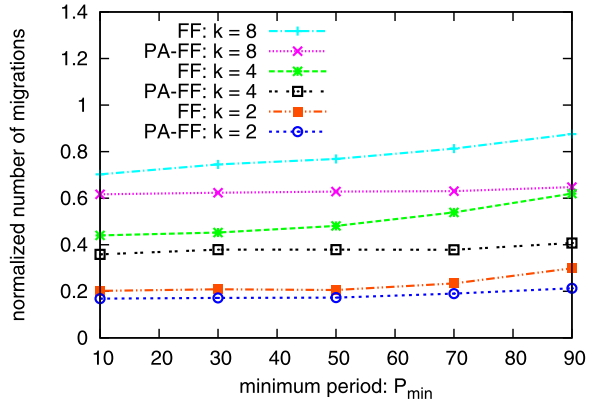
in scheduling points is more significant for global Bfair algorithm, which leads to a much reduced number of context switches (Zhu et al. 2003, 2009). Therefore, for the simple FF heuristic, the normalized number of context switches for the schedules generated by cluster scheduling increases slightly as P_{min} increases. However, for the proposed PA-FF allocation scheme that allocates tasks with harmonic periods together, the normalized number of context switches stays roughly the same.

Figures 8a and 8b 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 allocation scheme.

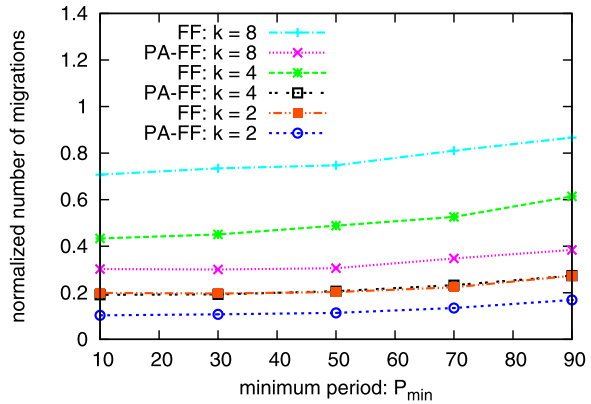
7 Conclusions

As a hierarchical approach to address the multiprocessor real-time scheduling problem, cluster scheduling has been studied recently, where processors are grouped into

Fig. 8 Normalized number of task migrations



(a) 16-processor systems



(b) 64-processor systems

clusters and tasks allocated to one cluster are scheduled by a global scheduler. In this paper, by adopting an optimal global scheduler within each cluster, we studied the worst-case utilization bound for cluster scheduling with different task allocation/partition heuristics. First, we obtained a lower limit on the utilization bounds for cluster scheduling with any reasonable task allocation schemes. Then, the lower limit is shown to be the utilization bound for cluster scheduling with the worst-fit task allocation scheme. For other task allocation heuristics (such as first-fit, best-fit, first-fit decreasing, best-fit decreasing and worst-fit-decreasing), higher utilization bounds were derived for systems with both *homogeneous* clusters (where each cluster has the same number of processors) and *heterogeneous* clusters (where clusters have different number of processors). In addition, focusing on an efficient optimal global scheduler, namely the *boundary-fair* (*Bfair*) algorithm, we propose a *period-aware* task allocation heuristic with the goal of reducing the scheduling overhead (e.g., the number of scheduling points, context switches and task migrations). The proposed schemes are evaluated through extensive simulations with synthetic real-time tasks and the simu-

lation 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., $k = 2$). In addition, 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). Moreover, when comparing to the simple generic task allocation scheme (e.g., first-fit), the proposed period-aware task allocation heuristic can further reduce the scheduling overhead of cluster scheduling with the Bfair scheduler.

Acknowledgements The authors acknowledge the support of US National Science Foundation (NSF) to this work through the awards CNS-0720651, CNS-0855247, CNS-1016855, CNS-1016974, and CAREER awards CNS-0546244 and CNS-0953005. Findings and opinions expressed in this work are those of the authors and not necessarily those of the funding agencies.

References

- Andersson B, Jonsson J (2003) The utilization bounds of partitioned and pfair static-priority scheduling on multiprocessors are 50%. In: Proc of the 15th Euromicro conference on real-time systems, pp 33–40
- Anderson J, Srinivasan A (2000a) Early-release fair scheduling. In: Proc of the 12th Euromicro conference on real-time systems, pp 35–43
- Anderson J, Srinivasan A (2000b) Pfair scheduling: Beyond periodic task systems. In: Proc of the 7th int'l workshop on real-time computing systems and applications, pp 297–306
- Anderson J, Srinivasan A (2001) Mixed pfair/erfair scheduling of asynchronous periodic tasks. In: Proc of the 13th Euromicro conference on real-time systems, pp 76–85
- Andersson B, Tovar E (2006) Multiprocessor scheduling with few preemptions. In: Proc of the IEEE int'l conference on embedded and real-time computing systems and applications, pp 322–334
- Andersson B, Baruah S, Jonsson J (2001) Static-priority scheduling on multiprocessors. In: Proc of the 22th IEEE real-time systems symposium, pp 193–202
- Andersson B, Bletsas K, Baruah S (2008) Scheduling arbitrary-deadline sporadic task systems on multiprocessors. In: Proc of the IEEE real-time systems symposium, pp 385–394
- Baker T (2003) Multiprocessor edf and deadline monotonic schedulability analysis. In: Proc. of the 24th IEEE real-time systems symposium, pp 120–129
- Baruah S (2007) Techniques for multiprocessor global schedulability analysis. In: Proc of the IEEE real-time systems symposium, pp 119–128
- Baruah S, Baker T (2008) Schedulability analysis of global edf. *Real-Time Syst* 38(3):223–235
- Baruah SK, Gehrke J, Plaxton CG (1995) Fast scheduling of periodic tasks on multiple resources. In: Proc of the international parallel processing symposium, pp 280–288
- Baruah SK, Cohen NK, Plaxton CG, Varel DA (1996) Proportionate progress: a notion of fairness in resource allocation. *Algorithmica* 15(6):600–625
- Bletsas K, Andersson B (2009) Preemption-light multiprocessor scheduling of sporadic tasks with high utilisation bound. In: Proc of the IEEE real-time systems symposium, pp 447–456
- Burchard A, Liebeherr J, Oh Y, Son SH (1996) New strategies for assigning real-time tasks to multiprocessor systems. *IEEE Trans Comput* 44(12):1429–1442
- Calandrino JM, Anderson JH, Baumberger DP (2007) A hybrid real-time scheduling approach for large-scale multicore platforms. In: Proc of the Euromicro conference on real-time systems, pp 247–258
- Cho H, Ravindran B, Jensen E (2006) An optimal real-time scheduling algorithm for multiprocessors. In: Proc of the IEEE real-time systems symposium, pp 101–110
- Cho H, Ravindran B, Jensen ED (2010) T-1 plane-based real-time scheduling for homogeneous multiprocessors. *J Parallel Distrib Comput* 70(3):225–236
- Darera VN (2006) Bounds for scheduling in non-identical uniform multiprocessor systems. Master's thesis, Indian Institute of Science, India
- Davis R, Burns A (2009a) Priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. In: Proc of the IEEE real-time systems symposium, pp 398–409

- Davis R, Burns A (2009b) A survey of hard real-time scheduling algorithms and schedulability analysis techniques for multiprocessor systems. Tech Rep YCS-2009-443, University of York, Department of Computer Science
- Dertouzos ML, Mok AK (1989) Multiprocessor on-line scheduling of hard-real-time tasks. *IEEE Trans Softw Eng* 15(12):1497–1505
- Devi U, Anderson J (2005) Tardiness bounds under global edf scheduling on a multiprocessor. In: Proc of the 26th IEEE real-time systems symposium, pp 330–341
- Dhall SK, Liu CL (1978) On a real-time scheduling problem. *Oper Res* 26(1):127–140
- Funaoka K, Kato S, Yamasaki N (2008) Work-conserving optimal real-time scheduling on multiprocessors. In: Proc of the Euromicro conference on real-time systems, pp 13–22
- Goossens J, Funk S, Baruah S (2003) Priority-driven scheduling of periodic task systems on multiprocessors. *Real-Time Syst* 25(2-3):187–205
- Guan N, Stigge M, Yi W, Yu G (2009) New response time bounds for fixed priority multiprocessor scheduling. In: Proc of the IEEE real-time systems symposium, pp 387–397
- Guan N, Stigge M, Yi W, Yu G (2010) Fixed-priority multiprocessor scheduling with Liu & Layland's utilization bound. In: Proc of the 16th IEEE real-time and embedded technology and applications symposium, pp 165–174
- Herbert S, Marculescu D (2007) Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In: Proc of Int'l symposium on low power electronics and design, pp 38–43
- Holman P, Anderson J (2001) Guaranteeing pfair supertasks by reweighting. In: Proc of the 22nd IEEE real-time systems symposium, pp 203–212
- Kato S, Yamasaki N (2007) Real-time scheduling with task splitting on multiprocessors. In: Proc of the IEEE int'l conference on embedded and real-time computing systems and applications, pp 441–450
- Kato S, Yamasaki N (2008) Portioned edf-based scheduling on multiprocessors. In: Proc of the 8th ACM int'l conference on embedded software, pp 139–148
- Kato S, Funaoka K, Yamasaki N (2009) Semi-partitioned scheduling of sporadic task systems on multiprocessors. In: Proc of the 21th Euromicro conference on real-time systems, pp 249–258
- Lauzac S, Melhem R, Mossé D (2003) An improved rate-monotonic admission control and its applications. *IEEE Trans Comput* 52:337–350
- Levin G, Funk S, Sadowski C, Pye I, Brandt S (2010) Dp-fair: a simple model for understanding optimal multiprocessor scheduling. In: Proc of the Euromicro conference on real-time systems, pp 3–13
- Liu JW (2000) Real-time systems. Prentice Hall/PTR, New York
- Liu CL, Layland JW (1973) Scheduling algorithms for multiprogramming in a hard real-time environment. *J ACM* 20(1):46–61
- Liu D, Lee YH (2004) Pfair scheduling of periodic tasks with allocation constraints on multiple processors. In: Proc of the 18th int'l parallel and distributed processing symposium, pp 119–126
- Lopez J, Garcia M, Diaz J, Garcia D (2000) Worst-case utilization bound for edf scheduling on real-time multiprocessor systems. In: Proc of the 12th Euromicro conference on real-time systems, pp 25–33
- Lopez J, Diaz J, Garcia D (2001) Minimum and maximum utilization bounds for multiprocessor rm scheduling. In: Proc. of the 13th Euromicro conference on real-time systems, pp 67–75
- Lopez J, Diaz J, Garcia D (2004) Utilization bound for edf scheduling on real-time multiprocessor systems. *Real-Time Syst* 28(1):39–68
- Moir M, Ramamurthy S (1999) Pfair scheduling of fixed and migrating tasks on multiple resources. In: Proc. of the 20th IEEE real-time systems symposium, pp 294–303
- Oh D, Baker T (1998) Utilization bound for n-processor rate monotone scheduling with stable processor assignment. *Real-Time Syst* 15(2):183–193
- Qi X, Zhu D (2008) Power management for real-time embedded systems on block-partitioned multicore platforms. In: Proc of the int'l conference on embedded software and systems, pp 110–117
- Shin I, Easwaran A, Lee I (2008) Hierarchical scheduling framework for virtual clustering of multiprocessors. In: Proc of the 20th Euromicro conference on real-time systems, pp 181–190
- Zhu D, Mossé D, Melhem R (2003) Periodic multiple resource scheduling problem: how much fairness is necessary. In: Proc of the 24th IEEE real-time systems symposium, pp 142–151
- Zhu D, Qi X, Mossé D, Melhem R (2009) An optimal boundary-fair scheduling algorithm for multiprocessor real-time systems. Tech Rep CS-TR-2009-005, Dept of CS, Univ of Texas at San Antonio



Xuan Qi received the B.S. degree in computer science from Beijing University of Posts and Telecommunications in 2005. He is now a Ph.D. candidate in Computer Science Department, University of Texas at San Antonio. His research interests include real-time systems, parallel systems, and high performance computing. His current research focuses on energy-efficient scheduling algorithms for multi-processor/multi-core real-time systems with reliability requirements.



Dakai Zhu received the BE in Computer Science and Engineering from Xi'an Jiaotong University in 1996, the ME degree in Computer Science and Technology from Tsinghua University in 1999, the MS and PhD degrees in Computer Science from University of Pittsburgh in 2001 and 2004, respectively. He joined the University of Texas at San Antonio as an assistant professor in 2005. His research interests include real-time systems, power aware computing and fault-tolerant systems. He has served on program committees (PCs) for several major IEEE and ACM-sponsored real-time conferences (e.g., RTAS and RTSS). He was a recipient of the US National Science Foundation (NSF) Faculty Early Career Development (CAREER) Award in 2010. He is a member of the IEEE and the IEEE Computer Society.



Hakan Aydin received the B.S. and M.S. degrees in Control and Computer Engineering from Istanbul Technical University in 1991 and 1994, respectively, and the Ph.D. degree in computer science from the University of Pittsburgh in 2001. He is currently an Associate Professor in the Computer Science Department at George Mason University, Fairfax, Virginia. He has served on the program committees of several conferences and workshops, including the IEEE Real-Time Systems Symposium and IEEE Real-time Technology and Applications Symposium. In addition, he served as the Technical Program Committee Chair of IEEE Real-time and Embedded Technology and Applications Symposium (RTAS'11). He was a recipient of the US National Science Foundation (NSF) Faculty Early Career Development (CAREER) Award in 2006. His research interests include real-time systems, low-power computing, and fault tolerance.