Harmonic-Fit Partitioned Scheduling For Fixed-Priority Real-Time Tasks On the Multiprocessor Platform*

Ming Fan Gang Quan
Electrical and Computer Engineering Department
Florida International University
Miami, FL, 33174
{mfan001, gaquan}@fiu.edu

Abstract

One common approach for partitioned multiprocessor scheduling problem is to transform this problem into a traditional bin-packing problem, with the utilization of a task being the "size" of the object and the utilization bound of a processor being the "capacity" of the bin. However, this approach ignores the fact that some implicit relations among tasks may significantly affect the feasibility of the tasks allocated to a processor. In this paper, we present a novel multiprocessor partitioned scheduling algorithm for fixed-priority sporadic real-time tasks based on the Rate Monotonic Scheduling (RMS) policy. Our approach takes advantage of the fact that harmonic tasks can achieve a much higher processor utilization than that defined by a utilization bound. As demonstrated in our experiment results, when taking the task period relationship into consideration, our algorithm can achieve a significant improvement over previous work.

I. Introduction

As semiconductor technology continues to scale down, the power consumption and heat generation issues significantly affect the performance and reliability of the computing systems [1], [2]. Thus, the development of computing systems, particularly the embedded real-time systems, relies more and more on multiprocessor platforms [3], [4]. A major issue in the software development for multiprocessor architecture is how to utilize the available computing resources most effectively. To this end, we study the problem of scheduling real-time tasks on multiprocessor architecture based on the *Rate Monotonic Scheduling* (RMS) policy.

It is a well-known fact that scheduling real-time tasks on multiprocessor platform is NP-hard [5]. Different from uniprocessor scheduling, the multiprocessor scheduling need to decide not only when but also where to execute a real-time task. The optimal uniprocessor scheduling algorithms, such as *Rate Monotonic Scheduling (RMS)* and *Earliest Deadline First (EDF)*, become no longer optimal any more [6] on multiprocessor system.

The general multiprocessor scheduling algorithms can be categorized into two classes [6], [7]: *global* scheduling and *partitioned* scheduling. In the global scheduling, all tasks are stored in a global queue, and each task can be scheduled into any available processor when selected from the queue. In the partitioned scheduling, each task is assigned to a particular processor, and will be executed on that processor without migration. Both approaches have their own pros and cons, and none of them dominates the other in terms of schedulability [7].

One common approach to solve the partitioned multiprocessor scheduling problem is based on the techniques for the *bin-packing* problem [5]. Coffman *et al*. [8]present a survey of the standard bin-packing techniques. The most commonly used bin-packing approaches for multiprocessor scheduling include first-fit approach, best-fit approach and worst-fit approach. The first-fit approach allocates each task to the first processor that can accommodate that task. The best-fit approach assigns a task to the processor with the largest total utilization that still can accommodate that task. The worst-fit approach always picks up the processor with the smallest total utilization when allocating each task. Abdelzaher *et al*. [9] develop a period-based load partitioning heuristic, which minimizes the average worst-case response time among all tasks. Naaman *et al*. propose a bin-packing based algorithm to deal with packets on TDMA network [10].

However, the bin-packing techniques consider only the utilization factor of a task and the utilization bound on a single processor, and totally ignore other parameters, such

* This work is supported in part by NSF under projects CNS-0969013, CNS- 0917021 and CNS-1018108.

978-0-7695-4552-3/11 $26.00 © 2011 IEEE
DOI 10.1109/EUC.2011.41

as the period (minimum inter arrival time) and execution requirement. In fact, certain specific parameters (i.e. period or execution time) may affect the total performance in terms of system utilization. For example, Kuo *et al.* [11] present an efficient scheduling algorithm for fixed-priority periodic tasks on single processor, which increases the system utilization by combining the tasks with harmonic relation of periods to reduce the effective task number. Han *et al.* [12] propose a feasibility test approach on single processor for RMS, in which the feasibility of a given task set can be determined by its corresponding harmonic counterpart. Conceivably, scheduling performance can be improved if these factors can be integrated into the multi-processor scheduling decisions.

In this paper, we present a new partitioned scheduling algorithm to schedule fix-priority sporadic tasks on multi-processor platform under RMS policy. We exploit the fact that harmonic tasks or tasks close to harmonic can utilize the processor more efficiently. We also conduct extensive experiments to study the performance of our approach, and our experimental results show that the proposed algorithm can significantly improve the scheduling performance compared with the previous work.

The rest of the paper is organized as follows. Section II describes the task model and other background information necessary for this paper. Section III presents our new partitioned scheduling algorithm. Experiments and results are discussed in Section IV, and we present the conclusions in Section V.

II. Preliminary

In this section, we first introduce the system model used in this paper, and then introduce some pertinent background information and concepts necessarily for this research. We then use an example to motivate our research.

A. System models

We first introduce the multiprocessor platform and task model used in this paper. The multiprocessor platform consists of M identical processors, $M \geq 2$, denoted as $\mathcal{P} = \{P_1, P_2, ..., P_M\}$. The task model considered in this paper consists of N sporadic tasks, denoted as $\Gamma = \{\tau_1, \tau_2, ..., \tau_N\}$. Each task τ_i, where $1 \leq i \leq N$, is characterized by a tuple (C_i, T_i). C_i is the *worst case execution time* of τ_i, and T_i is the *minimum inter-arrival time* between any two consecutive jobs of τ_i. For the sake of simplicity, we also refer to T_i as the *period* of τ_i. In this paper, we assume that Γ is sorted with non-decreasing period order, i.e. for any two tasks $\tau_i, \tau_j \in \Gamma$, $T_i \leq T_j$ if $i < j$. We also use Γ_k to denote the task set on processor P_k.

TABLE I. A task set with six real-time tasks

τ_i	C_i	T_i	u_i
1	1	4	0.25
2	2	8	0.25
3	3	10	0.30
4	8	16	0.50
5	8	20	0.40
6	12	40	0.30

To ease our presentation, we formally define several concepts as follows.

The *task utilization* of τ_i is denoted as u_i where

$$u_i = \frac{C_i}{T_i} \tag{1}$$

The *task set utilization* of Γ is denoted as $U(\Gamma)$ where

$$U(\Gamma) = \sum_{\tau_i \in \Gamma} u_i \tag{2}$$

Moreover, let $U(\Gamma_k)$ represent the total utilization of all tasks assigned to P_k.

The *system utilization* of a multiprocessor platform consisting of a task set Γ and M identical processors is denoted as $U_M(\Gamma)$, where

$$U_M(\Gamma) = \frac{U(\Gamma)}{M} \tag{3}$$

Liu and Layland [13] showed that a task set Γ can be feasibly scheduled by RMS on a single processor as long as

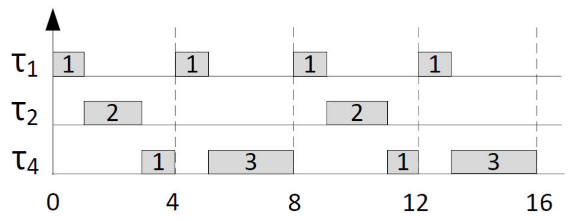
$$U(\Gamma) \leq \Theta(N) = N(2^{1/N} - 1). \tag{4}$$

$\Theta(N)$ is also traditionally referred to as the *Liu&Layland bound*. For a multiprocessor platform with M processors, the best known utilization bound for either global or partitioned fixed-priority schedule is no more than 50% [14], [15], [16].

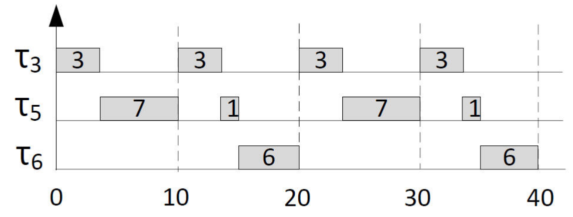
B. Motivation example

Before presenting our approach in detail, we first use an example to motivate our research.

Consider a multiprocessor platform with two processors, i.e. $M = 2$, and a task set consisting of six tasks with parameters shown in Table I. When scheduling those six tasks on two processors, traditional multiple scheduling techniques transform this problem to the *bin-packing problems* [5]. The utilization bound on each processor is usually used as the "capacity" of the bin, and the utilization factor of each task is treated as the "size" of the object. Unfortunately, it is not difficult to verify that none of the existing bin-packing heuristics (e.g. "first-fit", "best-fit" and "worst-fit") can successfully schedule the tasks listed in Table I.



(a) Processor 1



(b) Processor 2

Fig. 1. Assign tasks based on their harmonic relationship and tasks in Table I can be scheduled successfully on two processors.

Note that, current bin-packing based approaches allocate real-time tasks solely based on their utilization factors and simply ignore other factors such as the task period, which can significantly affect the schedulability of a real-time task. For example, it is a well known fact [11], [12] that a harmonic task set, i.e. the tasks with periods being integer multiples of each other, can have a much higher schedulability than other non-harmonic task sets. If we take this factor into consideration and assign τ_1, τ_2 and τ_4 to one processor, and τ_3, τ_5 and τ_6 to another processor, as shown in Figure 1(a), the task set in Table I can be perfectly scheduled on two processors.

Since tasks with harmonic relationship have much higher feasibility on a single processor, the direct and intuitive idea for partitioned multiprocessor scheduling would therefore be the one to group harmonic tasks together and assign them to one processor. However, there are few problems needed to be solved. First, since not all tasks are exactly harmonic in a task set, how to determine if they are close to harmonic. Second, how to incorporate the harmonic information into the allocation decision. In what follows, we present a novel partitioned scheduling algorithm, i.e. *Harmonic-Fit Partitioned Scheduling*, that can take the advantage of harmonic relationship among tasks.

III. New algorithm

The *Harmonic-Fit Partitioned Scheduling* (HFPS) algorithm is a harmonic-aware multiprocessor partitioned scheduling algorithm developed for fixed-priority sporadic tasks. When employing the harmonic relationship to improve the scheduling performance, it is not necessary that all tasks in the task set are strictly harmonic.

To this end, we first establish a metric, namely the *harmonic ratio*, to quantify the degree of harmonic for a task set. Then, we present our new partitioned scheduling algorithm based on this harmonic metric, and study the feasibility of the new algorithm.

A. Quantifying the harmonic

Since not all tasks in a given task set are harmonic, it is desirable that we can quantify the harmonic of a task set. Conceivably, the higher the harmonic of a task, the higher the system utilization can be. To achieve this goal, we first introduce the following concept.

Definition 1: Given a task set $\Gamma = \{\tau_1, \tau_2, ..., \tau_N\}$ where $\tau_i = (C_i, T_i)$, let $\Gamma' = \{\tau'_1, \tau'_2, ..., \tau'_N\}$ where $\tau'_i = (C_i, T'_i)$, $T'_i \le T_i$, and $T'_i | T'_j$ if $i < j$. (Note $a|b$ means "a divides b" or "b is an integer multiple of a".) Then Γ' is called a *sub harmonic task set* of Γ.
For a real-time task set and its sub harmonic task sets, it is not difficult to prove the following theorem [12].

Theorem 1: [12] Let Γ' be a sub harmonic task set of Γ. Then Γ is feasible on single processor under RMS if $U(\Gamma') \le 1$.

Moreover, for a given task set, there may be infinite numbers of different sub harmonic task sets. There is one type of sub harmonic task sets that is of most interest to us, which we call the *primary harmonic task set* and is formally defined as follows.

Definition 2: Let $\Gamma' = \{\tau'_1, \tau'_2, ..., \tau'_n\}$ be a sub harmonic task set of Γ. Then Γ' is called a *primary harmonic task set* of Γ if there exists no other sub harmonic task set (i.e. $\Gamma'' = \{\tau''_1, \tau''_2, ..., \tau''_n\}$) such that $T'_i \le T''_i$ for all $1 \le i \le n$.

One approach to identify primary harmonic task sets for a given task set is to employ the *DCT* algorithm [12]. For example, with respect to τ_i, the DCT approach can be briefly described as below.

$$\begin{aligned} C'_j &= C_j \\ T'_j &= \begin{cases} T_j, & \text{if } j = i, \\ T'_{j-1} \cdot \lfloor T_j / T'_{j-1} \rfloor, & \text{if } j > i, \\ \frac{T'_{j+1}}{\lceil T'_{j+1}/T_j \rceil}, & \text{if } j < i. \end{cases} \end{aligned} \quad (5)$$

We are now ready to define a metric, i.e. the *harmonic ratio*, to measure the harmonicity of a task set with its counterpart after harmonic transformation in terms of period.

Definition 3: Given a task set Γ, let Γ^P represent the set of all prime harmonic task sets of Γ. Then the *harmonic* *ratio* of Γ, denoted as $H(\Gamma)$, is defined as

$$H(\Gamma) = \min_{\Gamma' \in \Gamma^P} \sum_{\tau_i \in \Gamma} (1 - \frac{\Delta T_i}{T_i}) \tag{6}$$

where $\Delta T_i = T_i - T_i'$.

Intuitively, the harmonic ratio represents the relative *distance* of a task set to its prime harmonic task sets. In what follows, we introduce how we develop the our proposed Harmonic-Fit approach based on the above harmonic ratio.

B. Harmonic-Fit Partitioned Scheduling

In this subsection, we introduce a new multiprocessor partitioned scheduling algorithm, *Harmonic-Fit Partitioned Scheduling* (HFPS), to schedule fixed-priority sporadic tasks under RMS policy. The most significant differences between HFPS and the traditional bin-packing based approaches, (i.e. first-fit, best-fit and worst-fit), are that: 1) we take the period relation into consideration when allocating tasks. 2) instead of allocating tasks one by one, we allocate tasks group by group. To take advantage of the harmonic relationship among tasks, it is desirable to allocate tasks with high harmonic ratio to the same processor.

The basic idea of HFPS can be briefly described as below:

- For each task τ_i, construct a sub harmonic task set Γ' based on Equation (5).
- Pick up K_i tasks, denoted as Γ_{K_i}, from high harmonic ratio to low harmonic ratio by maximizing $U(\Gamma_{K_i})$ while keeping $U(\Gamma'_{K_i}) \leq 1$.
- Find the Γ_{opt} such that $U(\Gamma_{opt}) = \max_{i=1}^{N} \Gamma_{K_i}$.
- Allocate Γ_{opt} to an empty processor.

The HFPS is described in more details in Algorithm 1. Γ contains all unassigned tasks, and $\mathcal{P}$ contains all available processors. We assume that Γ is sorted with non-decreasing period order. When both Γ and $\mathcal{P}$ are not empty, we pick up a group of tasks with optimal combination, in terms of total utilization, and allocate them to one empty processor(from line 1 to line 17). In each iteration of the "while" loop, we first get the number of unassigned tasks from Γ (line 2). Let U_{opt} denote the utilization of the optimal task group picked up in this iteration and initialized to minus infinity in line 3. The "for" loop (from line 4 to line 13) contains three steps: 1) transforming each task in Γ based on Equation (5); 2) picking up K_i tasks with the higher harmonic relation according to the harmonic ratio under the three constrains, which can guarantee the feasibility of those K_i tasks while maximizing the processor utilization; 3) among all n harmonic transformations, choosing the group that has the maximum utilization in order to optimize the total system utilization. After finding the optimal group of tasks by the "for" loop, assign the corresponding tasks to the same processor (line 14). Consequently, update the unassigned task set by removing the optimal task group from Γ (line 15), and update the available processors by removing the occupied one for $\mathcal{P}$ (line 16). The algorithm succeeds if all tasks could be allocated, otherwise, it fails (from line 18 to line 22). In what follows, we conduct further feasibility analysis for this algorithm.

Algorithm 1 Harmonic-Fit Partitioned Scheduling (HFPS)

```
Require:
    1) Task set :$\Gamma = \{\tau_1, \tau_2, ...\tau_N\}$;
    2) Multiprocessor : $\mathcal{P} = \{P_1, P_2, ..., P_M\}$;
1:  while $\Gamma \neq \emptyset$ and $\mathcal{P} \neq \emptyset$ do
2:    $n = |\Gamma|$;
3:    $U_{opt} = -\infty$;
4:    for $i = 1$ to $n$ do
5:      $T_i' = T_i$
6:      for $j = i+1$ to $n$ do $T_j' = T_{j-1}' \cdot \lfloor T_j / T_{j-1}' \rfloor$;
7:      for $j = i-1$ downto 1 do $T_j' = \frac{T_{j+1}'}{\lceil T_{j+1}'/T_j \rceil}$;
8:      $\Gamma_{K_i}$ = pick up $K_i$ tasks from $\Gamma$ such that
          (1) $U(\Gamma'_{K_i}) \leq 1$ and
          (2) $H(\Gamma_{K_i})$ is maximized
          (3) $U(\Gamma_{K_i})$ is maximized;
9:      if $U(\Gamma_{K_i}) > U_{opt}$ then
10:       $U_{opt} = U(\Gamma_{K_i})$;
11:       $\Gamma_{opt} = \Gamma_{K_i}$;
12:     end if
13:   end for
14:   pick up $P_k \in \mathcal{P}$, and assign $\Gamma_{opt}$ to $P_k$;
15:   $\Gamma = \Gamma \setminus \Gamma_{opt}$;
16:   $\mathcal{P} = \mathcal{P} \setminus P_k$;
17: end while
18: if $\Gamma = \emptyset$ then
19:   return "success";
20: else
21:   return "failure";
22: end if
```

After successfully partitioning all tasks by HFPS, the RMS policy is used on each processor. Theorem 2 formally guarantees that a real-time task is feasible if it is successfully allocated with HFPS.

Theorem 2: If a task set Γ is successfully partitioned by HFPS on M processors and scheduled according to RMS, then all tasks can meet their deadlines.

Proof sketch: Consider any arbitrary processor P_k and the corresponding task set Γ_k. According to HFPS in Algorithm 1, we know that if HFPS finishes successfully, then all tasks in Γ_k are partitioned by group. Moreover, there must exist a harmonic task set of Γ_k, denote as Γ_k', such that

$$U(\Gamma_k') \leq 1 \tag{7}$$

According to Theorem 1, we know that Γ_k is feasible on single processor under RMS. Thus, we see the task set of any processor is feasible. Therefore, all tasks can meet their deadline. □

IV. Experiments and results

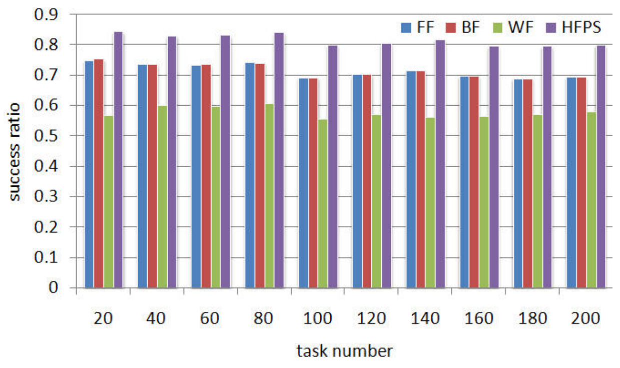
In this section, we investigate the performance of our proposed algorithm HFPS with experiments. We compare our approach with three well-known bin-packing based algorithms: first-fit (FF), best-fit (BF) and worst-fit (WF). The simulation models and results are represented below.

We conducted two sets of experiment. In the first one, we varied the task number from 20 to 200 with an increment of 20. In the second one, we varied the system utilization within $[0.5, 1.00]$. For both experiments, test cases were randomly generated based on the number of processors, i.e. $m = 4, 8$, and 16. And for each test point, we generated 1,000 task sets. The minimum inter-arrival time of each task (T_i) was uniformly distributed within $[50, 500]$. The utilization of each task (u_i) was randomly generated with uniform distribution within $[0.02, 1.00]$. Then the execution time (C_i) was calculated by $C_i = T_i \cdot u_i$. We only chose task sets with system utilization within $[0.5, 1.00]$ since task sets with smaller utilizations could be easily schedulable. The *success ratios* by different approaches, i.e. the ratio between the number of feasible tasks and the total number of tasks, under a particular setting were recorded correspondingly. Final results of both sets of experiment were plotted in Figure 2 and Figure 3, respectively.

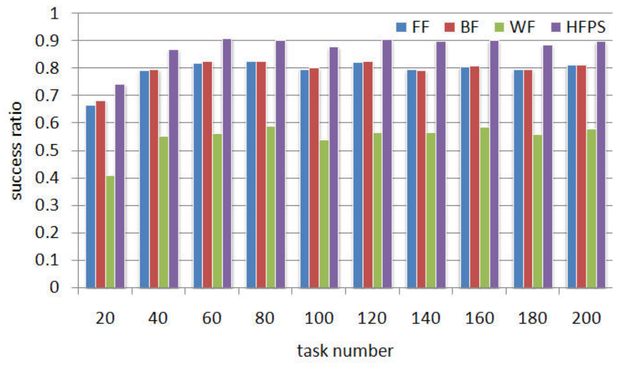
Figure 2(a), 2(b) and 2(c) show the simulation results for 4, 8 and 16 processors, respectively. From these figures, we can readily observe that the proposed algorithm HFPS outperforms others.

First, we can see that as the number of tasks increases, HFPS achieves higher success ratio than other approaches. For example, in Figure 2(a), when task number is 120, the success ratio of HFPS is 80%, while that ratios of FF, BF and WF are no larger than 70%. This is because the more of tasks, the higher probability of HFPS to increase the system utilization by grouping high harmonic task together. Second, we can also see that as the number of processors increases, the success ratio for the same task number increases. Note that, in Figure 2(b), HFPS achieves 90% success ratio when task number equal to 120, which is 10% greater than that in Figure 2(a). This is because the more processors we have, the more harmonic groups we can find. This provides us more chance to partition the strong harmonic tasks into the same group and thus utilize the processor efficiently. Therefore, we see that in the experiment based on task number, HFPS outperforms the previous work.

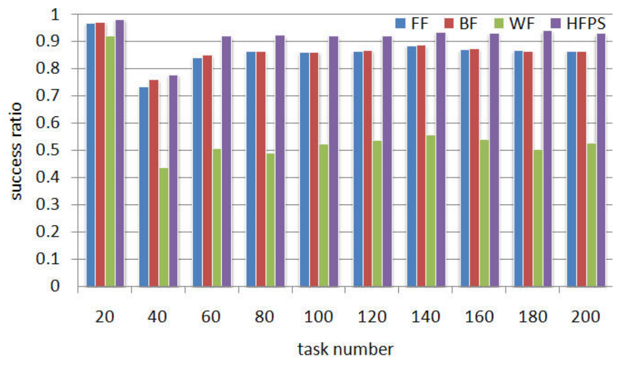
From the experiment results of Figure 3, we can see that



(a) Processor number m = 4

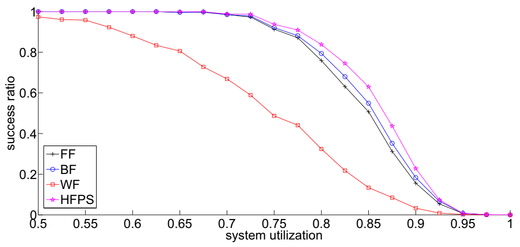


(b) Processor number m = 8

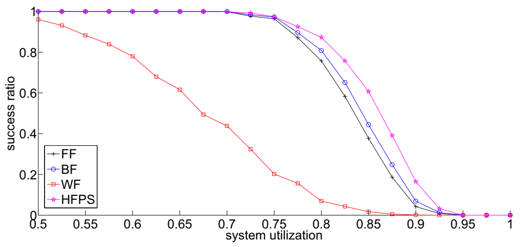


(c) Processor m = 16

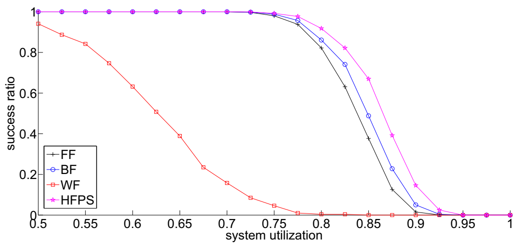
Fig. 2. Experiment results for general task sets



(a) Processor number m = 4



(b) Processor number m = 8



(c) Processor m = 16

Fig. 3. Experiment results for general task sets

the success ratio of HFPS also outperforms the other three approaches for task sets with different task utilizations. The larger the task utilization is, the more improvement that our approach can achieve. For example, in Figure 3(b), when system utilization is 0.85, the success ratio of HFPS is up to 60%, which is at least 20% more than any of other three approaches. Thus, we see by taking the harmonic relation into account during partitioned scheduling, the new algorithm can obtain a better outcome than the traditional bin-packing based approaches.

V. Conclusions

In this paper, we present a new multiprocessor partitioned scheduling algorithm, HFPS algorithm, for fixed-priority sporadic task systems. Our approach can take the advantage of the harmonic relations among the tasks and improve the feasibility. Particularly, HFPS allocates tasks group by group in order to find the optimal combination in terms of system utilization with respect of period. We also present the feasibility analysis of HFPS. Furthermore, our extensive experiment results demonstrate that the proposed algorithm can significantly improve the scheduling performance compared with the traditional work.

References

[1] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," *ICSA*, pp. 2–13, 2003.

[2] L.-T. Yeh and R. C. Chu, *Thermal Management of Microelectronic Equipment: Heat Transfer Theory, Analysis Methods, and Design Practices*. New York, NY: ASME Press, 2002.

[3] S. Chaudhry, R. Cypher, M. Ekman, M. Karlsson, A. Landin, S. Yip, H. Zeffer, and M. Tremblay, "Rock: A high-performance sparc cmt processor," *Micro, IEEE*, vol. 29, no. 2, pp. 6 –16, March-April 2009.

[4] W. Wolf, "Multiprocessor system-on-chip technology," *Signal Processing Magazine, IEEE*, vol. 26, no. 6, pp. 50 –54, November 2009.

[5] K. Shin and P. Ramanathan, "Real-time computing: a new discipline of computer science and engineering," *Proceedings of the IEEE*, vol. 82, no. 1, pp. 6 –24, Jan 1994.

[6] S. K. Dhall and C. L. Liu, "On a real-time scheduling problem," *Operations Research*, vol. 26, no. 1, pp. pp. 127–140, 1978. [Online]. Available: http://www.jstor.org/stable/169896

[7] J. Carpenter, S. Funk, P. Holman, A. Srinivasan, J. Anderson, and S. Baruah, "A categorization of real-time multiprocessor scheduling problems and algorithms," in *Handbook on Scheduling Algorithms, Methods, and Models*. Chapman Hall/CRC, Boca, 2004.

[8] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson, *Approximation algorithms for bin packing: a survey*. Boston, MA, USA: PWS Publishing Co., 1997, pp. 46–93. [Online]. Available: http://portal.acm.org/citation.cfm?id=241938.241940

[9] T. Abdelzaher and K. Shin, "Period-based load partitioning and assignment for large real-time applications," *Computers, IEEE Transactions on*, vol. 49, no. 1, pp. 81 –87, jan 2000.

[10] N. Naaman and R. Rom, "Packet scheduling with fragmentation," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, 2002, pp. 427 – 436 vol.1.

[11] T.-W. Kuo and A. Mok, "Load adjustment in adaptive real-time systems," in *Real-Time Systems Symposium, 1991. Proceedings., Twelfth*, Dec 1991, pp. 160 –170.

[12] C.-C. Han and H.-Y. Tyan, "A better polynomial-time schedulability test for real-time fixed-priority scheduling algorithms," in *Real-Time Systems Symposium, 1997. Proceedings., The 18th IEEE*, Dec 1997, pp. 36 –45.

[13] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, pp. 46–61, January 1973. [Online]. Available: http://doi.acm.org/10.1145/321738.321743

[14] B. Andersson, "Global static-priority preemptive multiprocessor scheduling with utilization bound 38%," in *Principles of Distributed Systems*, ser. Lecture Notes in Computer Science, T. Baker, A. Bui, and S. Tixeuil, Eds. Springer Berlin / Heidelberg, 2008, vol. 5401, pp. 73–88, 10.1007/978-3-540-92221-6_7. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-92221-6_7

[15] B. Andersson, S. Baruah, and J. Jonsson, "Static-priority scheduling on multiprocessors," in *Real-Time Systems Symposium, 2001. (RTSS 2001). Proceedings. 22nd IEEE*, Dec 2001, pp. 193 – 202.

[16] B. Andersson and J. Jonsson, "The utilization bounds of partitioned and pfair static-priority scheduling on multiprocessors are 50%," in *Real-Time Systems, 2003. Proceedings. 15th Euromicro Conference on*, July 2003, pp. 33 – 40.