

Hybrid Multiprocessor Real-Time Scheduling Approach

Ali A. Safaei[1], Mehdi Alemi[2], Mostafa S. Haghjoo[3], Shirin Mohammadi[4]

**[1] Department of Computer Engineering, Iran University of Science and Technology
Tehran, Iran**

**[2] Department of Computer Engineering, Iran University of Science and Technology
Tehran, Iran**

**[3] Department of Computer Engineering, Iran University of Science and Technology
Tehran, Iran**

**[4] Department of Computer Engineering, Iran University of Science and Technology
Tehran, Iran**

Abstract

Real-time scheduling is one of the most important aspects of a real-time system design. To achieve a real-time system's requirement, especially to be fast, multiprocessor systems are used. Generally, multiprocessor real-time scheduling algorithms fall into one the two well-known approaches: Partitioning or Global. The partitioning approach has acceptable overhead for underlying system but can NOT guarantee to provide an optimal schedule. The global approach can provide this guarantee by holding some preconditions and considerable overheads.

In this paper, an intermediate hybrid multiprocessor real-time scheduling approach is proposed in which optimality will be reached via the minimum overheads for underlying system. Presenting and analyzing different feasible paradigms for combination of the two existing approaches, the proposed hybrid approach satisfies the two major goals of this combination: optimality and lightweightness. Experimental results show that the hybrid approach outperforms the two existing ones.

Keywords: *real-time scheduling, multiprocessor systems, hybrid partitioning and global approaches.*

1. Introduction

A real-time system has two notions of correctness: logical and temporal. In particular, in addition to producing correct outputs (logical correctness), such a system needs to ensure that these outputs are produced at the correct time (temporal correctness). Selecting appropriate methods for scheduling activities is one of the important considerations in the design of a real-time system. A major obstacle is that scheduling algorithms are significantly more complex for multiprocessor systems than for uniprocessor systems since the scheduling algorithm must not only specify an execution ordering of tasks scheduling problem on each processor, but also determine the specific processor on which each task must execute assignment problem. Traditionally, there have been two approaches for scheduling real-time tasks on multiprocessor systems: Partitioning and Global scheduling approaches.

In partitioning, each task is assigned to a single processor, on which each of its jobs will execute, and processors are scheduled independently. In other words, each processor has its own task waiting queue. The set of tasks is partitioned and each task is assigned to the proper processor (task waiting queue) according to the allocation algorithm. Each processor executes tasks in its task waiting queue according to its real-time scheduling policy (figure 1.(a)). The main advantage of partitioning approaches is that they reduce a multiprocessor scheduling problem to a set of uniprocessor ones. Unfortunately, partitioning has two negative consequences. First, finding an optimal assignment of tasks to processors is a bin-packing problem, which is NP-hard in the strong sense. Thus, tasks are usually partitioned using non-optimal heuristics. Second, task systems exist that are schedulable if and only if tasks are not partitioned. Still, partitioning approaches are widely used by system designers.

In global scheduling, all eligible tasks are stored in a single priority-ordered queue; the global scheduler selects for execution the highest priority tasks from this queue. In other words, each task can be executed over all processors. In fact, a task which is started in a processor can migrate to any other processor to be continued (figure 1.(b)) [1]. Unfortunately, using this approach with optimal uniprocessor scheduling algorithms, such as the rate-monotonic (RM) and earliest-deadline-first (EDF) algorithms may result in arbitrarily low processor utilization in multiprocessor systems. However, recent research on proportionate fair (Pfair) scheduling has shown considerable promise in that it has produced the only known optimal method for scheduling periodic tasks on multiprocessors.

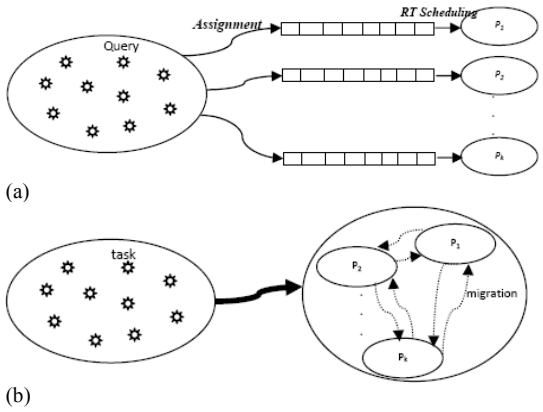

Fig. 1 Multiprocessor real-time scheduling approaches. (a) The partitioning approach (b) The global approach.

Generally, online real-time scheduling in multiprocessor systems is a NP-hard problem [2]. The partitioning approach may not be optimal but is suitable for real-time DSMS because: *(1)* Independent real-time scheduling policies can be employed for each task queue. Therefore, the multiprocessor real-time scheduling problem is simplified to single processor real-time scheduling. *(2)* It has low run-time overhead which helps for better performance [1]. The global approach has the ability to provide optimal scheduling due the migration capability, but has considerable overhead. Furthermore, to have the optimal schedule, some preconditions must be held which is not possible in all applications [1].

To handle these deficits of the two existing approaches, in this paper, an intermediate hybrid multiprocessor real-time scheduling approach is proposed which tries to benefits the advantages of the two approaches, partitioning and global.

This paper is continued as follows: we present related work in section 2. Different feasible paradigms for combination of the two well-known approaches are introduced and analyzed in section 3. Our proposed hybrid multiprocessor real-time scheduling approach is presented in section 4. Simulation and performance evaluation of the presented approach is presented in section 5. Finally, we conclude in section 6.

2. Related Work

In [3] eight misconceptions in real-time databases are discussed. One of the most a common and important misconception is: "*real-time computing is equivalent to fast computing.*" In fact, fast processing does NOT guarantee time constraints. In other words, although being fast is necessary but is not sufficient. For a real-time system, there is a need for other mechanisms (real-time scheduling, feedback control, etc) to handle and satisfy time constraints. To meet fast operation, many of real-time systems are multiprocessor systems.

On the other hand, the main contribution in a real-time system design is its real-time scheduling. History of important events and key results in real-time scheduling is reviewed in [4]. Multiprocessor real-time scheduling which is totally different from traditional single processor real-time scheduling is classified into two approaches: global and partitioning. Problems and algorithms related to these approaches are discussed in [2]. Despite optimality of PFair scheduling algorithms (such as PF [5], PD [6] and PD^2 [7]), partitioning is currently favored [8]. The reasons are: *(a)* PFair scheduling algorithms have excessive overhead due to frequent preemptions and migrations *(b)* PFair scheduling are limited to periodic tasks *(c)* though partitioning approaches are not theoretically optimal, they tend to perform well in practice [8]. Utilization bound of EDF scheduling policy with partitioning approach in multiprocessor system is increased in comparison with single processor systems. Utilization bound in these environments depends on the employed allocation algorithm and task size. Utilization bound of EDF for multiprocessor systems with extended and complex task model (e.g., resource sharing, jitter of task release, deadlines less than period, aperiodic and non-preemptive tasks) is studied in [9]. In order to schedule soft deadline tasks in multiprocessor systems efficiently, PFair scheduling algorithm (known as optimal for hard real-time applications) is extended in [55]. This extension (known as EPDF

PFair) considers tardiness bound and uses the global approach as well as PFair.

In [11] supertasking is proposed to improve processor utilization in multiprocessor real-time systems. In this scheme, a set of tasks, called component tasks, is assigned to a server task, called a supertask, which is then scheduled as an ordinary Pfair task. Whenever a supertask is scheduled, its processor time is allocated to its component tasks according to an internal scheduling algorithm.

3. Paradigms of Hybrid Multiprocessor Real-Time Scheduling Approach

In general, real-time scheduling algorithms in multiprocessor systems fall into one the two well-known approaches: *partitioning* and *global*.

Partitioning multiprocessor real-time scheduling approach is simpler and has an acceptable overhead for the underlying multiprocessor system. So, it is preferred to be employed in the real-world operational systems, but it can't guarantee to provide an optimal schedule.

The reason is that, due to its static task assignment and disability of tasks migration and sharing processors' capacity for executing a task, we can NOT use the whole of system capacity (utilization) in some cases and therefore, it may be suboptimal. Besides, the partitioning approach simplifies the multiprocessor real-time scheduling problem to the uniprocessor ones (*i.e.,* using the optimal uniprocessor real-time scheduling algorithms such as *EDF*[1] or *RM*[2] for each of processors' tasks waiting queues). Roughly speaking, partitioning approach is more profitable when tasks' set is static and predefined.

However, in global approach which is proper for dynamic task systems, we can expect for optimality of the scheduling via employing task migration. But, to reach this goal, some preconditions must be held which are not promising forever in the real-world operational systems. Moreover, the most important deficit of the global multiprocessor real-time scheduling approach is its considerable overhead for the underlying system which causes the global approach be out of use for applying in the real-world multiprocessor real-time systems [2].

In this paper, in order to satisfy operational systems requirements and make a trade-off between optimality of the scheduling and system overloads, an intermediate *hybrid* multiprocessor real-time scheduling approach is proposed which combines the two existing approaches. It aims to employs the benefits and advantages of the two existing multiprocessor real-time scheduling approaches, partitioning and global. The main goals of the proposed hybrid approach are as followings:

a. ***Optimality:*** *maximizing task system's scheduleability w.r.t. processors utilization capacity.*

b. ***Lightweightness:*** *minimizing scheduling overheads for underlying system.*

In order to propose new hybrid multiprocessor real-time scheduling approach to satisfy the desired goals, different paradigms which could be imagined via combing the two existing approaches are considered and analyzed first. These paradigms can be considered from the following perspectives.

3.2 Base of combination

By the two existing approaches, partitioning and global, we can use each one as the base of the combination while using the other one's capabilities to complete it and improve system performance. Accordingly, the following options are possible:

I. Partitioning-based

In this method of combination, base of the hybrid multiprocessor real-time scheduling approach is the partitioning approach and whenever needed, the global approach can be used to improve system utilization and to be near optimal (*e.g.,* via task migration).

II. Global-based

In this method, base of the hybrid multiprocessor real-time scheduling approach is the global approach and tasks can migrate among different processors. But in order to employ the partitioning approach (for overhead reduction) some of the tasks (as a batch of tasks) must be bound to a specific processor.

Since, binding tasks to a specific processor causes the ability of migration to be deprived from them, binding all of the tasks in the task system leads to no migration to happen in the system which contradicts the claim of employing the global approach.

So, increasing the number of binding tasks to processors (due to applying partitioning approach) causes decreasing the number of task migrations.

Roughly speaking, applying the partitioning approach conflicts with applying the global approach. So, the method of combination of approaches practically is contrary to its definition.

[1] Earliest Deadline First
[2] Rate Monotonic

In [11], this method is used by modifying the definition of the task. In other words, a new abstraction of task is proposed in which real tasks as component tasks of a supertask are bound to processors whilst the supertask (logical concept) is able to migrate between processors. So, we can say that supertasking[11] uses the global-based method of combination.

Also, from another point of view, precedence of employing each of the two well-known multiprocessor real-time scheduling approaches is important.

3.2 Sequence of combination

Based on the precedence (time) of applying an approach (base approach) rather than the other, combination methods to have a hybrid approach is classified as follows:

A. Serial employment

The two existing multiprocessor real-time scheduling approaches are employed sequentially and isolated (*i.e.,* the second one will be started whenever the first one completed) as shown in figure 2.

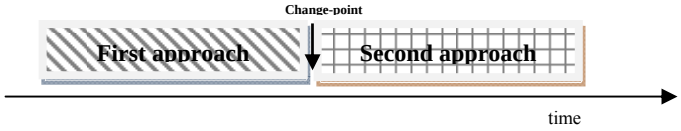

Fig. 2 Serial employment of multiprocessor real-time scheduling approaches.

B. Concurrent employment

The two existing multiprocessor real-time scheduling approaches are employed concurrently over the scheduling time, as shown in figure 3:

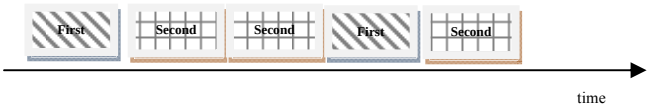

Fig. 3 An example of concurrent employment of multiprocessor real-time scheduling approaches.

Analogous to serial and concurrent scheduling of transactions, the serial method of employment is simpler and easier to implement and manage, while the concurrent method, which is more complicated, improves system performance.

There are many different cases of concurrent employment of the two well-known multiprocessor real-time scheduling approaches that discussing and analyzing them is out of the scope of this paper.

On the other hand, in the serial method, w.r.t. the existence of two approaches for multiprocessor real-time scheduling (partitioning and global), the two following cases are imaginable for serial employment method:

a) Partitioning-first

As shown in figure 4, in the partitioning-first method, first, tasks of the task system are assigned and bound to the processors via the partitioning approach until the situation in which no more task can be assigned to any of the processors, correspondingly.

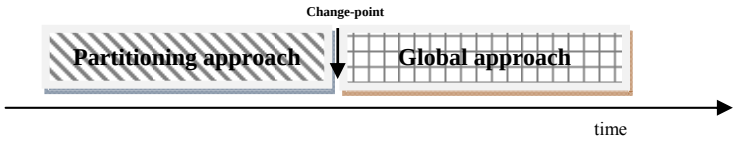

Fig. 4 change point in partitioning-first method.

Change-point condition: *all of the tasks (in the task system* $\tau = \{t_i \mid 1 \le i \le N\}$ *) before task k are assigned and bound to one of the processors and weight of task k is greater than remained utilization capacity of all of the processors*.

$$\forall 1 \le i \le k, \exists 1 \le j \le M; (P_j \leftarrow t_i) \wedge (w(t_k) > \max_j \{U_r(P_j)\}) \quad (1)$$

In which M is number of processors, $P_j \leftarrow t_i$ *means that tasks* t_i *is bound to processor* P_j*,* $w(t_k)$ *indicates weight of task k (i.e.,* e/p*) and* $U_r(P_j)$ *denotes remained utilization of processor* P_j*.*

In such a situation, the global approach will be used and the remained tasks can migrate among the processors to be completed.

Theorem 1: *the partitioning-first method of combination has the minimum system overhead.*

Proof: the partitioning approach has minimum overhead (w.r.t. the other choice, the global approach). Since the partitioning-first method uses partitioning approach as much as possible, therefore, there will be no other feasible option to have less overhead (i.e., no more chance to schedule the remained tasks via the partitioning approach).

Also, in the second phase of the partitioning-first method (i.e., employing the global approach), employing the EPDF PFair scheduling algorithm [10] which has the least overhead (among the algorithms that are based on the global approach), will have the minimum overhead in this phase.

b) Global-first

Since breaking down and migration of the tasks can be possible forever (while processors' utilization capacity is not completely full), this method is contrary with its definition; because we can use the first approach (i.e., global) forever whilst the second

one never is used. Albeit, the second approach can be used if we modify the definition of change-point (condition for starting the second approach). But, where is the proper choice for the change-point? The later the change-point, the more the system overhead (*i.e.,* more usage of the global approach).

Fig. 5 Late change point in global-frist method.

On the other hand, the early change-point, as shown in figure 6, does NOT necessarily provide the least overhead. The reason is that in this method, at first, tasks are scheduled using global approach (via migrations between processors) which has more overhead rather than the case that we could schedule them on the empty processors via the partitioning approach. Late change point and early change point are shown in figure 5 and 6, respectively.

Fig. 6 Early change point in global-first method.

This deficit of the global-first method raises this fact that in order to achieve the major goals of our hybrid approach, the partitioning-first method is strongly preferred.

4. Hybrid Multiprocessor Real-Time Scheduling Approach

As discussed in section3, different paradigms are imaginable for combining the two well-known multiprocessor real-time scheduling approaches (partitioning and global) in order to provide a hybrid approach which have the advantages of the two approaches. The major goals in the designing the hybrid multiprocessor real-time scheduling approach is to be optimal as well as have low overhead.

Among different paradigms that can be considered via combination of the two existing approaches, the portioning-based and partitioning-first combination method introduces the best method. The reason is that, this method has the most possible usage of the partitioning approach (which has the minimum overhead) and then uses the global approach to make use of processors empty capacity for execution of tasks in order to have optimal system utilization.

Accordingly, the proposed hybrid multiprocessor real-time scheduling approach is partitioning-based and partitioning-first approach in which the best algorithm of each of the partitioning and global approaches (*i.e.,* FF+EDF [12,2] and PD^2 [7]) are employed.

The proposed hybrid multiprocessor real-time scheduling approach consists of the following steps:

- All of the tasks in the task system τ are assigned and bound to the specific processors (*e.g.,* using the First-Fit algorithm [2]).
- Each processor schedules the tasks waiting in its waiting queue with the EDF policy [2].
- These will be continued until the change-point condition (equation (*1*)) holds.
- After this, remained tasks in the task system (or even the tasks that are entered the system newly (dynamic task system)) are considered as tasks that must be scheduled on set of processors which their utilization capacity is reduced (updated with their remained utilization capacity as equation (*2*)):

$$\forall 1 \le i \le N, 1 \le j \le M : U_r(P_j) = U(P_j) - \sum_i w(t_i) \quad (2)$$

In the the global approach (migration is allowed) the proper PFair scheduling algorithm (*e.g.,* PD^2) is employed.

5. Performance Evaluation

5.1 Experimental Setup

We implemented a real-time system prototype in Java in Linux [13] environment on a machine with Core i7 2930 processor and 6GB RAM. Each task is modeled with two characteristics as (e, p) in which *e* indicates its execution time and *p* its period. Partitioning approach in which First-Fit algorithm [2] is used as allocation algorithm and *EDF* is used as single processor real-time scheduling algorithm (named *Partitioning*) is compared with the global approach in which PD^2 PFair algorithm is employed (named *Global*) and the proposed hybrid approach which uses Partitioning-based and partitioning-first combination method (in which FF-EDF and PD^2 algorithms are employed respectively) (named *Hybrid*). The task system consists of 15 each with the

utilization (weight i.e., e/p) of 0.4. M (number of processors) is equal to 4. Simulation duration is 1e+10 seconds and average values for 12 different execution of this scenario is measured and compared. The most important evaluated parameters are:

- ***DMR***: deadline miss ratio according to equation (*3*), which is the most important parameter for a real-time system.

  DMR = (number of rejected tasks + number of missed tasks) / total number of tasks *(3)*

- ***Throughput***: number of tasks executed in a time unit.

Overheads such as communication or context-switching are negligible because the employed machines are cores of a multi-core CPU.

5.2 Experimental Results

Experimental results are shown in the following figures. As expected, the global approach has better performance compared to the partitioning approach; also, as shown in figures 7 and 8, the proposed hybrid multiprocessor real-time scheduling approach has a considerable improvement in terms of deadline miss ratio and system throughput rather than the existing approaches, partitioning and global. To have a comparison at a glance, the average value of the two parameters are computed and illustrated in figures 9 and 10.

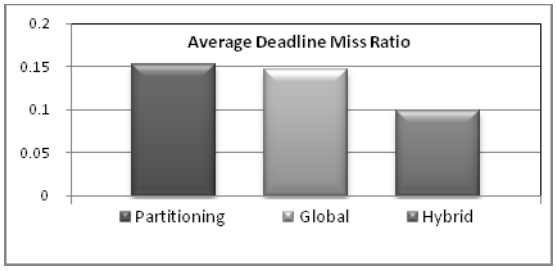

Fig. 9 Comparison of deadline miss ratio in average case.

Figures 7 and 8 illustrate comparison of deadline miss ratio and system throughput, respectively between the proposed hybrid multiprocessor real-time scheduling approach and the two existing approaches, partitioning and global.

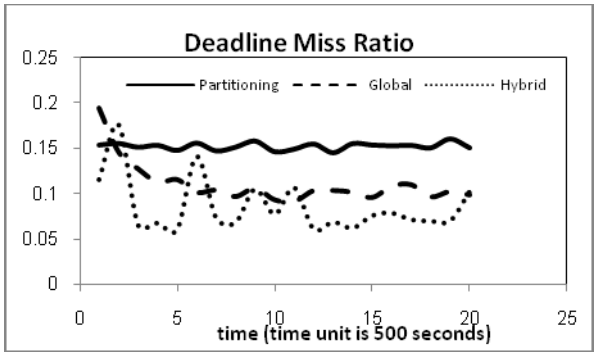

Fig. 7 Comparison of deadline miss ratio.

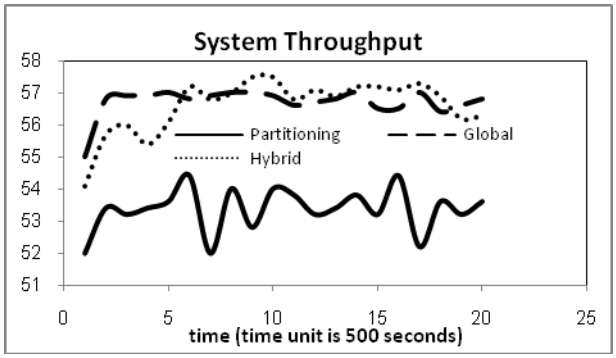

Fig. 8 Comparison of system throughput.

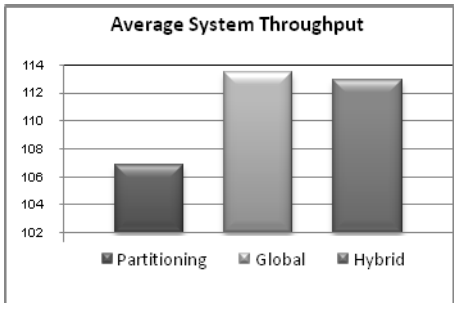

Fig. 10 Comparison of system throughput in average case.

Furthermore, to analyze the behavior of the proposed approach w.r.t. increment of number of the tasks, the values of deadline miss ratio and system throughput and their average values when number of tasks in task system is 100 are measured and shown in figures 11 through 14.

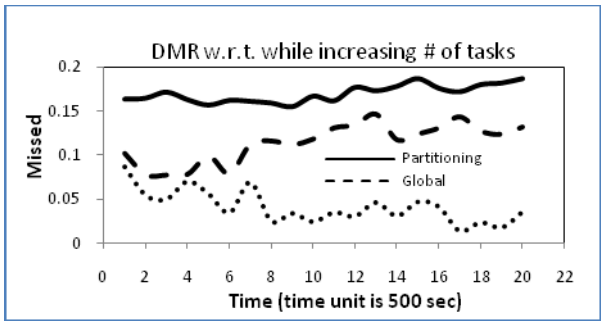

Fig. 11 Comparison of deadline miss ratio when number of tasks increases.

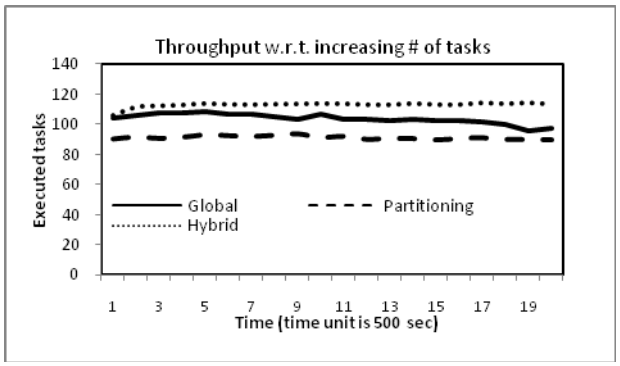

Fig. 12 Comparison of system throughput when number of tasks increases.

According to the results shown in figures 8 and 9, while number of tasks increases, the proposed hybrid multiprocessor real-time scheduling approach outperforms the partitioning and global approaches in terms of deadline miss ratio and system throughput. Also, the average values of deadline miss ratio and system throughput while increasing the number of tasks is shown in figures 13 and 14, respectively.

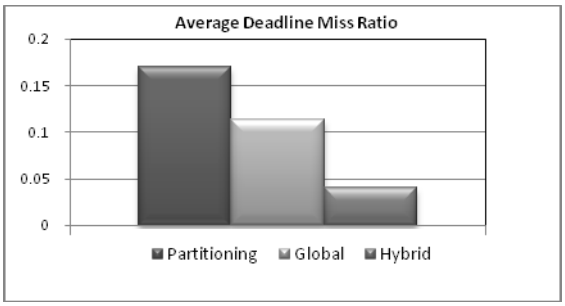

Fig. 13 Comparison of average DMR when number of tasks increases.

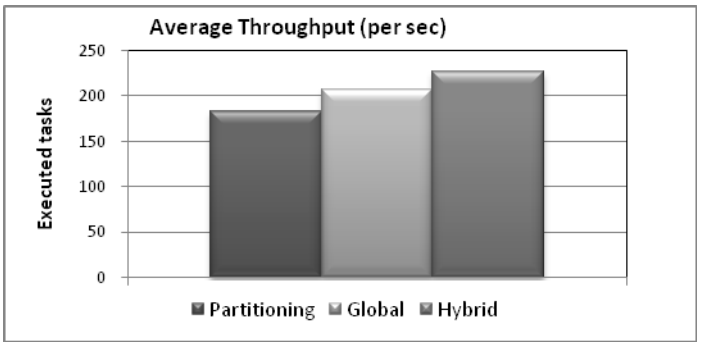

Fig. 14 comparison of average system throughput when number of tasks increases.

6. Conclusion and Future Work

Generally, there are two approaches for multiprocessor real-time scheduling, partitioning and global. Although the partitioning approach provides an acceptable overhead for the underlying system but it doesn't guarantee to be optimal. The global approach can provide this guarantee but it needs some preconditions to be hold; also, the most important deficit of the global approach is its considerable overheads.

In this paper, an intermediate hybrid multiprocessor real-time scheduling approach is proposed in which optimality will be reached via the minimum overheads for underlying system. Presenting and analyzing different feasible paradigms for combination of the two existing approaches, the proposed hybrid approach satisfies the two major goals of this combination: optimality and lightweightness. Experimental results showed that the hybrid approach outperforms the two existing ones.

Some future works are as follow:

- Providing proper algorithms w.r.t. this hybrid approach
- Formal analyzing of the proposed approach

Improving the usage of global-first method

References

[1] P. Holman and J. Anderson, *"Group-based Pfair Scheduling"*, *Real-Time Systems* , Volume 32, Numbers 1-2, pp. 125-168, February 2006.
[2] John Carpenter, et. al., "*A Categorization of Real-time Multiprocessor Scheduling Problems and Algorithms"*, in Handbook on Scheduling: Algorithms, Models and Performance Analysis, (2004).
[3] J. A. Stankovic, et. al., "*Misconceptions About Real-Time Databases*", Journal of Computer, Volume 32 Issue 6, June 1999.
[4] L. Sha, et. al., "*Real Time Scheduling Theory: A Historical Perspective*", Real-Time Systems, 28, pp. 101-155, 2004
[5] N. Baruah,et. al., *"Proportionate progress: A notion of fairness in resource allocation," Algorithmica*, vol. 15, pp. 600–625, 1996
[6] S. Baruah, J. Gehrke, and C. Plaxton, *"Fast scheduling of periodic tasks on multiple resources",* in Proceedings of the 9th International Parallel Processing Symposium, April 1995, pp. 280–288.
[7] J. Anderson and A. Srinivasan, *" Mixed Pfair/ERfair Scheduling of Asynchronous Periodic Tasks", Journal of Computer and System Sciences* , Volume 68, Issue 1, pp. 157-204, February 2004
[8] Anand Srinivasan, *"Effcient and Flexible Fair Scheduling of Real-time Tasks on Multiprocessors",* Ph. D. thesis, University of North Carolina at Chapel Hill, 2003
[9] J. Lopez, M. Garcia, J. Diaz, and D. Garcia. *"Worst-case utilization bound for EDF scheduling on real-time multiprocessor systems"*. In Proceedings of the 12th Euromicro Conference on Real-time Systems, pp. 25–33, June 2000
[10] A. Srinivasan, and J. H. Anderson, *"Efficient Scheduling of Soft Real-time Applications on Multiprocessors*", Journal of Embedded Computing, VOL. 1, NO. 3, pp. 51-59, June 2004
[11] Philip Holman and James H. Anderson, *"Using Supertasks to Improve Processor Utilization in Multiprocessor Real-time Systems",* 15th Euromicro Conference on Multiprocessor Real-Time Systems, ECRTS'03, pp.2-4, 2003
[12] Ali Safaei, et. al., "*QRS: A Quick Real-Time Stream Management System*", submitted to Journal of Real-Time Systems, feb, 2010.
[13] M. Alemi, "*Real-Time Task Scheduling in Data Stream Management Systems*", M. Sc. Thesis, Iran University of Science and Technology, 2011

Ali A. Safaei received his B.S. and M.S. degrees in computer engineering in 2001 and 2004, respectively. He is currently a PhD student of computer engineering at the Iran University of Science and Technology since 2005. His research interests include parallel and real-time query processing, quality of services and overload handling in data stream management systems, semantic cache and multiple-query optimization.

Mehdi Alemi received his B.S. degree in computer science in 2009. He is currently M.S. student of computer engineering at the Iran University of Science and Technology. He is interested in data stream systems, IR, data mining.

Mostafa S. Haghjoo is an associate professor at the Iran University of Science and Technology. He received his B.Sc. in mathematics
and computer science from the Shiraz University in 1976. He received his M.Sc. degree in computer science from the George Washington University in 1978. He obtained his Ph.D. degree in computer science from the Australian National University in 1995..

Shirin Mohammadi received her B.S. degree in computer engineering in 2007, She is currently a M.S. student of computer engineering at the Iran University of Science and Technology since 2008. Her research interests include data stream management systems, adaptive query processing, query response time estimation, and real time scheduling.