Abstract. In this work, we consider the scheduling problem of a set of periodic implicit-deadline and synchronous tasks, on a real-time multiprocessor composed of \( m \) identical processors. It is known that the cost of migrations and preemptions has significant influence on global system performances. The EKG algorithm can generate a great number of migrant tasks, but it has the advantage that each migrant task migrates between two processors only. Later, the EDHS algorithm has been proposed in order to minimize the number of migrant tasks of EKG. Although EDHS minimizes the number of migration compared to EKG, its drawback is the generation of additional preemptions caused by the migrations on several processors. In this paper we propose a new tasks allocation algorithm that aims to combine the advantages of EKG (migrations between two processors only) and those of EDHS (reduction of number of migrations).

Key words: Real time scheduling, EKG, semi-partitioning real time scheduling, inter processor migrations.

AMS subject classifications. 68W15, 68W10

1. Introduction. There are two basic approaches for scheduling real-time tasks on multiprocessor / multi-core platforms: global and partitioned scheduling. In partitioned scheduling, tasks are organized in groups, and each task group is assigned to a specific processor. After their allocation to processors, the tasks are not allowed to migrate from one processor to another. When selected for execution, a task can only be dispatched to its assigned processor. On each processor the tasks are scheduled using standard known uniprocessor algorithms e.g. RM (Rate-Monotonic) or EDF (Earliest-Deadline-First) [1]. The main disadvantage of the partitioning approach is that the tasks allocation problem is analogous to the bin packing problem which is known to be NP-Hard [2]. So a task cannot be assigned to any of the processors even if the total available capacity of the whole system is still large. When the individual task utilization is high, this waste could be significant, and in the worst-case only half of the system resource can be used. The alternative to partitioned scheduling is global scheduling in which there is a single queue for tasks that are ready to run. At each time, the \( m \) highest priority tasks are dispatched to any available processor according to a global priority scheme. The tasks can migrate from one processor to another which makes it possible to achieve a better use of the platform. Partitioned scheduling has gaps due to the absence of task migration from one processor to another. It is shown that a non schedulable system under partitioned policy can be scheduled if given the opportunity to unassigned tasks to run on multiple processors in global scheduling assuming that the cost of preemptions and migrations is neglected. This assumption is not realistic since this cost has an influence on global system performance. Several works have been proposed in the literature to reduce the number of preemptions and migrations [3, 4, 5].

To overcome the problem of the partitioned approach and increase the utilization rate of the system, recent works [6, 7, 8, 9, 10] have introduced the semi-partitioning scheduling in which most of tasks are assigned to particular processors as the partitioned scheduling, but the remaining tasks (unassigned tasks) are allowed to migrate between processors. In other words, each remaining task is splitted into a set of sub-tasks and each one of them is affected to a processor. This approach allows migration but reduces the number of migrant tasks compared to the global approach.

The Semi-partitioning algorithm EKG [11] cuts the set of processors into groups each one is composed of \( k \) processors and limits migration within the same group. In addition, a task can migrate between two processors only. Note that EKG allows to schedule optimally a set of periodic implicit tasks on \( m \) processors when \( k = m \) (EKG with one group). Since EKG allocates migrant and non-migrant tasks simultaneously, this can generate a great number of migrant tasks.

Kato et al. [12] have proposed the EDHS algorithm which improves the EKG algorithm. It proceeds into two separate steps to allocate the tasks: during the first one, the tasks are assigned according to a given partitioning algorithm in order to minimize the number of migrant tasks generated by the EKG algorithm. The second one consists in allocating migrant tasks on multiple processors according to a second algorithm.

Our contribution aims to combine the advantages of the EKG (migration between two processors only) and those of EDHS (reduction of migrant tasks). We proceed also into two steps to allocate the tasks: the first step is similar to the EDHS one, so we generate the same number of migrations as EDHS algorithm. In order to ensure the schedulability as
EKG algorithm, our proposed algorithm avoid that a task migrates between more than two processors during the second step of the algorithm. In order to achieve this goal, our key idea consists in reassigning the first allocated task of processors involved by migrations. In this case our algorithm achieves the optimality like EKG for $k = m$.

The remainder of this paper is organized as follows: in Section 2 we present EKG and EDHS algorithms. Section 3 is devoted to present our proposed algorithm. In Section 4 we present experimental simulations and finally we give the conclusion.

2. Presentation of the EKG algorithm. The system is composed of $n$ periodic, implicit-deadline and synchronous tasks noted $\tau_1, \tau_2, ..., \tau_n$ and $m$ identical processors $P_1, P_2, ..., P_m$. All the tasks cannot be executed in parallel and are independent. Each processor can’t execute more than one task at any time. Each task $\tau_i$ has a period $T_i$ (that is also the implicit deadline) and an execution time $C_i$. The ratio $C_i/T_i = U(\tau_i)$ defines the utilization rate of the task $\tau_i$.

The EKG algorithm [11] cut the set of processors into groups each one is composed of $k$ processors and limits migration within the same group. In addition, a task can migrate between two processors only. It allows scheduling optimally a set of periodic tasks with implicit deadline on $m$ processors, when setting the parameter $k$ equal to the number of processors ($k = m$).

Basic principle: Unlike partitioned algorithms, EKG allows tasks to run on two different processors (at different times, without parallelism). The algorithm is divided into two stages:

- Tasks allocation (offline): each task is assigned to one or two processors. The algorithm treats heavy and light tasks differently. A task $\tau_i$ is heavy if $C_i/T_i > SEP$, otherwise it is light where $SEP$ is calculated as follows:

$$SEP = \begin{cases} 1, & \text{if } k = m \\ k/(k+1), & \text{if } k < m \end{cases}$$

First, the algorithm assigns one heavy tasks to one processor where one processor is dedicated for one heavy (one per task). Then the lighter tasks are assigned to the remaining processors where several light tasks may be assigned to the same processor. To obtain a processor load of 1, some tasks can be split to run on two different processors (migrant task) belonging the same group. If a task is attempted to be assigned to the last processor in a group and it fails, then it is not split, but it is simply assigned to the first processor in a new group. This ensures that tasks in a group do not interact with tasks in another group.

- Tasks scheduling on processors (online): For each group, cutting the time into EKG intervals. An EKG interval is defined by two successive wake-up dates of tasks in the same group. It is similar to slots in the DP-Fair terminology [13], but limited to the tasks of the same group. Similar to DP-Fair, the work of migrant tasks should run for a time proportional to their utilization rate and duration of an interval $[t_0,t_1]$ where $t_0$ denotes the time when a task arrives, and $t_1$ denotes the time when any task in that group arrives next. On an interval $[t_0,t_1]$, if a task $\tau_i$ migrate between processors $P_j$ and $P_{j+1}$, it will be splitted into subtasks $\tau_{11}$ and $\tau_{12}$ as shown on Figure 2.1. At $t_0$ it runs on $P_j$ for $U(\tau_{11}) \ast (t_1 - t_0)$ time units and ends its execution at $timea$. Towards the end of the interval at $timeb$, the execution of the task restarts on $P_{j+1}$ for a time duration of $U(\tau_{12}) \ast (t_1 - t_0)$ units and ends its execution at $t_1$. The non migrant tasks are scheduled according to EDF on $[timea,timeb]$. After assignment of tasks, at runtime, our algorithm uses the same technique as the EKG algorithm to execute them on each processor.

Reducing the number of preemption by mirroring: The mirror technique called (Mirroring) can be easily implemented by inverting simply $\tau_1$ and $\tau_2$. This halves the number of preemptions. Figure 2.2 shows an execution with this technique. Note that it can be reused for other scheduling policies; it is the case for example DP-WRAP [11].

3. Presentation of EDHS Algorithm. EKG assigns migrant and non-migrant tasks simultaneously. This assignment produces several migrant tasks. To minimize the number of migrant tasks Kato et al. [12] have proposed the EDHS algorithm which proceeds into two separate steps: during the first one, the tasks are assigned according to a given partitioning algorithm in order to minimize the number of migrant tasks. The second step consists in allocating, on multiple processors, migrant tasks (tasks that have not been allocated during the first step), according to a second algorithm, as shown on Figure 3.1.

At runtime, the non-migrant tasks run according to EDF but migrant tasks run with high priority without overlap in time. When migrant task has exhausted its running time on a processor, it continues its execution immediately on the next
2.1. Migration of the task $\tau_{11}$ between processors $P_j$ and $P_{j+1}$.

2.2. The execution with mirroring technique.

3.1. Allocation of migrant tasks with EDHS algorithm.

Processor and preempts the current task as shown on Figure 3.2. Although EDHS minimizes the number of migration compared to EKG, its drawback is the generation of additional preemptions caused by the high priority of migrant tasks on several processors.

4. The proposed processor allocation heuristic. The system $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$ is composed of a periodic, implicit deadline and synchronous tasks. We assume that $\sum U(\tau_i) \leq m$ and $U(\tau_1) \geq U(\tau_2) \geq \ldots \geq U(\tau_n)$. The following notations are used in the remaining of the paper:

- $M$: denotes the set of the not allocated tasks. Initially $M = \tau$.
- $\tau[j]$: denotes the set of allocated tasks to the $j^{th}$ processor of the list of processors denoted by $P_j$, for $1 \leq j \leq m$.
Initially, \( \tau[j] = 0 \), for all \( j \).

- \( U[j] \): denotes the sum of the utilization rates of all tasks allocated to processor \( P_j \).
- A processor \( P_j \) is full if \( U[j] = 1 \).
- \( cap[j] = 1 - U[j] \): denotes the remaining capacity of processor \( P_j \).

In order to reduce the number of migrant tasks generated by the EKG algorithm, we proceed into two phases for allocating tasks to processors:

4.1. First phase. During the first phase, the allocation of tasks is done by applying one of the most known heuristics based on bin packing problem [14] as EDHS algorithm, namely First-Fit Decreasing, Best-Fit Decreasing and Worst-Fit Decreasing. These algorithms allocate the tasks by sorting them according to their utilization rates in the decreasing order. After this phase, a set of tasks remain still not allocated (the set of migrant tasks). Algorithm 1 describes the First-Fit Decreasing heuristic.

**Algorithm 1 First-Fit Decreasing heuristic**

```plaintext
for each \( \tau \) in \( M \) do
  \( j \leftarrow 1 \)
  affecter \leftarrow 1
  while ( \( U[j] + U(\tau) > 1 \) ) and affecter = 1 do
    \( j \leftarrow j + 1 \)
    if \( j > m \) then
      affecter \leftarrow 0
    end if
  end while
  if affecter = 1 then
    \( \tau[j] \leftarrow \tau[j] \cup \{ \tau \} \)
    \( U[j] \leftarrow U[j] + U(\tau) \)
    \( M \leftarrow M' \setminus \{ \tau \} \)
  end if
end for
```

4.1.1. Examples.

- **Example 1:** In the following example, we consider the tasks system \( \tau = (\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6) \) with \( U(\tau_1) = 0.7 \), \( U(\tau_2) = 0.6 \), \( U(\tau_3) = 0.4 \), \( U(\tau_4) = 0.4 \) and \( U(\tau_5) = 0.3 \). Figure 4.1 shows that the allocation of tasks using EKG algorithm gives rise to two migrant tasks \( \tau_2 (\tau_21 \text{ and } \tau_22) \) and \( \tau_1(\tau_11 \text{ and } \tau_12) \), but with the First-Fit Decreasing heuristic, there is no migrant task.
Reducing the Interprocessors Migrations of the EKG Algorithm

2.97

Example 2: In the following example we will show that even if we apply the heuristics based on the bin packing problem, we cannot avoid the migration of the tasks. We consider the system \( \tau = (\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6, \tau_7) \) with \( U(\tau_1)=0.9; U(\tau_2)=0.8; U(\tau_3)=0.5; U(\tau_4)=0.3; U(\tau_5)=0.3; U(\tau_6)=0.15 \) and \( U(\tau_7)=0.04 \). In Figure 4.2, it is clear that the First-Fit Decreasing heuristic could not affect the task \( \tau_5 \), so it must migrate on processors \( P_1, P_2 \) and \( P_3 \).

4.1.2. Experimentations. In Figure 4.3 [5] we compare the number of migrations obtained with the EKG and the proposed algorithms by using the heuristics First-Fit, Best-Fit, Worst-Fit. For simulations, we have considered 10000 task systems and we have calculated the average of the number of migrations in a given interval, for each heuristic. The tasks are randomly generated with the respect of the schedulability condition that is \( \sum U(\tau) \leq m \). Experimental results show that the heuristics First-Fit, Best-Fit, Worst-Fit reduce significantly the number of migrations. The reduction can reach 60% with the Best Fit and the First Fit heuristics.

4.2. Second phase. The second phase consists in allocating the set of remaining tasks (set of migrant tasks). Note that, by construction, the sum of utilization rates of migrant tasks is lower or equal to the sum of remaining processor capacities. Assume that, processors are sorted by decreasing order according to their remaining capacities, \( \text{cap}[1] \geq \text{cap}[2] \geq \cdots \geq \text{cap}[m] \) and task \( \tau_k \) can migrate on processors \( P_1, P_2, \ldots, P_h \) which means that \( \text{cap}[1] + \text{cap}[2] + \cdots + \text{cap}[h] \geq U(\tau_k) \) and \( \text{cap}[1] + \text{cap}[2] + \cdots + \text{cap}[h-1] < U(\tau_k) \); the width of a time interval is denoted \( L \).

Note that according to the first phase of the heuristic, the first task of each processor \( P_j \), for \( 2 \leq j \leq h \), noted \( \rho_j \), verifies \( U(\rho_j) \geq U(\tau_k) \). The basic idea is to increase recursively the remaining processor capacities as follows:

- Subdividing the task \( \rho_2 \) into two subtasks \( \rho_{21} \) and \( \rho_{22} \), such that \( U(\rho_{22}) = \text{cap}[1] \) and \( U(\rho_{21}) = U(P_2[1]) - \text{cap}[1] \)
- Assigning \( \rho_{22} \) to \( P_1 \). In this case \( P_1 \) becomes full and the capacity of \( P_2 \) is increased with \( U(\rho_{22}) = \text{cap}[2] + U(\rho_{22}) \). Thus, task \( \rho_{21} \) becomes a migrant task on processors \( P_1 \) and \( P_2 \). At runtime:
  - Processor \( P_2 \) starts its execution by task \( P_2[1] \) during \( U(\rho_{21})L \).
  - Processor \( P_1 \) ends its execution by task \( P_2[1] \) during \( U(\rho_{22})L \).
- Recursively, the same process is repeated between processors \( P_{j-1} \) and \( P_j \), for \( 2 < j < h \), where the task \( \rho_j \) is subdivided into two subtasks \( \rho_{j1} \) et \( \rho_{j2} \), such that \( U(\rho_{j2}) = \text{cap}[j-1] \). In this case \( \text{cap}[j] = \text{cap}[j] + U(\rho_{j2}) \). At runtime:
In Figure 4.4 we show the steps of the algorithm in order to allocate a migrant task $\tau_k$ to two processors only instead of allocating it to four processors.

5. Conclusion. In this work we have proposed a new semi-partitioned algorithm that reduces the number of migrations like EDHS and limits migrations between two processors only like EKG. The proposed algorithm is designed into

---

**Algorithm 2** allocation of migrant tasks

```plaintext
for each $\tau_k$ in $M$
    sort in decreasing order the list of processors according to their remaining capacities
    calculate $h$ such as $\text{cap}[h-1] + \text{cap}[h] \geq U(\tau_k)$ and $\text{cap}[h-1] + \text{cap}[h-2] < U(\tau_k)$.
    $j \leftarrow 1$
    while $j < h - 1$
        Subdivide $\rho_{j+1}$ into two subtasks $\rho_{j+1,1}$ and $\rho_{j+1,2}$ such that $U(\rho_{j+1,1}) = \text{cap}[j]$ and $U(\rho_{j+1,2}) = U(\rho_{j+1}) - \text{cap}[j]$
        Assign $\rho_{j+1,1}$ to $P_j$, then $\text{cap}[j+1] = \text{cap}[j+1] + \text{cap}[j]$ and $P_j$ becomes full.
        Processor $P_j$ starts its execution by executing task $\rho_{j+1}$ during $U(\rho_{j+1,1}) \ast L$
        Processor $P_j$ ends its execution by executing task $\rho_{j+1,2}$ during $U(\rho_{j+1,2}) \ast L$.
        $j \leftarrow j+1$
    end while
    /* $\tau_k$ migrates only between $P_{h-1}$ and $P_h$ */
    Subdivide $\tau_k$ into two subtasks $\tau_{k,1}$ and $\tau_{k,2}$, such that $U(\tau_k) = \text{cap}[h-1]$ and $U(\tau_{k,2}) = U(\tau_k) - U(\tau_{k,1})$.
    Assign $\tau_{k,1}$ to $P_{h-1}$ and $\tau_{k,2}$ to $P_h$
    Processor $P_{h-1}$ starts its execution by task $\tau_{k,1}$ during $U(\tau_{k,1}) \ast L$
    Processor $P_h$ ends its execution by task $\tau_{k,2}$ during $U(\tau_{k,2}) \ast L$.
end for
```
two steps. During the first step we used the well-known bin packing heuristics [14] (the First Fit Decreasing, the Best Fit Decreasing and the Worst fit Decreasing). This step is similar to the first step of the EDHS and it consists in reducing the number of migrant tasks compared to EKG. During the second step of the algorithm, we proposed a new technique that allocates the migrant tasks. Our key idea consists in increasing the number of migrant tasks, each one migrates on two processors, while keeping the same number of migrations: instead to migrate a task between h processors (h-1 migrations), we migrate (h-1) tasks each one between two processors. This reallocation has the advantage that we remain in the same condition of optimality of the EKG for m=k. Experimental simulations show that the number of migrations, compared to EKG, is significantly reduced. This reduction can reach 60%.

REFERENCES


Edited by: Dana Petcu
Received: Mar 29, 2018
Accepted: Aug 24, 2018