# An Investigation of Synchronization under Multiprocessors Hierarchical Scheduling*

Farhang Nemati, Moris Behnam and Thomas Nolte
Mälardalen Real-Time Research Centre
Mälardalen University, Box 883, 72123, Sweden
{farhang.nemati, moris.behnam, thomas.nolte}@mdh.se

## Abstract

*In the multi-core and multiprocessor research community, considerable work has been done on real-time multiprocessor scheduling algorithms where it is assumed the tasks are independent. However in practice a typical real-time system includes tasks that share resources. On the other hand, synchronization in the multiprocessor context has not received enough attention.*

*In this paper we propose an extension to multiprocessor hierarchical scheduling to support resource sharing. We extend the scheduling framework with an existing synchronization protocol for global scheduling in multi-core systems.*

## 1. Introduction

Multi-core and multiprocessor architectures are receiving more interest due the performance they offer as improving performance in single-core architectures is limited due to the problems with power consumption and related thermal problems.

To take advantage of the performance offered by a multi-core/multiprocessor architecture, appropriate scheduling algorithms and synchronization protocols are required. However, in the research community, scheduling has received much more attention than synchronization [7].

There are two main approaches for scheduling sporadic and periodic task systems on multiprocessor architectures [2, 3, 11, 15]; partitioned and global scheduling. Under partitioned scheduling tasks are statically assigned to processors and tasks within each processor are scheduled by uniprocessor scheduling such as FPS (Fixed Priority Scheduling) or EDF (Earliest Deadline First). Under global scheduling, e.g., G-EDF (Global Earliest Deadline First), tasks are scheduled by a single scheduler and each task can be executed on any core. A combination of global and partitioned scheduling called the two-level hybrid scheduling [10] is used for systems in which some tasks cannot migrate between cores while other tasks can migrate.

A more general approach which is a generalization of partitioned and global scheduling is called *cluster-based* scheduling [19]. In this approach tasks are statically assigned to clusters and tasks within each cluster are globally scheduled. In turn, clusters are transformed into tasks and scheduled on multiprocessor architectures. Cluster-based scheduling seems to be the way to improve utilization bounds on the multiprocessor platform. However the existing approaches for cluster-based scheduling do not consider synchronization and assume that tasks are independent.

The contribution of this paper is an extension to the hierarchical scheduling framework for multiprocessor virtual clustering presented in [19], to consider lock-based synchronization. We will explain this framework later in more detail. We have extended the scheduling condition for each cluster to consider blocking times. Some assumptions of the scheduling framework also need to be changed. We have used a technique similar to an existing protocol for synchronization under global scheduling proposed in [7].

**Related work.** In the context of uniprocessor hierarchical scheduling, there have been studies on allowing for sharing of mutually exclusive resources within components [1, 16] and across components [5, 9, 12].

For multiprocessor systems, Rajkumar present MPCP (Multiprocessor Priority Ceiling Protocol) [18], which extends PCP to multiprocessors hence allowing for synchronization of tasks sharing mutually exclusive resources using partitioned FPS. Gai et al. [13, 14] present MSRP (Multiprocessor SRP), which is a P-EDF (Partitioned EDF) based synchronization protocol for multiprocessors. The shared resources are classified as either (i) local resources that are shared among tasks assigned to the same processor, or (ii) global resources that are shared by tasks assigned to different processors. In MSRP, tasks synchronize local resources using SRP, and access to global resources is guaranteed a bounded blocking time. Lopez et al. [17] present an implementation of SRP under P-EDF. Devi et al. [10] present a synchronization technique under G-EDF. The work is restricted to synchronization of non-nested accesses to short, simple objects, e.g., stacks, linked lists, and queues. In addition, the main focus of the method is on soft real-time systems.

Block et al. [7] present FMLP (Flexible Multiprocessor Locking Protocol), which is the first synchronization protocol for multiprocessors that can be applied to both partitioned and global scheduling algorithms, i.e., P-EDF and G-EDF. We will use this protocol for synchronization under the hierarchical scheduling for multiprocessor virtual clustering; hence we will spend more time on details of this protocol in Section 3.

## 2. Task and system model

We assume a sporadic task model [4] in which a sporadic task $\tau_i$ is specified by its minimum inter arrival time $T_i$, its

* This work was partially supported by the Swedish Foundation for Strategic Research (SSF) via the strategic research centre (PROGRESS) at Mälardalen University.

worst-case execution time $C_i$, and its relative deadline $D_i$. We refer to $j^{th}$ *job* (each being an instance of a task) of task $\tau_i$ as $\tau_i^j$.

A request *R* issued by a job for exclusive access to a resource *l* is satisfied as soon as the job holds the resource. A request which is not contained within any other request is called an *outermost*.

We assume a multiprocessor system consisting *m* identical, unit-capacity processors each of which has a scheduling utilization of one. We also assume that migration of job is allowed, i.e., a job can be preempted on one processor and be resumed on another processor. Preemption and migration overheads are assumed to be negligible.

## 3. FMLP

In the FMLP, resources are categorized into *short* and *long* resources which is user defined. There is no limitation on nesting resource accesses, except that requests for long resources cannot be nested in requests for short resources.

In FMLP, deadlock is prevented by grouping resources. A group includes either global or local resources, and two resources are in the same group if a request for one may be nested in a request for the other one. A group lock is assigned to each group and only one task at any time can hold the lock.

Under FMLP, the jobs that are blocked on short resources perform busy-wait and are added to a FIFO queue. Jobs that access short resources hold the group lock and execute non-preemptively. A job accessing a long resource under G-EDF holds the group lock and executes preemptively using priority inheritance, i.e., it inherits the maximum priority of any higher priority job blocked on any resource within the same group. Tasks blocked on a long resource are added to a FIFO queue.

Actually FMLP works under a variant of G-EDF for *suspendable and preemptable* jobs (GSN-EDF) [7] which guarantees that a job $\tau_i^j$ can only be blocked (with a constraint duration) by another non-preemptable job when job $\tau_i^j$ is released or resumed.

### 3.1. Blocking under GSN-EDF and FMLP

**Busy-wait blocking** of task $\tau_i$ specified by $BW_i$ is the maximum duration of time that any job of the task can busy-wait on a short resource.

**Non-preemptive blocking** occurs when a preemptable job $\tau_i^j$ is one of the *m* highest priority jobs but it is not scheduled because a lower priority job is non-preemptively executing instead. Non-preemptive blocking of task $\tau_i$ denoted by $NPB_i$ is the maximum duration time that any job of task $\tau_i$ is non-preemptively blocked.

**Direct blocking** occurs when job $\tau_i^j$ is one of the *m* highest priority jobs but it is suspended because it issues a request for an outermost long resource from group *G* but another job holds a resource from the same group (holds the group's lock). Direct blocking of task $\tau_i$ specified by $DB_i$ is the maximum duration of time that any job of the task can be direct blocked.

## 4. Hierarchical scheduling for multiprocessor virtual clustering

Under Cluster-based scheduling tasks are statically assigned to clusters and scheduled globally among themselves (*intra-cluster scheduling*). A cluster is a set of $m'$ processors where $m' \leq m$. A cluster with its tasks and scheduler is denoted as a *component*. The clusters are in turn globally scheduled on the multiprocessor (*inter-cluster scheduling*). The cluster-based scheduling is a generalization of partitioned and global scheduling

Cluster-based scheduling can be *physical* or *virtual*. In physical cluster-based scheduling each of cluster's $m'$ processors are statically mapped to one of $m$ processors of the multiprocessor [8]. In the virtual cluster-based scheduling the $m'$ processors of each cluster are dynamically mapped on $m^o$ out of $m$ processors of the multiprocessor. Virtual clustering is more general and less sensitive to task-cluster mapping compared to physical clustering.

Physical clustering only needs the intra-cluster scheduling because the clusters do not share processors, On the other hand, virtual clustering requires a hierarchical scheduling which includes intra-cluster and inter-cluster scheduling. Under hierarchical scheduling processors of the multiprocessor are dynamically assigned to virtual clusters (inter-cluster scheduling) and processor resources assigned to a virtual cluster are used by that cluster to schedule its tasks (intra-cluster scheduling).

### 4.1. Multiprocessors resource model

The notion of *component interface* is used to specify the required processor resources to schedule the tasks within the component [20]. A multiprocessor resource model specifies the characteristics of resource provided to a cluster by the multiprocessor platform. As a component interface, a multiprocessor resource model specifies the resource requirement for the component.

A *multiprocessor periodic resource* (MPR) model denoted by $\Gamma = \langle \Pi, \theta, m' \rangle$ specifies that the multiprocessor collectively provides $\theta$ units of processor resource in every $\Pi$ time units to a cluster consisting $m'$ processors. A feasible MPR model has to satisfy $\theta/\Pi \leq m'$.

The lower bound of amount of resource supply that a resource model $\Gamma$ in time interval *t* provides is specified by supply bound function $sbf_\Gamma(t)$. In schedulability conditions, $sbf$ is used to generate MPR based component interfaces. The $sbf$ for MPR model $\Gamma = \langle \Pi, \theta, m' \rangle$ is presented as follows [19]:

$$sbf_\Gamma(t) = \begin{cases} k\theta + \max\{0, (I - k\Pi)m' + \theta\}, & t \geq \Pi - \left\lceil \frac{\theta}{m'} \right\rceil \\ 0 & , \ otherwise \end{cases} \quad (1)$$

where $k = \left\lfloor \frac{t - (\Pi - \left\lceil \frac{\theta}{m'} \right\rceil)}{\Pi} \right\rfloor$ and $I = t - 2\Pi + \left\lceil \frac{\theta}{m'} \right\rceil$

### 4.2. Component processor demand

The *workload* of a task $\tau_i$ in an interval $[a, b]$ is total duration of all intervals that any job of task $\tau_i$ is executing. The task workload consists of (i) the *carry-in* demand which is generated by a job of task $\tau_i$ released before $a$ but did not

complete its execution until $a$ (ii) the summation of demands of all jobs of task $\tau_i$ with their both release time and deadline within the interval $[a, b]$ (iii) the *carry-out* demand is generated by a job of task $\tau_i$ with release time in the interval $[a, b)$ but did not complete its execution until $b$.

An upper bound for workload of task $\tau_i$ under G-EDF in an interval $[a, b]$ has been obtained [6] under two assumptions; some job $\tau_k^j$ has a deadline at $b$ and $\tau_k^j$ misses its deadline. This upper bound workload of task $\tau_i$ is specified by $W_i(t)$ in interval $[a, b]$ with length $t = b - a$.

### 4.3. Schedulability condition

The schedulability condition of a component, $C$ (compromising a cluster and its scheduler), using the MPR model $\Gamma = \langle \Pi, \theta, \mathrm{m}' \rangle$, is obtained under assumption that some job of task $\tau_k$ (denoted as $\tau_{k,b}$) has deadline at $b$ and it misses its deadline. Then to check the schedulability of the task $\tau_k$ all different values for $a$ should be considered such that (i) at least one of the $\mathrm{m}'$ processors is idle at $a$ (such a time instant is denoted as $t_{idle}$), (ii) $a \leq r$, where $r$ is the release time of job $t_{k,b}$, (iii) there is no $t_{idle}$ in the interval $(a, r]$. Figure 1 depicts one such instant.

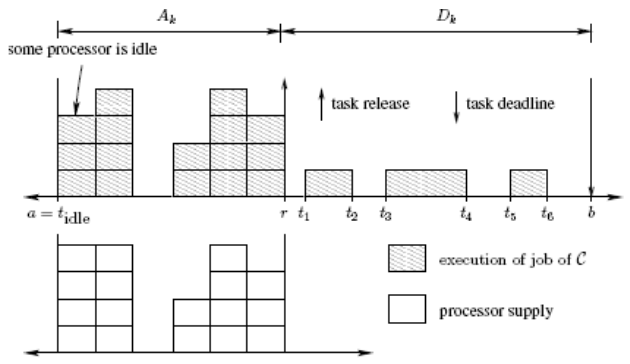


**Figure 1** [19]

To check schedulability of component $C$, for each task $\tau_k$ of $C$, all intervals $[a, b]$ with assumptions explained above are considered and the condition that guarantees a deadline miss for job $t_{k,b}$ is derived. Let denote $H(\tau_{k,b})$ as the set of jobs of all tasks with priority higher or equal to the priority of $t_{k,b}$. When $t_{k,b}$ misses its deadline it means that in interval $[a, b]$ the total workload of all jobs in $H(\tau_{k,b})$ is greater than the processor supply for $C$ in the interval:

$$\sum_{i=1}^{n} I_i > sbf_{\Gamma}(A_k + D_k) \qquad (2)$$

where $I_i$ denotes the total workload in interval $[a, b]$ of all jobs of task $\tau_i$ in $H(\tau_{k,b})$, $A_k$ denotes the interval $[a, r]$ and $A_k + D_k$ shows the length of interval $[a, b]$ (Figure 1).

If it can be shown that for all tasks $\tau_k$ and for all values of $A_k$ equation (2) is invalid, then $C$ is schedulable.

To obtain an upper bound for each $I_i$, the workload of task $\tau_i$ in two interval classes; (C1) time intervals within $[a, b]$ where job $\tau_{k,b}$ executes, and (C2) other intervals in $[a, b]$. These two interval classes are denoted by $I_{i,1}$ and $I_{i,2}$ respectively.

The upper bound for total workload of all tasks in the interval, specified by $dem(A_k + D_k, m')$, is obtained in [19].

The component $C$ is schedulable if for all tasks $t_k$ within $C$ and all for values $A_k$:

$$dem(A_k + D_k, m') \leq sbf_{\Gamma}(A_k + D_k) \qquad (3)$$

## 5. Synchronization under Multiprocessor Hierarchical Scheduling

The multiprocessor hierarchical scheduling explained in Section 4 assumes that tasks within components (clusters) are independent and do not share any resources other than the processor. We now extend the assumptions of the framework with that tasks within a component may share resources and require exclusive access to them.

Since the FMLP protocol (Section 3) works under GSN-EDF (a variant of G-EDF), we assume that components in which tasks share resources use this scheduling protocol. The upper bound workload for each task as well as the schedulability condition has to be extended to consider the blocking time overheads generated under GSN-EDF and FMLP.

As it was shown in Section 3, the total blocking time for each task $\tau_i$ consists of three terms ($BW_i, NPB_i$, and $DB_i$). A job in its busy-wait intervals is executing, although it does not perform any work, hence its worst-case execution time $C_i$ can be increased by $BW_i$. The new worst-case execution time of any job of task $\tau_i$, denoted by $C'_i$ is $C'_i = C_i + BW_i$. Thus we replace $C_i$ by $C'_i$ in the upper bound workload of task $\tau_i$ in [19]:

$$W_i(t) = N_i(t)C'_i + CI_i(t) \qquad (4)$$

where $N_i(t) = \left\lfloor \frac{t + (T_i - D_i)}{T_i} \right\rfloor$ and
$CI_i(t) = min\{C'_i, max\{0, t - N_i(t)T_i\}\}$.

**Schedulability condition.** To derive a schedulability condition:
**(a)** We have to extend $I_i$ (the total workload in interval $[a, b]$ of all jobs of task $\tau_i$ in $H(\tau_{k,b})$) to include busy-waits of jobs in $H(\tau_{k,b})$.

The upper bound for total workload of all tasks in intervals of type $I_{i,1}$ in [19] is extended by busy-wait times as follows:

$$\sum_{i=1}^{n} I_{i,1} \leq \mathrm{m}' C_k'$$

The upper bound for $I_{i,2}$ should also be extended in the same way to include the busy-wait times. Note that according to (4), $W_i(A_k + D_k)$, $CI_i(A_k + D_k)$, $W_k(A_k + D_k)$ and $CI_k(A_k + D_k)$ already include busy wait times:

$$I_{i,2} \leq \bar{I}_{i,2} = min\{W_i(A_k + D_k), A_k + D_k - C_k'\}, \qquad i \neq k$$

$$I_{k,2} \leq \bar{I}_{k,2} = min\{W_k(A_k + D_k) - C_k', A_k\}, \text{ and}$$

By definition of $t_{idle}$, at most $\mathrm{m}' - 1$ jobs can be executing at time instant $a$, thus it is only needed to consider $\mathrm{m}' - 1$ largest carry-in demands ($CI_i$):

$$\hat{I}_{i,2} = min\{W_i(A_k + D_k) - CI_i(A_k + D_k), A_k + D_k - C_k'\}$$
$$where\ i \neq k$$

$$\hat{I}_{k,2} = min\{W_k(A_k + D_k) - C_k' - CI_k(A_k + D_k), A_k\}$$

**(b)** We have to extend the condition in Section 4.3 under which the deadline miss for job $\tau_{k,b}$ is guaranteed in interval $[a, b]$. The condition states that the total workload of all jobs in $H(\tau_{k,b})$ is greater than the processor supply for $C$ in the interval. In the presence of lock-based resources the condition must be extended. We denote $B(\tau_{k,b})$ as the set of jobs of all tasks with priority less than the priority of $\tau_{k,b}$. The total workload in addition to workload for jobs in $H(\tau_{k,b})$ within interval $[a, b]$ must also include:

1. The total non-preemptively execution parts of all jobs in $B(\tau_{k,b})$. An upper bound for this is denoted by $NPB(\tau_{k,b})$.
2. The total busy-waiting of all jobs in $B(\tau_{k,b})$. We denote the upper bound for the total busy-waiting as $BWB(\tau_{k,b})$.
3. The total direct blocking of all jobs of all tasks in $H(\tau_{k,b})$ in interval $[a, b]$. An upper bound for the total direct blocking is specified by $DBH(\tau_{k,b})$.

Considering the three terms, the total workload of jobs in interval $[a, b]$, specified by $WB(\tau_{k,b})$, will be:

$$WB(\tau_{k,b}) = NPB(\tau_{k,b}) + BWB(\tau_{k,b}) + DBH(\tau_{k,b}) \quad (5)$$

Currently we are working on obtaining upper bounds for these three terms.
Considering (a) and (b) we will have:

$$dem(A_k + D_k, m') = \sum_{i=1}^{n} \hat{I}_{i,2} + \sum_{\substack{m'-1 \\ largest}} (\bar{I}_{i,2} - \hat{I}_{i,2}) + m' C_k{}' + WB(\tau_{k,b})$$

Now we can use the condition (3) with the new workload upper bound as the schedulability condition of component $C$.

## 6. Summary

We have discussed a way of generalizing for multiprocessor hierarchical scheduling framework presented by [19], through allowing for shared logical resources between tasks within the same component. We have used a technique similar to FMLP [7] to synchronize the access of shared resources by tasks, and we have shown how the synchronization will affect the schedulability analysis of the hierarchical framework. Currently, we are working on deriving upper bounds for different blocking times in equation (5). After that, we will evaluate this approach by means of simulation and implementation.

## References


[1] L. Almeida and P. Pedreiras. Scheduling within temporal partitions: response-time analysis and server design. In 4th *ACM international conference on Embedded software (EMSOFT'04)*, Sep. 2004.

[2] T. Baker. A comparison of global and partitioned EDF schedulability test for multiprocessors. Technical report, January 2005.

[3] S. Baruah and N. Fisher. The partitioned multiprocessor scheduling of sporadic task systems. In *RTSS '05: Proceedings of the 26th IEEE International Real-Time Systems Symposium*, pages 321–329, Washington, DC, USA, 2005. IEEE Computer Society.

[4] S. Baruah, A. Mok, and L. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Proceedings of the* 11th *IEEE International Real-Time Systems Symposium(RTSS'90)*, pages 182–190, Lake Buena Vista, Florida, USA, December 1990. IEEE Computer Society.

[5] M. Behnam, I. Shin, T. Nolte, and M. Nolin. SIRAP: a synchronization protocol for hierarchical resource sharing in real-time open systems. In 7th *ACM and IEEE Int. Conference on Embedded Software (EMSOFT'07)*, Oct. 2007.

[6] M. Bertogna, M. Cirinei, and G. Lipari. Improved schedulability analysis of edf on multiprocessor platforms. In *ECRTS*, 2005.

[7] A. Block, H. Leontyev, B. Brandenburg, and J. Anderson. A flexible real-time locking protocol for multiprocessors. In *Embedded and Real-Time Computing Systems and Applications, 2007. RTCSA 2007. 13th IEEE International Conference on*, pages 47–56, Aug. 2007.

[8] J. M. Calandrino, J. H. Anderson, and D. P. Baumberger. A hybrid real-time scheduling approach for large-scale multicore platforms. In *ECRTS*, 2007.

[9] R. I. Davis and A. Burns. Resource sharing in hierarchical fixed priority pre-emptive systems. In 27th *IEEE Int. Real-Time Systems Symposium (RTSS'06)*, Dec. 2006.

[10] U. Devi, H. Leontyev, and J. Anderson. Efficient synchronization under global edf scheduling on multiprocessors. In *Real-Time Systems, 2006. 18th Euromicro Conference on*, pages 10 pp.–84, 0-0 2006.

[11] U. C. Devi. Soft real-time scheduling on multiprocessors. PhD thesis, Chapel Hill, NC, USA, 2006. Adviser Anderson, James H.

[12] N. Fisher, M. Bertogna, and S. Baruah. The design of an EDF-scheduled resource-sharing open environment. In 28th *IEEE Real-Time Systems Symposium (RTSS'07)*, Dec. 2007.

[13] P. Gai, G. Lipari, andM. D. Natale. Minimizing memory utilization of real-time task sets in single and multi-processor systems-on-a-chip. In *RTSS '01: Proceedings of the 22nd IEEE Real-Time Systems Symposium*, page 73, Washington, DC, USA, 2001. IEEE Computer Society.

[14] P. Gai, M. D. Natale, G. Lipari, A. Ferrari, C. Gabellini, and P. Marceca. A comparison of MPCP and MSRP when sharing resources in the janus multiple-processor on a chip platform. In *RTAS '03: Proceedings of the The 9th IEEE Real-Time and Embedded Technology and Applications Symposium*, page 189, Washington, DC, USA, 2003. IEEE Computer Society.

[15] J. Carpenter, S. Funk, P. Holman, A. Srinivasan, J. Anderson, and S.Baruah. A categorization of real-time multiprocessor scheduling problems and algorithms. In *J. Y. Leung, editor, Handbook on Scheduling Algorithms, Methods, and Models*, pages 30.1–30.19. ChapmanHall/CRC, Boca Raton, Florida, 2004.

[16] T.-W. Kuo and C.-H. Li. A fixed-priority-driven open environment for real-time applications. In 20th *IEEE International Real-Time Systems Symposium (RTSS'99)*, Dec. 1999.

[17] J. M. Lopez, J. L. Dıaz, and D. F. Garcia. Utilization bounds for edf scheduling on real-time multiprocessor systems. *Real-Time Syst.*, 28(1):39–68, 2004.

[18] R. Rajkumar. *Synchronization in multiple processor systems. In Synchronization in Real-Time Systems: A Priority Inheritance Approach*. Kluwer Academic Publishers, 1991.

[19] I. Shin, A. Easwaran, and I. Lee. Hierarchical scheduling framework for virtual clustering of multiprocessors. In *Proceedings Of the 20th Euromicro Conf. on Real-Time Systems*, pages 181-190, July 2008.

[20] I. Shin and I. Lee. Periodic resource model for compositional real-time guarantees. In 24th *IEEE International Real-Time Systems Symposium (RTSS'03)*, Dec. 2003.