# Comparison of Empirical Success Rates of Global vs. Partitioned Fixed-Priority and EDF Scheduling for Hard Real Time TR-050601

Theodore P. Baker

Department of Computer Science
Florida State University
Tallahassee, FL 32306-4530
e-mail: baker@cs.fsu.edu

1 July 2005*

**Abstract**

Improvements in schedulability tests for global fixed-priority and EDF scheduling in a homogeneous multiprocessor (symmetric multiprocessing) environment have shown that the *worst-case* guaranteed achievable utilization levels for global EDF scheduling equals what can be achieved with partitioned scheduling, and both ways of applying EDF scheduling out-perform fixed-priority scheduling, for sets of independent periodic or sporadic hard-deadline tasks with deadline equal to period. However, less is known about the comparative performance of the partitioned vs. global and EDF vs. fixed-priority approaches in the average and without the restriction that deadline equal period, and particular which of the known combinations of a scheduling algorithm and a sufficient a priori test of schedulability is more likely to succeed in verifiably scheduling a set of tasks to meet all deadlines. This paper compares the performance of several such combinations on a variety of pseudo-randomly chosen sets of sporadic tasks.

## 1 Introduction

Judging from the majority of published accounts, the predominant approach to scheduling multiprocessor hard-real-time systems is partitioned, in which each task is assigned statically (more or less) to one processor. Partitioned scheduling has the virtue of permitting schedulability to be verified using well-understood single-processor analysis techniques. It once seemed this was the only sensible approach to obtain hard real-time performance guarantees, since the analysis of global multiprocessor scheduling lagged far behind the analysis of the single-processor case.

The global approach now deserves more consideration. Recent improvements in the worst-case analysis of global (single job queue) hard-deadline multiprocessor scheduling (e.g., [1, 4, 9, 10, 19, 3]) show that, in terms of the worst-case achievable utilization on systems of hard-deadline sporadic and periodic tasks with deadline equal to period, global scheduling equals or approaches the performance of partitioned scheduling. Global EDF scheduling is able to guarantee worst-case schedulability up to the same processor utilization level as partitioned

*Last revision *Date* : 2005/07/2000 : 13 : 43.

EDF scheduling. The same cannot quite be said for global RM scheduling, as there no known tight utilization bound and the best current approximation is lower than the best known bound for partitioned RM scheduling. Still, the utilization bounds are only achieved for worst-case task sets, and the bounds apply only to cases where deadline equals period. There are certainly classes of systems – including any system where the tasks with utilization greater than 1/2 outnumber the processors – where partitioned scheduling will not work but global scheduling may be feasible. Moreover, it is a well-known result of queueing theory that single-queue scheduling produces better average response times than queue-per-processor scheduling[12]. After all, a partitioned schedule can result in a processor being idle while another processor has backlogged jobs, while a global schedule for the same task set would not idle any processor while there are any backlogged jobs.

This paper attempts to shed more light on the relative merits of partitioned vs. global fixed task-priority scheduling and EDF scheduling, by measuring the performance of several variations of partitioned and global scheduling on collections of pseudo-randomly generated task sets. Unlike the utilization bound comparisons mentioned above, this empirical comparision is not limited to worst-case combinations of periods, execution times, and deadlines (which can be avoided in practice by sensible design), nor to cases where task's deadlines are equal to their periods.

# 2 Prior Work

As mentioned above, the theoretical worst-case achievable processor utilizations of the global and partitioned scheduling approaches are very similar, for sporadic or aperiodic task sets with deadline equal to period.

Andersson, Baruah, and Jonsson[1] showed that the utilization guarantee for any static-job-priority multiprocessor scheduling algorithm – partitioned or global – cannot be higher than $(m + 1)/2$ for an $m$-processor platform. Doing such partitioning optimally is reducible to the bin packing and integer partition problems, which are known to be NP complete. Therefore, research has focused on the analysis of heuristic algorithms for the assignment of tasks to processors, and on bounding how badly they can do compared to an optimal algorithm. Some of this research has looked at average-case performance. For example, in [13], Lee *et al.* studied a general multi-resource allocation problem, and empirically compared the average solution quality and execution time of a search-bounding heuristic algorithm against those of dynamic programming and integer linear programming algorithms. Other research has attempted to find tight bounds on the worst-case performance of heuristic partitioning algorithms.

## 2.1 Partitioned Scheduling

**Fixed Priority** Oh and Baker[17] showed that Liu and Layland's $m(2^{1/m} - 1)$ utilization bound test for single processor rate monotone scheduling could be applied to show that any system of independent periodic tasks with total utilization $U < m(2^{1/2} - 1)$ can be scheduled on $m$ processors using First-Fit Decreasing (FFD) assignment of tasks to processors and RM local scheduling. They also showed that for any $m \geq 2$ there is a systems of tasks with $U = (m + 1)/(1 + 2^{1/(m+1)})$ that cannot be scheduled with $m$ processors using any partitioning algorithm and RM local scheduling.

Lopez, Diaz and Garcia[14] refined and generalized this result, showing that any system of periodic tasks with total individual utilizations $u_i \leq u_{\max}$ and total utilization $U < (m\beta_{LLB} + 1)(2^{1/(\beta_{LLB}+1)} - 1)$ can be scheduled on $m$ processors using First-Fit Decreasing (FFD) or any other "reasonable allocation decreasing" assignment of of tasks to processors, where where $\beta_{LLB} = \lfloor 1/\log_2(u_{\max} + 1) \rfloor$. They showed that this result is tight for "reasonable" partitioning schemes based on the Liu & Layland utilization bound, and claimed that the tightness extends to all other partitioning schemes. For the unrestricted case, where $u_{\max} = 1$ and $\beta_{LLB} = 1$, this would mean that the guaranteed utilization bound is $(m + 1)(2^{1/2} - 1)$.

 

**EDF** Lopez, Diaz, Garcia and Garcia[15] obtained a similar result for partitioned EDF scheduling, showing that it is possible to schedule on $m$ processors any system of $n$ independent periodic tasks with maximum individual utilization $u_{\max}$ and total utilization $U < \frac{m\beta_{EDF}+1}{\beta_{EDF}+1}$ where $\beta_{EDF} = \lfloor 1/u_{\max} \rfloor$. For the unrestricted case, where $u_{\max} = 1$ and $\beta_{EDF} = 1$, this says the guaranteed utilization bound is $(m+1)/2$. It follows from Andersson, Baruah, and Jonsson[1] that this result is tight.

## 2.2 Global Scheduling

**Fixed Priority** Andersson, Baruah, and Jonsson[1] looked at global rate monotonic scheduling on multiprocessors, and showed that any system of independent periodic tasks for which the utilization of every individual task is at most $m/(3m-2)$ can be scheduled successfully on $m$ processors using RM scheduling if the total utilization is at most $m^2/(3m-1)$. Baruah and Goossens[4] also looked at global RM scheduling, showing that a total utilization of at least $m/3$ can be achieved if the individual task utilizations do not exceed 1/3.

Baker[3] derived a sufficient feasibility test for $m$-processor preeemptive fixed-priority scheduling of sets of sporadic (or periodic) tasks with arbitrary deadlines. A task set is schedulable if, for each task $\tau_k$, $k = m+1, \ldots, N$, there exists a positive value $\mu \leq m(1 - \frac{c_k}{\min\{d_k, T_k\}})$ such that

$$\sum_{i=1}^{k-1} \beta_{\mu,k}(i) \leq \mu \tag{1}$$

where $\lambda = \frac{m-\mu}{m-1}$ and $\beta_{\mu,k}(i)$ is as defined in the table below:

| Case | $\beta_{\mu,k}(i)$ |
|---|---|
| $\lambda \geq \frac{c_i}{T_i}$ | $\frac{c_i}{T_i}(1 + \frac{T_i - c_i}{d_k})$ |
| $\lambda < \frac{c_i}{T_i}$ | $\frac{c_i}{T_i}(1 + \frac{T_i - c_i}{d_k}) + \frac{d_i}{d_k}(\frac{c_i}{T_i} - \lambda)$ |

**EDF** Goossens, Funk, and Baruah [10] showed that a system of independent periodic tasks can be scheduled successfully on $m$ processors by EDF scheduling if the total utilization is at most $m(1 - u_{\max}) + u_{\max}$, where $u_{\max}$ is the maximum utilization of any individual task. They also showed that this utilization bound is tight, in the sense that there is no utilization bound $\hat{U} > m(1 - u_{\max}) + u_{\max} + \epsilon$, where $\epsilon > 0$, for which $U \leq \hat{U}$ guarantees EDF schedulability. Srinivasan and Baruah[19] also examined the global EDF scheduling of periodic tasks on multiprocessors, and showed that any system of independent periodic tasks for which the utilization of every individual task is at most $m/(2m-1)$ can be scheduled successfully on $m$ processors if the total utilization is at most $m^2/(2m-1)$.

Baker[3] derived a sufficient feasibility test for $m$-processor preemptive EDF scheduling of sets of sporadic (or periodic) tasks with arbitrary deadlines. A task set is schedulable if, for each task $\tau_k$, $k = m+1, \ldots, N$, there exists a positive value $\mu \leq m(1 - \frac{c_k}{\min\{d_k, T_k\}}) + \frac{c_k}{\min\{d_k, T_k\}}$ such that

$$\sum_{i=1}^{N} \beta_k(i) \leq \mu \tag{2}$$

where $\lambda = \frac{m-\mu}{m-1}$ and $\beta_k(i)$ is as defined in the table below.

| | $\frac{c_i}{T_i} \leq \lambda$ | $\frac{c_i}{T_i} > \lambda$ |
|---|---|---|
| $d_i \leq T_i$ | $\frac{c_i}{T_i}(1 + \frac{T_i - d_i}{d_k})$ | $\frac{c_i}{T_i}(1 + \frac{T_i}{d_k}) - \lambda\frac{d_i}{d_k}$ |
| $d_i > T_i$ | $\frac{c_i}{T_i}$ | $\frac{c_i}{T_i}(1 + \frac{T_i}{d_k})$ |

 

The the $m$-processor utilization bounds cited above for arbitrary sets of periodic tasks with deadline equal to period are summarized in the following table:

| | Partitioned | Global |
|---|---|---|
| RM | $(m\lfloor 1/\log_2(u_{\max}+1)\rfloor+1)(2^{1/(\lfloor 1/\log_2(u_{\max}+1)\rfloor+1)}-1)$ | $\frac{m}{2}(1-u_{\max})+u_{\min}$ |
| EDF | $\frac{m\lfloor 1/u_{\max}\rfloor+1}{\lfloor 1/u_{\max}\rfloor+1}$ | $m(1-u_{\max})+u_{\max}$ |

The exact value of the worst-case guaranteed achievable utilization for global RM scheduling still seems to be unknown but it is not smaller than the value in the table above and not be greater than $u_{\max}+m\ln(\frac{2}{1+u_{\max}})$[3].

**Hybrids** In 2002, Srinivasan and Baruah[19] and Andersson, Baruah, and Jonsson[1] showed how to relax the restriction on the largest individual task utilization in the RM and EDF utilization-bound tests. They proposed scheduling policies that give highest priority to tasks of utilization greater than some constant $\zeta$ and schedule the other tasks according to the basic RM or EDF algorithm. These algorithms are called RM-US[$\zeta$] and EDF-US[$\zeta$], respectively. Algorithm EDF-US[$m/(2m-2)$] was shown to correctly schedule on $m$ processors any periodic task system with total utilization $U \leq m^2/(2m-2)$, and RM-US[$m/(2m-2)$] was shown to correctly schedule on $m$ processors any periodic task system with total utilization $U \leq m^2/(3m-2)$.

Lundberg[16] has argued that the optimal value for $\zeta$ in RM-US[$\zeta$] for maximizing the worst-case guaranteed schedulable utilization is $\zeta = 0.37482$, for which the corresponding utilization bound is $0.37482(m+1)$.

It follows from Baker[3] that the optimal value of $\zeta$ for EDF-US[$\zeta$] is $\zeta = 1/2$, for which the utilization bound is $(m+1)/2$. It follows from an argument in [1] that this bound is tight.

The utilization bounds cited above for these hybrid global scheduling algorithms are compared against those for partitioned scheduling in the following table:

| | Partitioned | Global | |
|---|---|---|---|
| RM | $(m+1)(2^{1/2}-1)$ | $(m+1)0.37482$ | RM-US[0.37482] |
| EDF | $(m+1)/2$ | $(m+1)/2$ | EDF-US[0.5] |

It is plain that the worst-case performances of global scheduling with EDF-US[1/2] and FFD partitioning with local RM scheduling are identical and tight. The known bounds for RM-US[0.37482] and partitioned RM scheduling are both lower than with EDF, but are still interesting because of the prevalence of support for fixed-priority scheduling and the dearth of support for deadline scheduling in most of today's operating systems. Of those two, the advantage in worst-case schedulable utilization (when deadline equals period) seems to go do the partitioned approach.

# 3 Empirical Comparisons

In order to better understand the differences between partitioned and global scheduling, a series of experiments were conducted on pseudo-randomly generated sets of periodic tasks. Collections of task sets were generated according to several different models. For each task set in each collection, schedulability was tested using the best available sufficient tests for several global scheduling algorithms, and for first-fit partitioned scheduling according to various ordering heuristics and local scheduling algorithms.

The intent of the experiments was to identify the best performing combinations of global scheduling algorithm and schedulability test, and compare their performance against that of partitioned scheduling. If the performance

 

of any global scheduling algorithm turned out to be better than the results of first-fit partitioning (FFD), other partitioning schemes would have been considered. However, it turned out then FFD partitioning significantly outperformed the global algorithms.

## 3.1 Methodology

Both deadline-based and fixed-priority scheduling were considered. Even though more task sets can be scheduled using deadline scheduling, there are some cases where fixed priority scheduling is more effective, and it remains true that deadline-based does not have the universal support that is found for fixed-priority scheduling among today's off-the-the-shelf operating systems.

The procedure for generating task sets was as follows. An initial set of $m + 1$ tasks was pseudo-randomly generated, and all the schedulability tests were applied to that set. Then another pseudo-randomly generated task was added to the previous set, and all the schedulability tests were run on the new set. This process of adding tasks was repeated until the total processor utilization exceeded $m$. The whole procedure was then repeated, starting with a new initial set of $m + 1$ tasks.

This method of generating random task sets produces a fairly uniform distribution of total utilizations, except at the two extremes. Figure 1 shows histograms of the distribution of per-task-set total utilization three collections of task sets, with total utlizations in the range 0-200%, 0-400%, and 0-800%. The histograms show the number of task sets with total utilization in each of 1000 equal-sized intervals over the three respective ranges of utilizations. The distributions are sparse at the low end because all the task sets are of size greater than $m$. They are also sparse at the very highest end, because all time values are represented as integers.

In all the experiments the task periods were chosen randomly with a uniform distribution between 1 and 1000. The compute times were chosen according to different rules for different experiments, including the following distributions, truncated to bound the utilization between 0.001 and 0.999, including:

1. inverse exponential distribution
2. uniform distribution

The deadlines were chosen in different ways for different experiments, including the following:

1. period: deadline equals period
2. tight: deadline equals computation time
3. double: deadline equals twice the period
4. random: deadlines uniformly distributed between zero and twice the period
5. mixed: deadlines chosen with equal probability from the four cases above

Only the results of a few of the experiments are reported here, due to space limitations and because the trends across the experiments were quite similar. Specifically, histograms are provided for the results of experiments on three collections of 1,000,000 task sets with total utilization in the ranges 0-200%, 0-400%, and 0-800% (for scheduling on 2, 4, and 8 processors, respectively) whose utilizations distributions are shown in Figure 1. For these three collections the compute times were generated to an approximation of the inverse exponential distribution with mean individual task utilization 0.25.

 

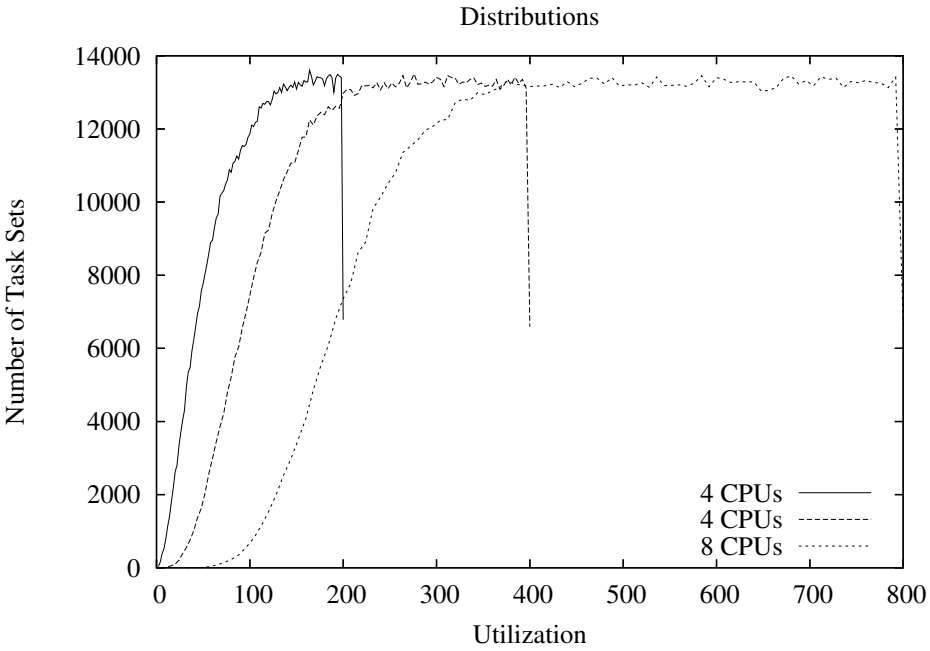


Figure 1: Distribution of total utilizations.

## 3.2 Champion Global Fixed Priority Scheme

**Basic Priority Rules** The goal of the first group of experiments was to identify a "champion" priority assignment rule to represent fixed-priority global scheduling. Since there are no known efficient (less than exponential time) necessary and sufficient schedulability test for global fixed-priority scheduling, it is not practical to determine absolutely whether each task set is schedulable with a given priority assignment. Instead, one is interested in which task sets are *verifiably* schedulable using one of the available sufficient tests. At this time, the most precise sufficient test for global fixed-priority schedulability seems to be Baker's fixed-priority test [3]. The following basic priority assignment rules were tested using the FP Test:

1. period ($T_i$) – rate monotonic scheduling
2. relative deadline ($d_i$) – deadline monotonic scheduling
3. utilization ($\frac{c_i}{T_i}$)
4. density ($\frac{c_i}{\min\{d_i,T_i\}}$)
5. random

An example histogram of the results of one experiment on 1,000,000 task sets, for 4 CPUs, with random deadlines, and exponentially distributed execution times with mean individual task utilization 0.25, is shown in Figure 2. The histogram shows the fraction of the tasks that are verifiably schedulable, for each range of total task-set utilizations. In this experiment the priority assignment rule that most often results in a verifiably schedulable system is deadline-monotonic. In other experiments the shapes of the histograms varied, and in some cases there was little or no statistical difference in performance between priority assignment rules, but in all cases deadline monotonic priority assignment was the statistical winner.

 

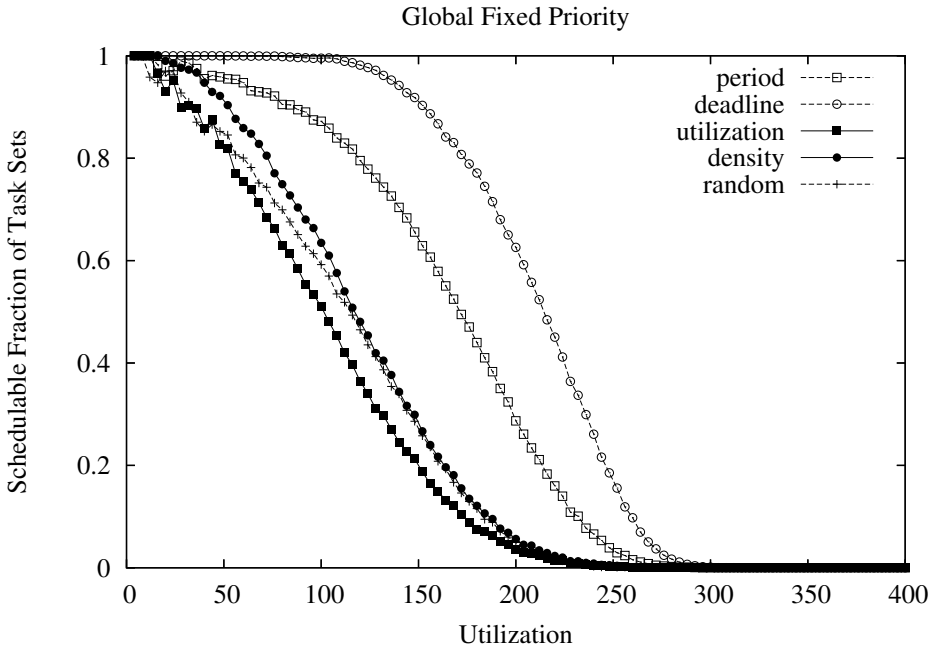


Figure 2: Comparison of basic fixed priority rules.

**Hybrids** In addition to the straight application of the deadline-monotonic rule, several hybrid rules generalizing the RM-US concept were considered. That is, up to $m$ tasks were selected to receive special (that is, highest) priority, and the remaining tasks were scheduled according to the deadline-monotonic priority rule. The following rules for choosing the special tasks were evaluated:

1. DM-US[1/3]: the tasks of utilization greater than 1/3;
2. DM-US[L]: the tasks of utilization greater than 0.3748225282;
3. DM-U: the $k$ tasks with highest utilization, where $k$ is the smallest value between 0 and $m$ for which the system can be verified schedulable according to the FP Test.
4. DM-D: the $k$ tasks with highest density, where $k$ is the smallest value between 0 and $m$ for which the system can be verified schedulable according to the FP Test.

Figure 3 compares the performances of the DM and hybrid priority schemes on the same collection of task sets as Figure 2. The DM-D hybrid scheme results in the highest number of verifiably schedulable task sets at every total utilization level. Although the differences were less in some experiments, this scheme always was the best statistical performer.

## 3.3 Champion Global EDF Scheme

The next step was to choose a champion for global EDF scheduling. The schedulability test chosen was Baker's EDF test[3], which seems to be the most precise sufficient test for global EDF schedulability known at this time. In addition to basic EDF scheduling, the following hybrids of EDF and highest-utilization-first scheduling were considered, with both homogeneous and split applications:

 

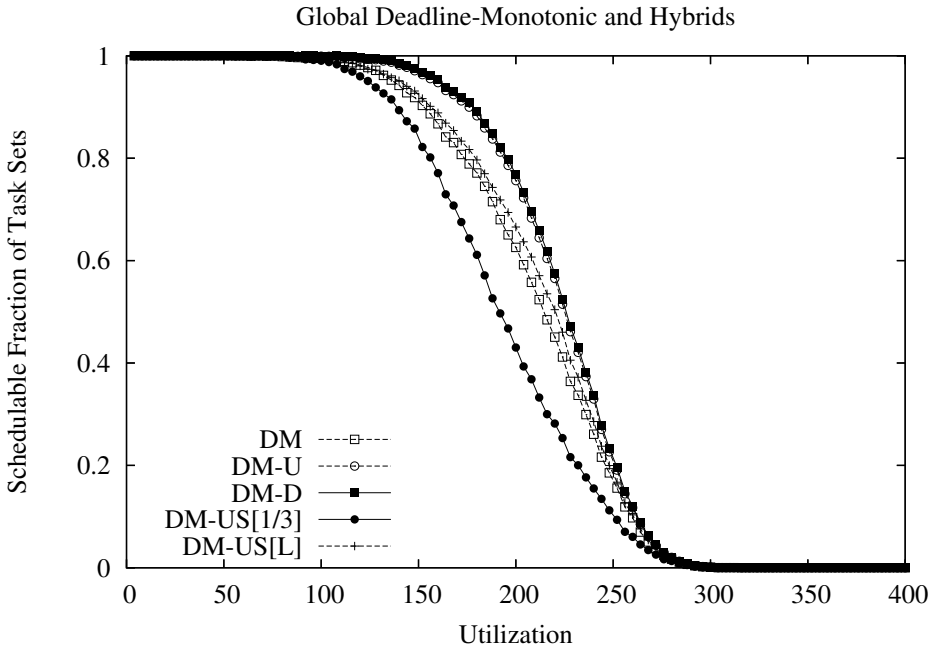


Figure 3: Comparison of global deadline monotonic and hybrid priority schemes.

1. EDF-US[1/2]: give special priority to the tasks of utilization greater than 1/2, which is the cut-off value that guarantees the highest worst-case utilization when deadline=period;
2. EDF-U: give special priority to the $k$ tasks with highest utilization, where $k$ is the smallest value between 0 and $m$ for which the system can be verified as schedulable according to the EDF Test.
3. EDF-D: give special priority to the $k$ tasks with highest density, where $k$ is the smallest value between 0 and $m$ for which the system can be verified as schedulable according to the EDF Test.

Figure 4 compares the performances of the EDF and hybrid schemes on the same collection of task sets as the other figures above. It is clear that the EDF-D hybrid scheme results in the highest number of verifiably schedulable task sets at every total utilization level. Although the performance differences were less in some experiments, EDF-D always was the best statistical performer.

## 3.4 Champion Partitioned Scheduling Schemes

To select champions for partitioned scheduling, the same task sets as used in the experiments above were tested with the first-fit decreasing (FFD) partitioning algorithm for both DM and EDF local scheduling. The results of these experiments for several partitioning orders are shown in Figure 5 and Figure 6.

The best performances observed in each class were from the following:

1. RT_FFD: first-fit decreasing partitioned scheduling based on deadline-monotonic priority order and local scheduling according to the same scheme, using the necessary-and-sufficient fixed-priority response-time test[2] to determine local schedulability.

 

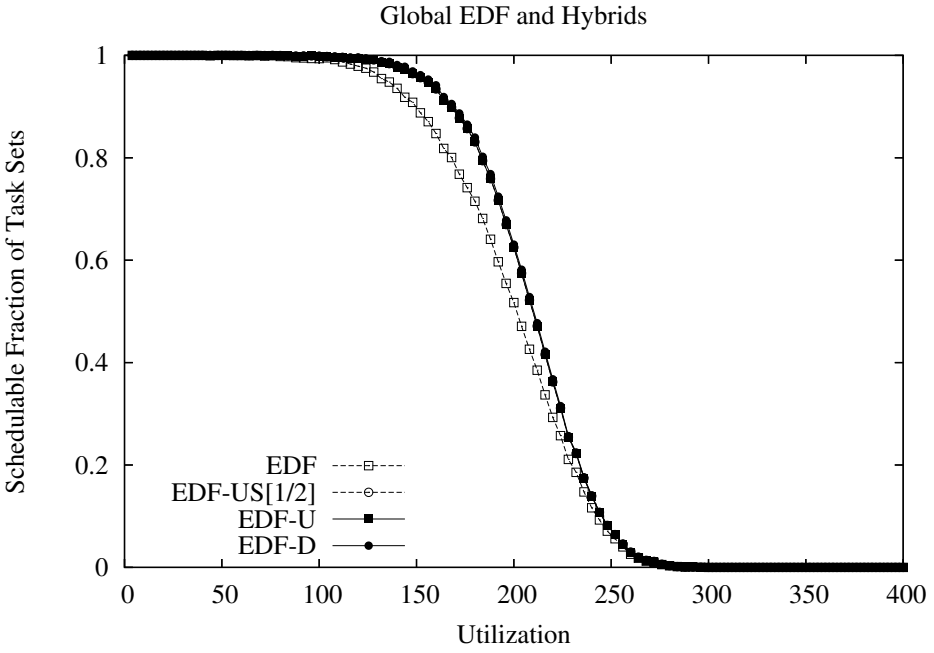


Figure 4: Comparision of hybrid EDF schemes.

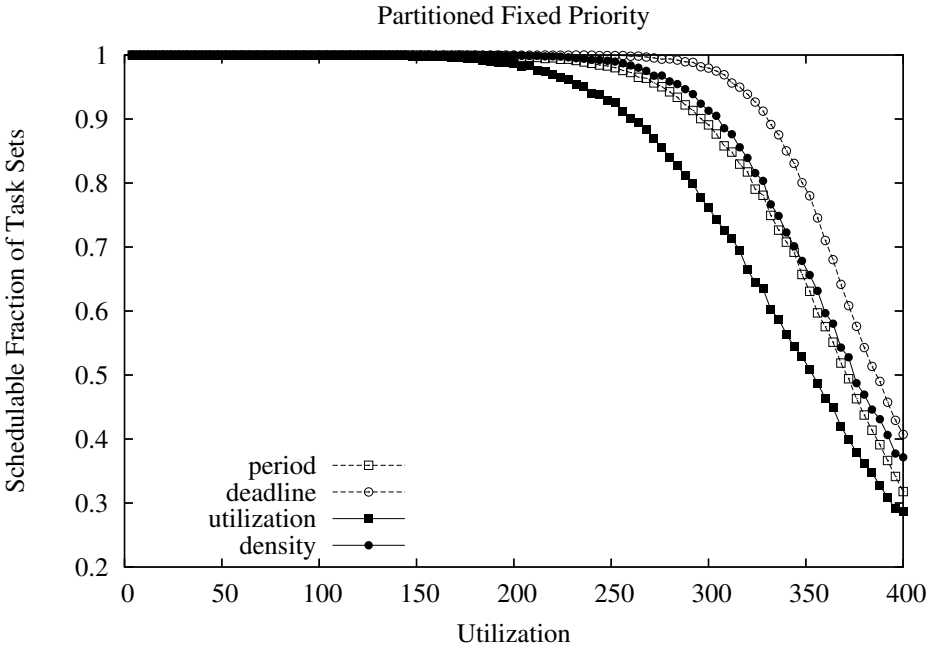


Figure 5: Comparision of partitioning orders for local DM scheduling.

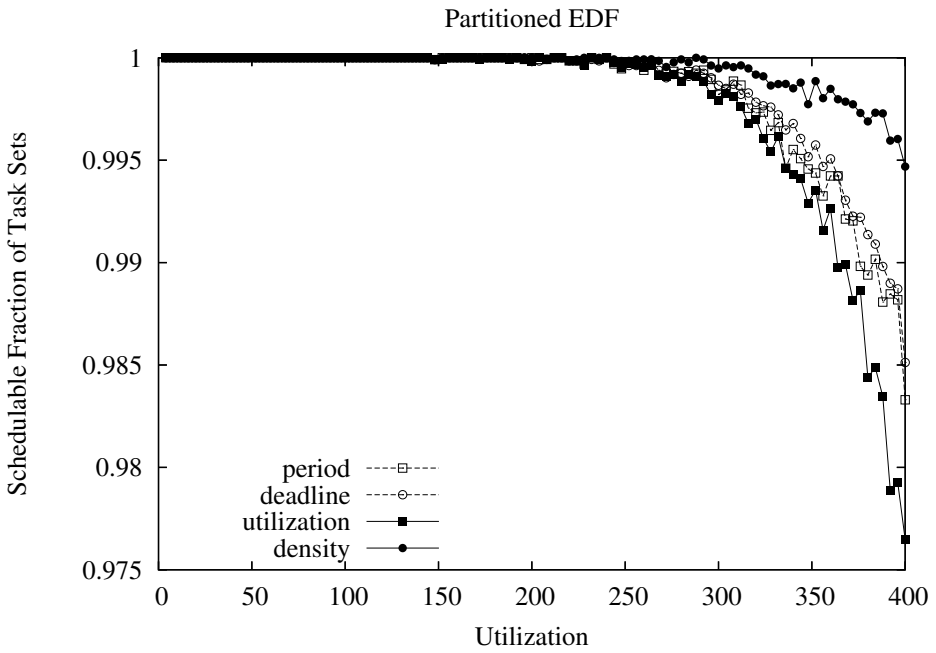


Figure 6: Comparision of partitioning orders for local EDF scheduling.

2. EDF_FFD: first-fit decreasing partitioned scheduling based on deadline-monotonic priority order and local EDF scheduling, using the density test ($\sum_{i=1}^{N} \frac{c_i}{\min\{d_i,T_i\}} \leq m$) as an initial screen and the necessary and sufficient test of Baruah *et al.*[7] to resolve uncertainties[1]

These were chosen as representatives of the partitioned scheduling approach in the competition. They are known to be reasonably good, are easily computed, and have known worst-case utilization bounds for the case when deadline equals period. (No claim is made that they are optimal among all partitioning schemes.)

Figure 7 compares the performances of these two fixed-priority schemes against the above two partitioned scheduling schemes on the same collection of task sets for which the performance of the basic priority schemes is shown in the other figures above. In this experiment, it is clear that the partitioned scheduling schemes result in the highest number of verifiably schedulable task sets at every total utilization level. Although the differences were less in some experiments, this pattern persisted.

The success rate of the global approach seems quite low in comparision to the partitioned approach. Of course, this could just be due to over-conservatism in Baker's global schedulability test. To see how much room for improvement there might be, execution of the same task sets was simulated, for just the case where all tasks are released together at time zero, and just until each task had reached the first point where it was not backlogged. Success in such a limited simulation is not a proof of schedulability, but it is a necessary condition. Certainly any task set that is observed to miss a deadline during such a simulation is not schedulable.

It can be seen in Figure 8 that there is a huge gap between the number of task sets that Baker's sufficient tests[3] are able to verify as schedulable and the number of task sets that *might* be schedulable, based on the limited simulations. This suggests that there is potential for significant improvement in Baker's tests. However, the area of cross-over between the observed actual performance of FFD partitioned scheduling and the optimistic

[1] In the few cases where completion of Baruah's test was not practicable, either because of integer overflow or compute time greater than one second, the task set was assumed to be unschedulable.

 

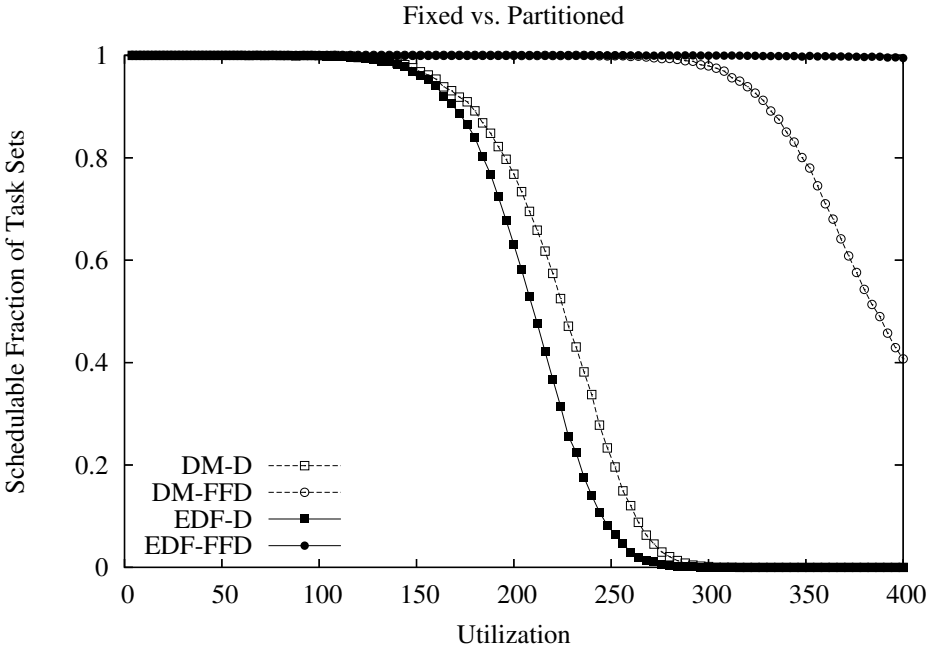


Figure 7: Performance of global vs. partitioned approaches.

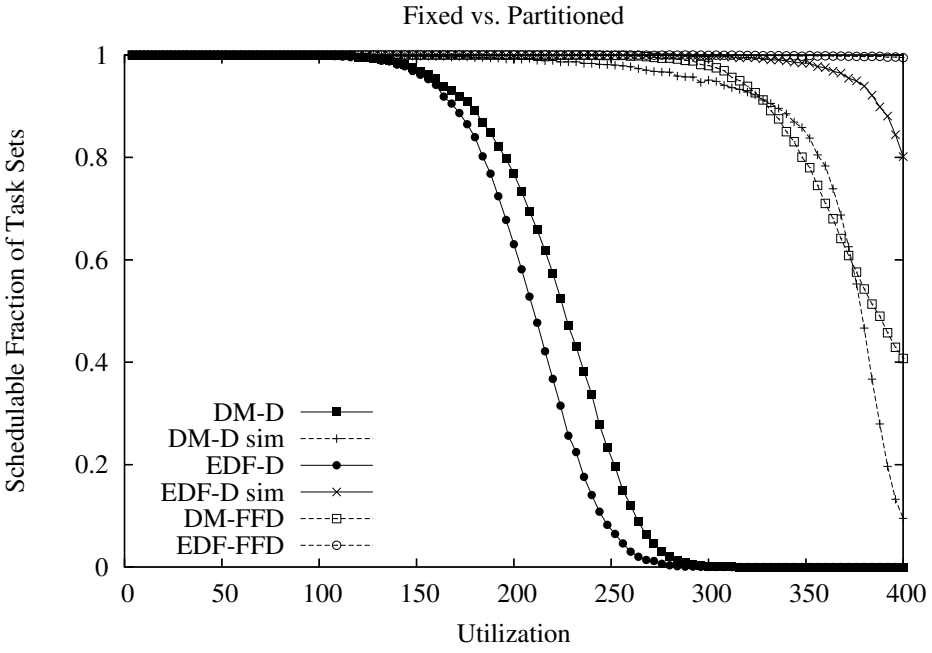


Figure 8: Upper bounds from simulations.

upper bound on the possible performance of global scheduling given by the simulations is non-existent for EDF and very small for DM. Therefore, while Baker's analysis may be overly conservative, improvements in global schedulability tests are not likely to reverse the apparent advantage of partitioned scheduling with regard to the likelihood of scheduling a random task set.

# 4 Conclusions and Future Work

The choice between global and partitioned approaches to multiprocessor scheduling is a conundrum. On the one hand, it has long been known from queueing theory[12] that single-queue (global) FIFO multiprocessor scheduling is superior to queue-per-processor (partitioned) FIFO scheduling, with respect to average response time. On the other hand partitioned scheduling seems to be superior to global scheduling with respect to hard-deadline tasks. All task sets with utilization below $m(2^{1/2} - 1)$ can be scheduled on $m$ processors using a first-fit-decreasing partitioning algorithm and and local rate monotonic scheduling[17], but there are hard-deadline periodic task sets with total utilization arbitrarily close to 1.0 that cannot meet all deadlines if scheduled on $m$ processors using any priority scheduling scheme with fixed job priorities[8]. Still, it is clear that global scheduling is superior on certain task sets, including any task set with more than $m$ tasks with individual task utilization higher than 1/2. Moreover, higher levels of schedulable utilization than 1.0 can be guaranteed, even in the worst case, by using hybrid global scheduling schemes such as EDF-RM and EDF-US[19, 1], and both the partitioned EDF and global EDF-US scheduling approaches appear to achieve the same worst-case level of processor utilization.

The experiments reported here provide some additional evidence on which to base a choice between these two approaches. Statistically, the chance of being able to satisfy all the deadlines of a randomly chosen periodic or sporadic task set appears to be highest with partitioned scheduling. In particular, the partitioned EDF scheduling appears to be the overall best performer in this statistical sense. At the same time, there are specific task sets where global scheduling is more effective. While the schedulability tests used in the experiments probably could be improved, simulations suggest that they cannot be improved enough to erase the advantage of partitioned scheduling.

This is not be the end of global vs. partitioned scheduling question. There are global scheduling schemes that can guarantee schedulability at higher processor utilization levels than the $(m + 1)/2$ worst-case bound for job-static priority scheduling. These include several variants of the PFAIR concept. Baruah, Cohen, Plaxton and Varvel[5, 6] showed that PFAIR scheduling is optimal for scheduling periodic tasks on a multiprocessor, has a linear-time necessary and sufficient schedulability test, and for sufficiently small quantum size can gurantee schedulability at processor utilization levels arbitrarily close to $m$. Srinivasan and Anderson showed that the PFAIR approach is also optimal for multiprocessor scheduling of sporadic and rate-based tasks[18], and there have been many more variations and extensions to the PFAIR theory made since that. The main problem with PFAIR scheduling, is the need to slice time into small quanta, and the consequently high implementation overhead. In this regard, the fixed-job-priority algorithms, like those considered in this paper have an advantage, whether applied globally or partiioned. Is there another algorithm, that can break the $(m + 1)/2$ bound, but does not require such frequent time slicing as the PFAIR approach? One possibility is throwforward, shown by Johnson and Maddison[11] to be optimal for scheduling independent jobs on a multiprocessor system. It will be interesting to see whether their analysis of throwforward scheduling can be extended to provide a sufficient test for schedulability of periodic and sporadic task systems.

There are also some remaining questions about the comparative implementation overhead of the global vs. partitioned approachs. Global scheduling can have higher overhead in at least two respects: the contention delay and the synchronization overhead for a single dispatching queue is higher than for per-processor queues; the cost of resuming a task may be higher if it is on a different processor (due to interprocessor interrupt handling and cache reloading) than on the processor where it last executed. The latter cost can be quite variable, since it depends on the actual portion of a task's memory that remains in cache when the task resumes execution, and how much of that remnant will be referenced again before it is overwritten. These issues are discussed at some

 

length by Srinivasan *et al.* in [20], which includes some simulation results comparing the overhead of global EDF and $PD^2$ scheduling, a PFAIR variant. It seems that only experimentation with actual implementations can make a conclusive case as to how serious are these overheads, and how they balance against any advantages global scheduling may have for on-time completion of tasks in real applications.

## References


[1] B. Andersson, S. Baruah, and J. Jonsson. Static-priority scheduling on multiprocessors. In *Proc. 22nd IEEE Real-Time Systems Symposium*, pages 193–202, London, UK, December 2001.

[2] N. C. Audsley, A. Burns, M. Richardson, and A. J. Wellings. Hard real-time scheduling: the deadline monotonic approach. In *Proc. 8th IEEE Workshop on Real-Time Operating Systems and Software*, pages 127–132, Atlanta, GA, USA, 1991.

[3] T. P. Baker. Multiprocessor EDF and deadline monotonic schedulability analysis. In *Proc. 24th IEEE Real-Time Systems Symposium*, pages 120–129, 2003.

[4] S. Baruah and Joel Goossens. Rate-monotonic scheduling on uniform multiprocessors. *IEEE Trans. Computers*, 52(7):966–970, July 2003.

[5] S. K. Baruah, N. Cohen, C. G. Plaxton, and D. Varvel. Proportionate progress: a notion of fairness in resource allocation. In *Proc. ACM Symposium on the Theory of Computing*, pages 345–354, May 1993.

[6] S. K. Baruah, N. Cohen, C. G. Plaxton, and D. Varvel. Proportionate progress: a notion of fairness in resource allocation. *Algorithmica*, 15:600–625, 1996.

[7] S. K. Baruah, R. R. Howell, and L. E. Rosier. Algorithms and complexity concerning the preemptive scheduling of periodic real-time tasks on one processor. *Real-Time Systems*, 2, 1990.

[8] S. K. Dhall and C. L. Liu. On a real-time scheduling problem. *Operations Research*, 26(1):127–140, February 1978.

[9] S. Funk, J. Goossens, and S. Baruah. On-line scheduling on uniform multiprocessors. In *Proc. 22nd IEEE Real-Time Systems Symposium*, pages 183–192, London, UK, December 2001. IEEE Computer Society.

[10] J. Goossens and R. Devillers. Feasibility intervals for the deadline driven scheduler with arbitrary deadlines. In *Proc. 6th International Conf. Real-Time Computing Systems and Applications (RTCSA'99)*, 1999.

[11] H. H. Johnson and M. S. Maddison. Deadline scheduling for a real-time multiprocessor. In *Proc. Eurocomp Conference*, pages 139–153, 1974.

[12] L. Kleinrock. *Queueing Systems - Volume 2: Computer Applications*. Wiley Interscience, 1976.

[13] C. Lee, J. Lehoczky, D. Siewiorek, R. Rajkumar, and J. Hansen. A scalable solution to the multi-resource QoS problem. In *Proc. of IEEE Real-Time Systems Symposium*, Phoenix, AZ, USA, December 1999.

[14] J. M. Lopez, J. L. Diaz, and D. F. Garcia. Minimum and maximum utilization bounds for multiprocessor RM scheduling. In *Proc. 13th Euromicro Conf. Real-Time Systems*, pages 67–75, Delft, Netherlands, June 2001.

[15] J. M. Lopez, J. L. Diaz, M. Garcia, and D. F. Garcia. Worst-case utilization bound for EDF scheduling on real-time multiprocessor systems. In *Proc. 12th Euromicro Conf. Real-Time Systems*, pages 25–33, 2000.

[16] L. Lundberg. Analyzing fixed-priority global multiprocessor scheduling. In *Proc. 8th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 145–153, San Jose, CA, USA, 2002. IEEE Computer Society.


 

[17] D. I. Oh and T. P. Baker. Utilization bounds for $N$-processor rate monotone scheduling with stable processor assignment. *Real Time Systems*, 15(2):183–193, September 1998.

[18] A. Srinivasan and J. Anderson. Optimal rate-based scheduling on multiprocessors. In *Proc. 34th ACM Symposium on Theory of Computing*, pages 189–198. ACM, May 2002.

[19] A. Srinivasan and S. Baruah. Deadline-based scheduling of periodic task systems on multiprocessors. *Information Processing Letters*, 84:93–98, 2002.

[20] A. Srinivasan, P. Holman, J. H. Anderson, and S. Baruah. The case for fair multiprocessor scheduling. In *Proc. 11th International Workshop on Parallel and Distributed Real-time Systems*, April 2003. Available from: `http://www.cs.unc.edu/$\tilde$anderson/papers/icdcs03a.ps`.

 