PRELIMINARY EVALUATION OF RUN AGAINST S-PD^2, P-EDF AND G-EDF

We report here the raw results of comparing our implementation of RUN against three multiprocessor algorithms: S-PD^2, P-EDF and G-EDF, already implemented in LITMUS^{RT}[1] (version 2013.1). In these experiments we were interested in studying the scheduling algorithm with respect to the interference it causes to the system rather than finding its empirical utilization cap. Our main focus was thus set on assessing the kernel overhead in terms of both the cost of its primitives and the number of preemptions/migrations incurred (cumulative and per job).

Experimental setting

The experimental process we followed consisted in collecting execution traces obtained by feeding identical task sets to the four scheduling algorithms and then comparing the respective execution statistics. It stands to reason that the performance of each algorithm depends on the task set features and the overall system workload in particular. To warrant fair evaluation, we focused on randomly generated task sets with increasing overall system utilization, between 50% and 100%. The granularity of our observations increases while we approach higher system utilization as we expect most important variations to occur then. When it comes to the task population, we wanted our task sets to be equally representative of light and heavy tasks. To this end, task utilizations were generated following a bimodal distribution over the two intervals [0.001,0.5) and [0.5, 0.9], with probabilities respectively equal to 45% and 55%. We are also aware that the performance of some algorithms may be affected by the distribution of task periods, which in turn affects the distribution of release events. For this reason, we performed two sets of experiments targeting harmonic and non-harmonic task sets. In both cases, tasks periods were drawn from a uniform distribution in [10ms; 200ms] so as to keep the experiment manageable in size and complexity. We used the lightweight tracing facilities offered by LITMUS^{RT} to collect execution traces for the three contenders, where each task set was executed for 30 seconds spanning multiple hyperperiods.

Experiments were run on a dual-processor AMD Opteron^{TM} 2356 system where each processor includes 4 cores running at 2.3 GHz with private L1 and L2 caches of 64 and 512 KB respectively. Power scaling and power management functions were intentionally disabled.

Raw results

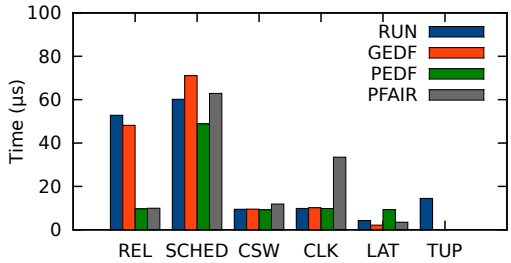
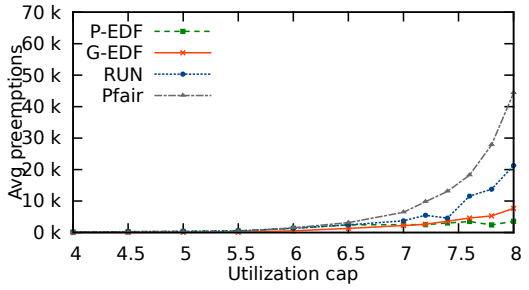
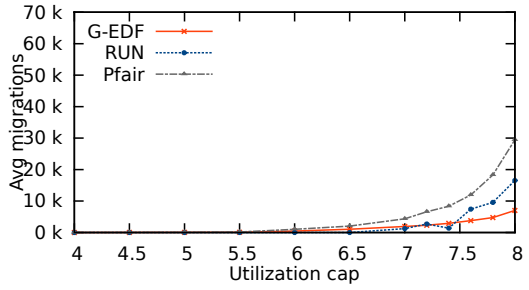


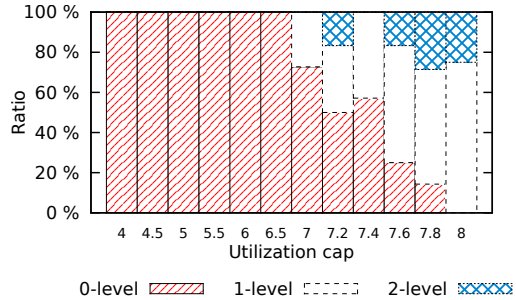
Fig. 1. Maximum observed system overheads. This diagram reports the maximum observed cost of each scheduling primitive: the *release* (REL) of a new job, a *scheduling* (SCHED) event and the induced *context switch* (CSW), the *interrupt latency* (LAT), including inter-processor interrupts (IPI), and the *tick interference* (CLK). RUN also incurs an additional overhead owing to the run-time *update* (TUP) of the reduction tree.



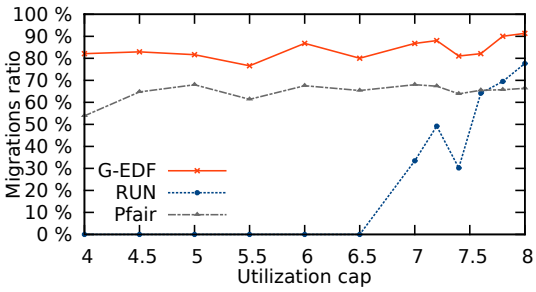
(a) Observed preemptions.



(b) Observed migrations.



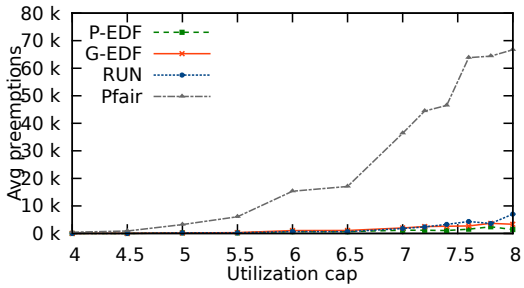
(c) Tree levels per utilization.



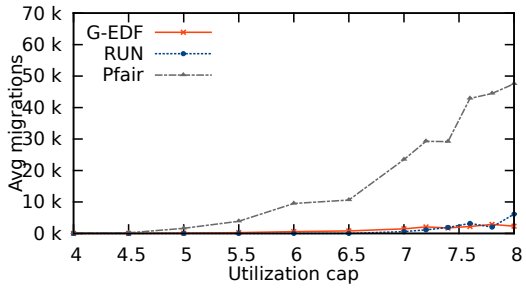
(d) Migration ratio.

Fig. 2. Kernel interference for NON-HARMONIC task sets in terms of the number of incurred preemptions and migrations. Diagram (c) allows to associate RUN behaviour with the height of its reduction tree.

[1] http://www.litmus-rt.org

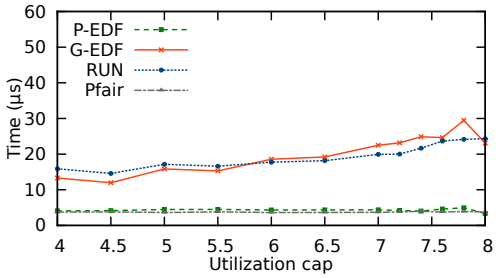


(a) Observed preemptions.

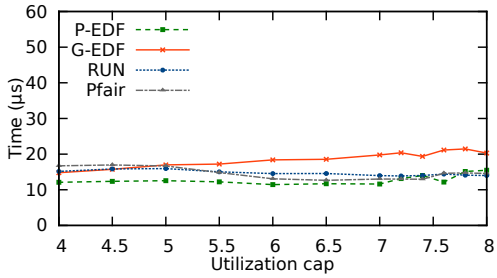


(b) Observed migrations.

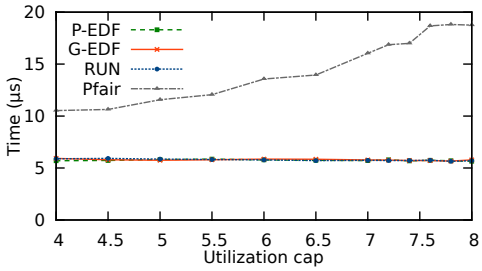
Fig. 3. Kernel interference for HARMONIC task sets in terms of the number of incurred preemptions and migrations.



(a) Average release overhead.

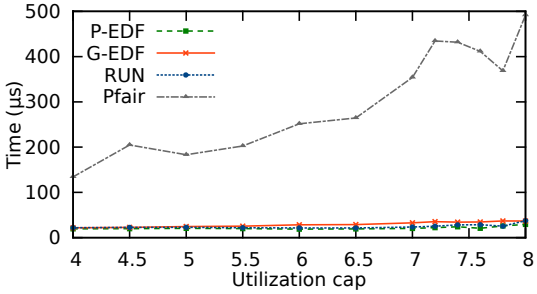


(b) Average schedule overhead.

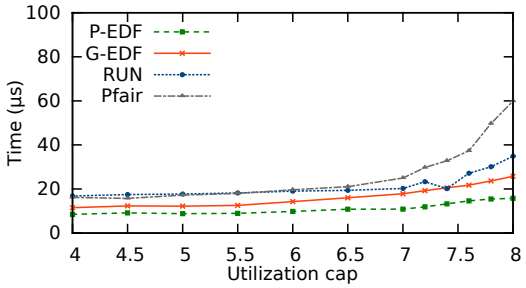


(c) Average tick overhead.

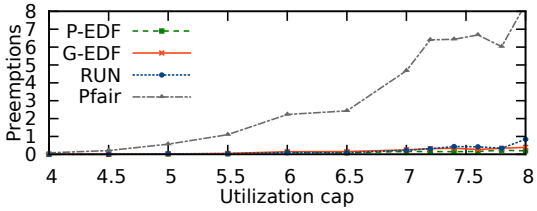
Fig. 4. Average cost of the release, schedule and tick primitives. Note that most part of S-PD^2 logic is attached to the tick handler, which incurs a variable (and non-negligible) tick overhead.



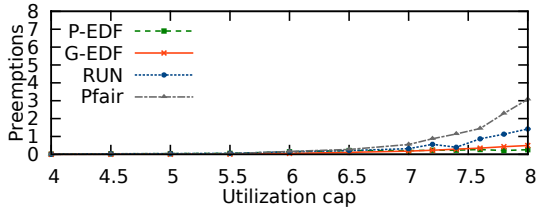
(a) Average overhead per job for harmonic task sets.



(b) Average overhead per job for non-harmonic task sets.



(c) Average preemptions per job for harmonic task sets.



(d) Average preemptions per job for non-harmonic task sets.

Fig. 5. Overall per-job scheduling overheads under the four algorithms, for both HARMONIC and NON-HARMONIC task sets.