1 Overhead Analysis

We consider six external sources of overheads: release overhead, tick overhead, scheduling overhead, context-switch overhead, preemption overhead, and migration overhead.

Release overhead, denoted H_r, refers to the overheads incurred when releasing a job.

Tick overhead, denoted H_t, refers to the overheads incurred by a scheduling algorithm in performing housekeeping functions but not effecting a change in the set of executing jobs at any boundary.

Scheduling overhead, denoted H_s, refers to the time taken in effecting a change to the set of jobs executing on a processor.

Context-switch overhead, denoted H_c, refers to the actual cost of switching between two tasks and does not include any task-specific cache-related overheads.

Preemption overhead, denoted H_p, refers to the delay associated with servicing the cache misses suffered by a preempted job when it resumes execution on the same cpu.

Migration overhead, denoted H_m, refers to the delay associated with servicing the cache misses suffered by a preempted job when it resumes execution on a different cpu.

For more detailed description of these overheads, please take a look at Bjorn, Uma, and John's related papers.

2 Algorithm EDF-fm

Let $T.e$ denote T's original worst case execution time and $T.d$ denote its deadline.

Release Overhead Each job of any task T should be charged by one release overhead since each job is released exactly once under EDF-fm.

Scheduling and Context-Switch Overheads Since preemption may happen under EDF-fm, additional scheduling and context-switch, and preemption overheads should be considered. When a job J of a task T preempts another job, it suffices to charge J with the scheduling and context-switch overheads associated with resuming the preempted lower-priority jobs, apart from charging J with one scheduling

and context-switching cost for its own execution.

Preemption and Migration Overheads If T is fixed, then as argued in Uma's dissertation, it can be shown that if a job of a task U resumes immediately after a job of T completes, then $U.d > T.d$ holds. Thus, only tasks with relative deadlines larger than that of T need to be considered in dertermining the amount of preemption overhead to charge.

If T is migrating, then given that under EDF-fm T has a higher priority than all fixed tasks assigned to the same processor, all these tasks need to be considered in determining the preemption overhead to charge to T. Note that under EDF-fm a job does not really migrate during its lifetime.

Tick Overhead After considering these overheads, we need to inflate the execution by the tick overhead. The following accounting method is stated in Uma's dissertation. The tick overhead is potentially incurred at the beginning of every quantum. That is, effectively at most $Q - H_t(fm)$ (where Q is the quantum size) time units are guaranteed to be available within each quantum for executing any job, including the time needed for handling other overheads. Hence, even if a job is allocated an entire quantum, the amount of time that the job actually executes for within that quantum can be lower by $H_t(fm)$. On the other hand, if a job either commences or resumes execution in the middle of a quantum, then the tick overhead is not incurred by the job in that quantum, which is only partially allocated to it. Given that we assume a job is not preempted in the middle of a quantum, there is at most one quantum, namely, the quantum in which the job completes execution, that is partially allocated to it in which it incurs tick overhead. Let the worst case execution time of T be $T.e$ quanta, that is, $T.e \cdot Q$ time units. Then the number of quanta spanned by any job of T after accounting for the tick overhead is given by $\frac{T.e \cdot Q}{Q - H_t(fm)}$. Since the number of quanta that are fully allocated to each job of T is at most $\lfloor \frac{T.e \cdot Q}{Q - H_t(fm)} \rfloor$ and each job may incur tick overhead in at most one quantum that is partially allocated to it, the number of tick costs to charge for each job of T is at most $\lceil \frac{T.e \cdot Q}{Q - H_t(fm)} \rceil$. Hence, the overhead due to tick scheduling is at most $\lceil \frac{T.e \cdot Q}{Q - H_t(fm)} \rceil \cdot H_t(fm)$ time units, which is $\lceil \frac{T.e \cdot Q}{Q - H_t(fm)} \rceil \cdot \frac{H_t(fm)}{Q}$ quanta.

Followed from the above discussion, overhead under EDF-fm can be fully accounted for if the worst case execution time of each task T is determined as follows.

$$T.e = \begin{cases} T.e + H_r(fm) + 2 \cdot H_s(fm) + 2 \cdot H_c(fm) + max_{\{U|U is fixed and U.D > T.D\}}(H_p(fm, U)) \text{ if } T \text{ is fixed} \\ T.e + H_r(fm) + +2 \cdot H_S(fm) + 2 \cdot H_c(fm) + max_{\{U|U \in \tau\}}(H_p(fm, U)) \text{ if } T \text{ is migrating} \end{cases}$$

Finally, we need to inflate the execution by the tick overhead, as discussed above.

3 Algorithm EDF-wm

The overhead analysis of EDF-wm differs from EDF-fm mainly because under EDF-wm, a task can migrate over multiple processors (up to m) and a migratory job is sliced into multiple windows. Let $T.y$ denote the number of processors on which T may execute. EDF-wm incurrs more release overheads than EDF-fm because each job is sliced into multiple subjobs that are released and executed on different processors. It also incurrs more scheduling , context-switch, migration, and preemption overheads due to the fact that a migratory task executes on multiple processors. Thus, each migratory job may preempt more than one lower-priority jobs on multiple processors.

Release Overhead If T is fixed, then each job of T should be charged by one release overhead since each job is released exactly once.

If T is migrating, then it is neccessary to charge each job of T $T.y \cdot H_r(wm)$ amount of release overheads since each of its job may be released $T.y$ time on $T.y$ different processors (under EDF-wm a migrating job is sliced into $T.y$ subjobs that are released on different processors).

Scheduling and Context-Switch Overheads If T is fixed, then it suffices to charge T with the scheduling and context-switch overheads associated with resuming the preempted lower-priority jobs, apart from charging T with one scheduling and context-switching cost for its own execution.

If T is migrating, given that each of T's job may migrate on $T.y$ processors, it suffices to charge T $2T.y \cdot (H_s(wm) + H_c(wm))$ amount of scheduling and context-switch overheads.

Preemption and Migration Overheads If T is fixed, then accounting preemption overhead is same as for a fixed task under EDF-fm, except that all jobs need to be considered in determining the preemption overhead to charge to T because both fixed and migratory jobs are prioritized according to EDF (while in EDF-fm a migratory task alwasys has higher static priority than other fixed tasks residing on the same processor).

If T is migrating, it is clear that $T.y \cdot max_{\{U|U \neq T \in \tau\}}(H_m(wm, U))$ amount of migration overheads should be charged to T. Since the worst-case migraion overhead is no worse than the worst-case preemption overhead, we do not need to inflate preemption overhead again.

Tick Overhead Same as under EDF-fm.

Followed from the above discussion, overhead under EDF-wm can be fully accounted for if the worst case execution time of each task T is determined as follows.

$$T.e = \begin{cases} T.e + H_r(wm) + +2 \cdot H_s(wm) + 2 \cdot H_c(wm) + max_{\{U|U \in \tau\}}(H_p(wm, U))) \text{ if } T \text{ is fixed} \\ T.e + T.y \cdot H_r(wm) + 2T.y \cdot H_S(wm) + 2T.y \cdot H_c(wm) + T.y \cdot max_{\{U|U \neq T \in \tau\}}(H_m(wm, U)) \text{ if } T \text{ is migr} \end{cases}$$

4 Algorithm EDF-ss

Accouting for overhead under EDF-ss is similar to EDF-fm, but its preemption/migration/scheduling/cs overheads largely depend on the number of slots it needs to fully complete the execution of one job.

According to the EDF-ss paper, a job of any migratory task T may execute on two processors in at most $\lfloor \frac{min(T.d, T.p)}{S} \rfloor$ slots. For conciseness, let $T.s = \lfloor \frac{min(T.d, T.p)}{S} \rfloor$. Within each slot, some computation is executed on one processor and then some computation is executed on another processor which implies that the job migrates between two processors within every slot. Clearly a job of T may migrate up to $2T.s - 1$ times during its lifetime.

Release Overhead Each job of any task T should be charged by one release overhead since each job is released exactly once under EDF-ss.

Scheduling and Context-Switch Overheads If T is fixed, then same as EDF-fm.

If T is migrating, then it suffices to charge T $(2T.s - 1) \cdot (H_s(ss) + H_c(ss))$ amount of scheduling and context-switch overheads since a job of T may migrate between two processors for at most $2T.s - 1$ times.

Preemption and Migration Overheads If T is fixed, then same as EDF-fm.

If T is migrating, then it is clear that $(2T.s - 1) \cdot max_{\{U|U \neq T \in \tau\}}(H_m(ss, U))$ amount of migration overheads should be charged to T.

Tick Overhead Same as under EDF-fm.

Followed from the above discussion, overhead under EDF-ss can be fully accounted for if the worst case execution time of each task T is determined as follows.

$$T.e = \begin{cases} T.e + H_r(fm) + 2 \cdot H_s(fm) + 2 \cdot H_c(fm) + max_{\{U|U is fixed and U.D > T.D\}}(H_p(fm, U)) \text{ if } T \text{ is fixed} \\ T.e + H_r(ss) + (2T.s - 1) \cdot H_S(ss) + (2T.s - 1) \cdot H_c(ss) + (2T.s - 1) \cdot max_{\{U|U \neq T \in \tau\}}(H_m(ss, U)) \text{ if } T \end{cases}$$