Semi-partitioned Model on Dual-core Mixed Criticality System

Hao Xu, Alan Burns
Background

- Mixed Criticality System
  - Task has different Worst Case Execution Time (C) for different criticality levels (L)
  - For the same task:
    \[ L_m > L_n \implies C_m > C_n \]
- Two criticality levels (HI, LO)
  - HI > LO
Background

- **Vestal’s Approach**\(^1\)
  - All tasks keeps schedulable
  - Pessimistic

- **Runtime monitors**
  - LO-crit task -> prevent from execution
  - HI-crit task -> mode change

- **AMC**\(^2\) and **EDF-VD**\(^3\) etc.

---


Motivation

- Explore a way that all of the tasks remain schedulable throughout the criticality level changes.
- Multi-core platform provides the option of migration
Assumptions

- Independent tasks
- Deadline < period
- Migration cost
- Mode change frequency and isolation
Semi-partitioned Model

- If all tasks execute within their LO-crit budget then all deadlines are met and no tasks migrate.
- No LO-crit task is allowed to exceed its LO-crit budget.
- If HI-crit tasks on one core exceed their LO-crit budget, then some LO-crit tasks will migrate, but ALL LO-crit tasks and HI-crit tasks remain schedulable.
- If HI-crit tasks on more than one core exceed their LO-crit budget, then some LO-crit tasks will be abandoned, but all HI-crit tasks remain schedulable (without migration).
State View

- Task set $S$ to represent the collection of all tasks sets:
  - $S = (LO_1 \cup LO_2) \cup (HI_1 \cup HI_2) \cup (MIG_1 \cup MIG_2)$

- State $X$ to represent the normal execution state:
  - $X_1 = LO_1 \cup HI_1 \cup MIG_1$
  - $X_2 = LO_2 \cup HI_2 \cup MIG_2$
  - $S = X_1 \cup X_2$
One core enters HI-crit Mode

- State $Y(1)$ to represent the state that Core $c_1$ enters HI-crit mode.
  - $Y(1)_1 = LO_1 U HI_1$
  - $Y(1)_2 = LO_2 U HI_2 U MIG_1 U MIG_2$
  - $S = Y(1)_1 U Y(1)_2$

- Release Jitter and Reduced deadline issues for migrating tasks ($MIG_1$)
Release Jitter Issue

- Task released at time $t_0$ and migrates at time $t$

\[
\begin{align*}
\text{Release:} & \quad t_0 \\
\text{Migrate:} & \quad t_0 + T \\
\text{Delay:} & \quad t_0 + D \\
\text{Total:} & \quad t_0 + R
\end{align*}
\]
Reduced Deadline Issue

\[ d_i'_{\text{max}} = d_i - (R_i(LO) - C_i) \]
Both Cores in HI-crit Mode (1)

- $BY(1)$ to represent the case that core $c_2$ also enters HI-crit mode from state $Y(1)$
  - $BY(1)_1 = LO_1 \cup HI_1$
  - $BY(1)_2 = HI_2$
  - $S = BY(1)_1 \cup BY(1)_2 \cup LO_2 \cup MIG_1 \cup MIG_2$
Both Cores in HI-crit Mode (2)

- $BX$ to represent the case that both cores enter HI-crit mode at the same time.
  - $BX_1 = LO_1 \cup HI_1$
  - $BX_2 = LO_2 \cup HI_2$
  - $S = BX_1 \cup BX_2 \cup MIG_1 \cup MIG_2$
Allocation

- Bin packing algorithms
  - Criticality aware utilisation decreasing
g
- Non-migration
- Migration
Choice of Migration

Regarding to the equation:

- Migrating LO-crit tasks with largest slack time

\[
\forall \tau_i \in (Y(1) \cup Y(2)) : \\
R_i(HI)^* = C_i(L_i) \\
+ \sum_{j \in chpH(i)} \left[ \frac{R_i(HI)^*}{T_j} \right] C_j(HI) \\
+ \sum_{k \in chpL(i)} \left[ \frac{R_i(HI)^*}{T_k} \right] C_k(LO) \\
+ \sum_{l \in chpMIG(i)} \left[ \frac{R_i(LO)}{T_l} \right] C_l(LO)
\]

- Schedulability issues after migration
Semi-partitioned Model Configuration

- Two approaches on deciding migrating tasks
  - Semi1: task fetched
  - Semi2: task assigned
- Three bin-packing algorithms
  - FF, BF, WF
- Audeslay’s priority assignment
Evaluation

- \( f = 3 \)
- \( P = 0.5 \)
- \( n = 12 \)
- \( ts = 10000 \)
Evaluation

- Weighted graphs
  - Taskset size
  - Percentage of HI-crit tasks
  - Factor of WCET differences
- All algorithms behave better than the non-migration algorithm
- Semi2WF and Semi2FF
Conclusion

- Semi-partitioned model
- Six possible approaches
- Suggestion
- Future Work