Extending Mixed Criticality Scheduling
WMC - RTSS 2013

Tom Fleming & Alan Burns

December 3rd, 2013
Why $n$ criticality levels?
- AMCrtb
- AMCmax
- Period Transformation
- Evaluation
- Conclusions
Why $n$ Criticality Levels?

Extending Mixed Criticality Scheduling to $n$ Criticality levels.

Why?

- IEC 61508 and DO-178B support up to 5 criticality levels.
- Future Standards might support more!
Adaptive Mixed Criticality

Assigns priorities via Audsley’s Algorithm [1].

On a criticality change (LO → HI) AMC suspends all LO criticality tasks. ¹

Baruah et al. [2] show that AMC dominates SMC for Dual Criticality systems.

Two analytical techniques: AMCrhb and AMCmax.

¹ Jobs currently executing are allowed to complete.
Stage 1A: Check the schedulability of the LO mode for all tasks.

\[ R_i(LO) = C_i(LO) + \sum_{j \in hp(i)} \left\lceil \frac{R_i(LO)}{T_j} \right\rceil C_j(LO) \]  \hspace{1cm} (1)

Stage 1B: Repeat 1A for HI criticality

Stage 2A: Calculate the schedulability of the criticality change for HI tasks.

\[ R_i^*(HI) = C_i(HI) + \sum_{j \in hpH(i)} \left\lceil \frac{R_i^*(HI)}{T_j} \right\rceil C_j(HI) + \sum_{k \in hpL(i)} \left\lceil \frac{R_i(LO)}{T_k} \right\rceil C_k(LO) \] \hspace{1cm} (2)
Stages 1A and 1B can be combined into an equation that considers the schedulability of all criticality levels.

\[ \forall L \in 1 \ldots n \]

\[ \forall \tau_i | L_i \geq L \]

\[ R_i(L) = C_i(L) + \sum_{j \in hp(i) | L_j \geq L} \left\lfloor \frac{R_i(L)}{T_j} \right\rfloor C_j(L) \]

(3)
We must consider those higher priority, but lower criticality tasks that have a bounded effect.

$$\sum_{k \in hp(i) \mid L_k < L_i} \left[ \frac{R_i(L_k)}{T_k} \right] C_k(L_k)$$
Therefore the complete equation for the stage 2A:

\[ \forall L \in 1 \ldots n \]

\[ \forall \tau_i \mid L_i \geq L \]

\[ R_i^*(L) = C_i(L) + \sum_{j \in hp(i) \mid L_j \geq L} \left[ \frac{R_j^*(L)}{T_j} \right] C_j(L) + \sum_{k \in hp(i) \mid L_k < L_i} \left[ \frac{R_i(L_k)}{T_k} \right] C_k(L_k) \]  

(4)
There are a finite number of points at which a criticality change might occur.

\[ R_i^s(HI) = C_i^s(HI) + I_L(s) + I_H(s) \]
There are now two possible points of $s$, $s_1$ and $s_2$. For each point of $s_1$ there are a number of points of $s_2$. 
Period Transformation (PT)

Three different groups of tasks:

- **LO** Criticality Tasks.
- **HI** criticality tasks with a period shorter than the shortest **LO** criticality task.
- **HI** criticality tasks with a period greater than that of the shortest **LO** criticality task.

Only the final group of tasks require transformation.
Tasks are transformed by a factor, $m$.

$$m = \left\lceil \frac{T_j}{T_l} \right\rceil$$

Where $\tau_l$ is the LO criticality task with the shortest period and $\tau_j$ is a HI criticality task that must be transformed.

At runtime, transformed tasks are expected to execute up to their HI criticality transformed execution budget ($C_j(HI)/m$) until they reach their untransformed LO criticality execution budget ($C_j(LO)$), only then can we determine if a task will overrun its LO execution bounds and a criticality change would need to occur.
\( P \) represents the remaining transformed executions that do not constitute a complete \( C_j(LO) \).
Vestal [3] calculates the number of complete \( LO \) executions and assumes the value of \( C_j(LO) \) for the remaining transformed \( HI \) executions that do not constitute a complete (untransformed) \( LO \).

\[
\left\lfloor \frac{R_i}{T_j} \right\rfloor C_j(LO) + C_j(LO)
\]
Rather than using an entire LO execution to account for those remaining transformed executions, it is possible to calculate their effect more accurately.

Calculate the size of the remaining interval:

\[ P = R_i - \left\lfloor \frac{R_i}{T_j} \right\rfloor T_j \]

Calculate the number of transformed executions.

\[ x = \left\lceil \frac{P}{T_j/m} \right\rceil \frac{C_j(HI)}{m} \]

Thus:

\[ \min\{x, C_j(LO)\} + \left\lfloor \frac{R_i}{T_j} \right\rceil C_j(LO) \]
The analysis for $n$ criticality levels is almost identical.

Transformed tasks execute at their own criticality level, $C_j(L_j)/m$ until they constitute a complete execution at the criticality level being considered.
The problem at $n$ criticality levels is ensuring that the resulting tasks set is criticality monotonic. Consider the following task set.

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>80</td>
<td>HI</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>110</td>
<td>ME</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>100</td>
<td>LO</td>
</tr>
</tbody>
</table>

- Initially it seems that only $\tau_2$ requires transformation.
- The resulting set, $(80,55,100)$ is not criticality monotonic.
- We must then transform $\tau_1$ to give it a period of 40.
Evaluation

We investigated the performance of each algorithm using randomly generated task sets.

- 5000 task sets per 2% utilisation.
- Evenly distributed criticality levels.
Two Criticality levels
Three Criticality levels

![Graph showing schedulable task sets (%) against utilisation for different algorithms: PT, AMCmax, AMCrhb, SMC, SMC-NO, CrMPO.](image)
Four Criticality levels
Five Criticality levels

![Graph showing the relationship between utilisation and schedulable task sets for different criticality levels. The graph features multiple lines representing different scheduling algorithms: PT, AMCmax, AMCrtb, SMC, SMC-NO, and CrMPO. The x-axis represents utilisation ranging from 0 to 1.6, while the y-axis shows schedulable task sets ranging from 0 to 100%. The lines indicate the proportion of tasks that can be scheduled as utilisation increases.]
Conclusions

- AMCrtb maintains its performance at greater than 2 criticality levels compared with SMC.
- AMCrtb continues to provide a good approximation of AMCmax at reduced processing cost.
- Period Transformation appears to perform well with lower numbers of criticality levels, however this performance tails off and the technique still suffers from high overheads.
N. Audsley.
Optimal priority assignment and feasibility of static priority tasks with arbitrary start times, 1991.

S. Baruah, A. Burns, and R. Davis.
Response-time analysis for mixed criticality systems.

S. Vestal.
Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance.