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 DOB-178B support up to 5 criticality levels.
- Future Standards might support more!

Adaptive Mixed Criticality

Adaptive Mixed Criticality

- Assigns priorities via Audsley's Algorithm [1].
- On a criticality change (LO $\longrightarrow$ HI) AMC suspends all LO criticality tasks. [1]
- Baruah et al. [2] show that AMC dominates SMC for Dual Criticality systems.
- Two analytical techniques: AMCrtb and AMCmax.

[1] Jobs currently executing are allowed to complete.

AMCrtb

Stage 1A: *Check the schedulability of the LO mode for all tasks.*

$$R_i(LO) = C_i(LO) + \sum_{j \in hp(i)} \left\lceil \frac{R_i(LO)}{T_j} \right\rceil C_j(LO) \quad (1)$$

Stage 1B: *Repeat* 1A *for HI criticality*

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

$$R_i^*(HI) = C_i(HI) + \sum_{j \in hpH(i)} \left\lceil \frac{R_i^*(HI)}{T_j} \right\rceil C_j(HI) + \sum_{k \in hpL(i)} \left\lceil \frac{R_i(LO)}{T_k} \right\rceil C_k(LO) \quad (2)$$

AMCrtb n Criticality Levels

Stages 1A and 1B can be combined into an equation that considers the schedulability of all criticality levels.

$$\forall L \in 1 \dots n$$

$$\forall \tau_i | L_i \geq L$$

$$R_i(L) = C_i(L) + \sum_{j \in hp(i) | L_j \geq L} \left\lceil \frac{R_i(L)}{T_j} \right\rceil C_j(L) \tag{3}$$

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

$$\sum_{k \in hp(i) | L_k < L_i} \left\lceil \frac{R_i(L_k)}{T_k} \right\rceil C_k(L_k)$$

Therefore the complete equation for the stage 2A:

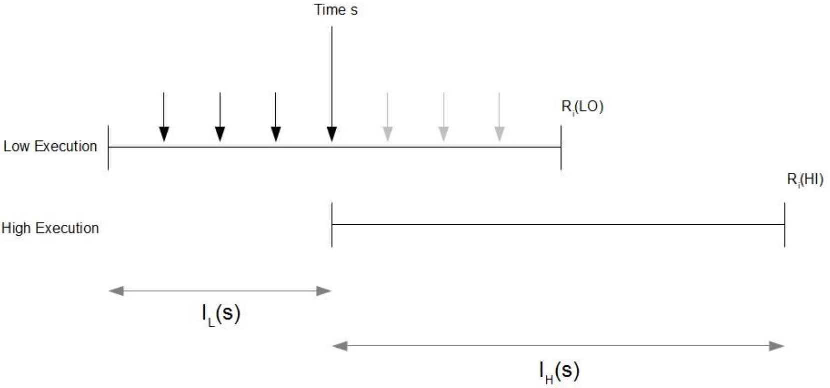
$$\forall L \in 1 \ldots n$$

$$\forall \tau_i | L_i \geq L$$

$$R_i^*(L) = C_i(L) + \sum_{j \in hp(i) | L_j \geq L} \left\lceil \frac{R_i^*(L)}{T_j} \right\rceil C_j(L) \quad + \sum_{k \in hp(i) | L_k < L_i} \left\lceil \frac{R_i(L_k)}{T_k} \right\rceil C_k(L_k) \tag{4}$$

AMCmax

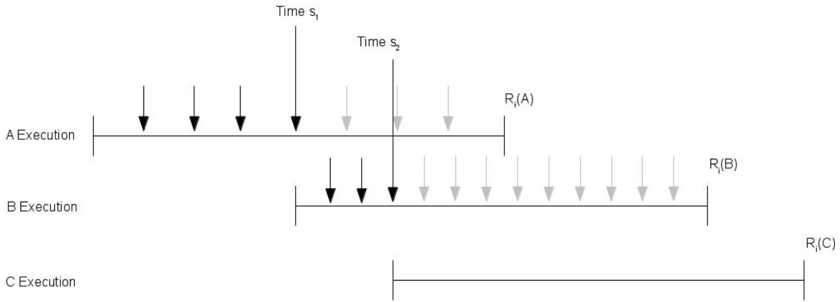
There are a finite number of points at which a criticality change might occur.



$$R_i^s(HI) = C_i(HI) + I_L(s) + I_H(s) \tag{5}$$

AMCmax *n* Criticality Levels

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.

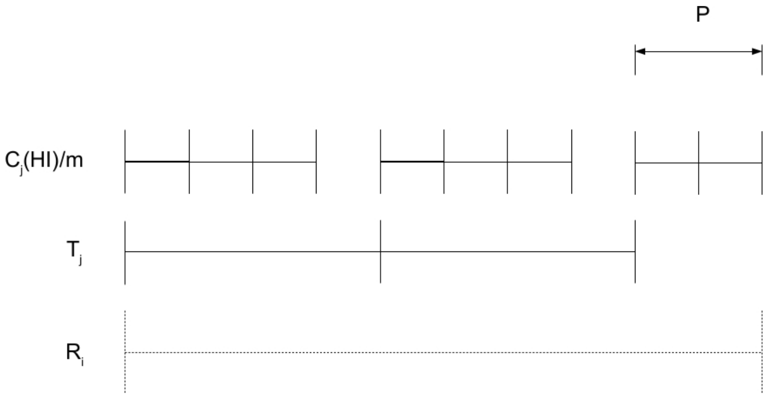
Only the final group of tasks require transformation.

Tasks are transformed by a factor, m.

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

Where τ_l is the LO criticality task with the shortest period and τ_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 consitute 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:

$$min\{x, C_j(LO)\} + \left\lfloor \frac{R_i}{T_j} \right\rfloor C_j(LO) \tag{6}$$

PT n Criticality Levels

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.

	T	L
τ_1	80	*HI*
τ_2	110	*ME*
τ_3	100	*LO*

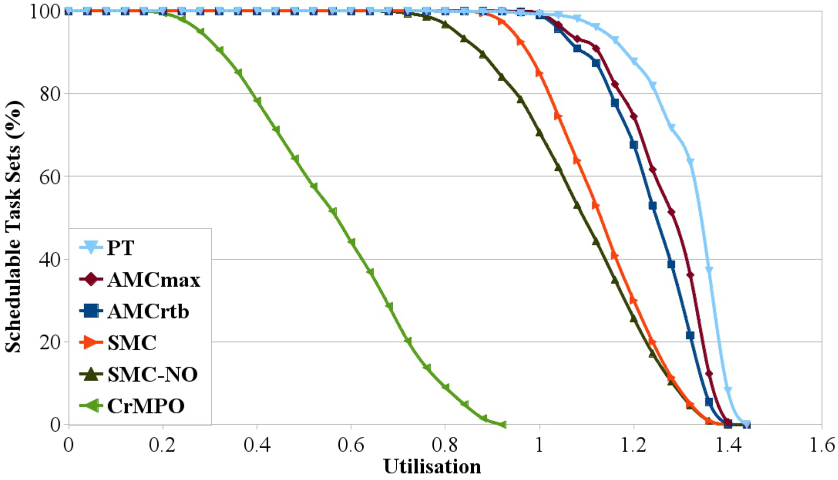
- Initially it seems that only τ_2 requires transformation.
- The resulting set, (80,55,100) is not criticality monotonic.
- We must then transform τ_1 to give it a period of 40.

Evaluation

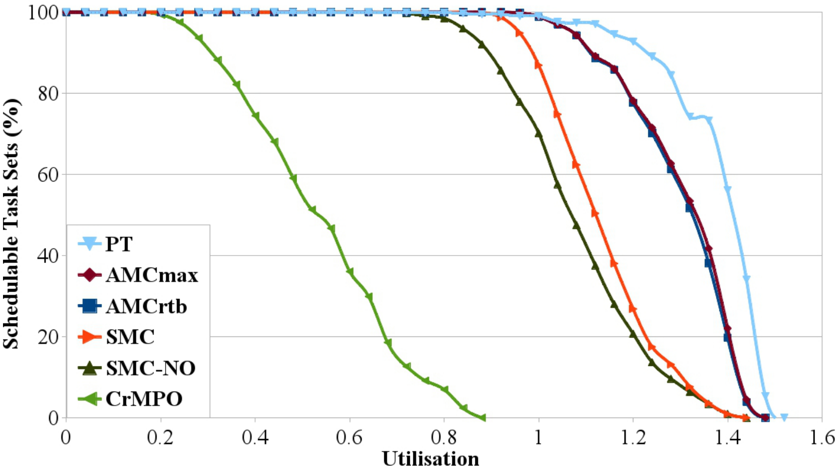
We investigated the performance of each algorithm using randomly generated task sets.

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

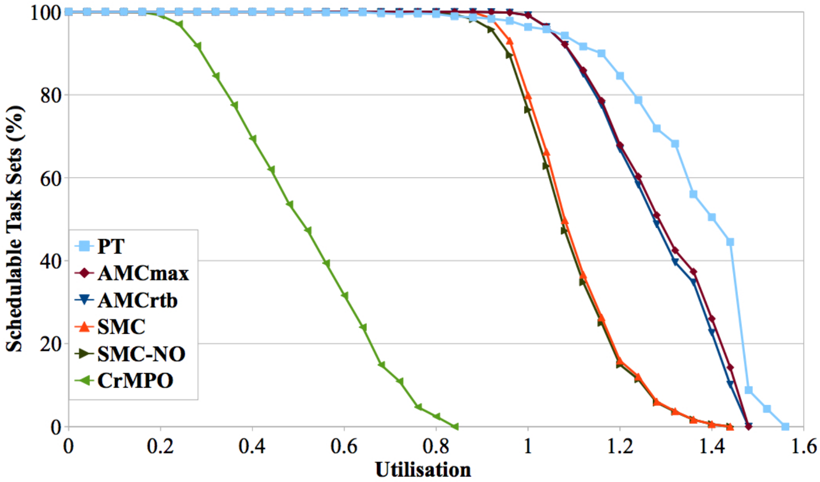
Two Criticality levels



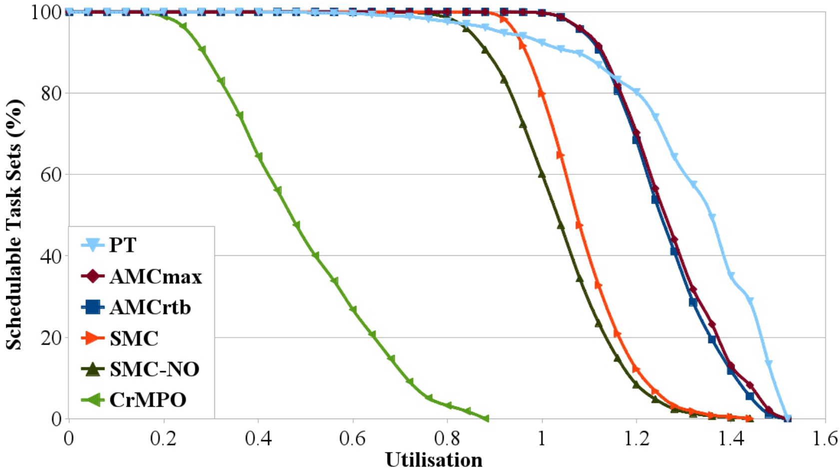
Three Criticality levels



Four Criticality levels



Five Criticality levels



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 sufferes 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.
In *Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd*, pages 34 –43, 29 2011-dec. 2 2011.

S. Vestal.
Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance.
In *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International*, pages 239 –243, dec. 2007.