Scheduling Multi-Periodic Mixed-Criticality DAGs on Multi-Core Architectures

Roberto MEDINA
Etienne BORDE
Laurent PAUTET

December 13, 2018
Outline

Research Context

Problem Statement

Scheduling MC-DAGs on multi-cores

Case Study

Performance tests

Conclusion and perspectives
Outline

Research Context
  WCET estimation
  Mixed-criticality execution
  Data-flow model of computation

Problem Statement

Scheduling MC-DAGs on multi-cores

Case Study

Performance tests

Conclusion and perspectives
Research context

- **Safety-critical systems**: stringent time requirements + software components with different criticalities.
  - Outputs on time.
  - Life-critical, mission-critical and non-critical.
  - Often isolated: architecture or software level.

Current industrial trends

- Reduce size, weight, power consumption, heat.
- Integrate and deliver more services.
- **Multi-core architectures**: great processing capabilities

- Large overestimation of execution time → waste of CPU.
Timeliness: WCET estimation

- Real-time systems dimensioned with Worst Case Execution Time (WCET).
- Estimating the WCET: a difficult problem.\(^1\)
  - Various methods to obtain an estimate.
  - Multi-core architectures hardly predictable.
  - Task rarely executes until its WCET.

Mixed-Criticality (MC) model

MC model to overcome poor resource usage\(^2\).

1. Different timing budgets.
   - \( C_i(LO) \): Max. observed execution time (system designers).
   - \( C_i(HI) \): Upper-bounded execution time (static analysis).

2. Incorporate tasks with different criticality levels: HI and LO.

3. Execution modes:
   - LO-criticality mode: HI tasks + LO tasks.
   - HI-criticality mode: only HI tasks \(\rightarrow\) LO tasks discarded.

Schedulability with mode transitions

- Example: schedule the task set \(\{\tau_1, \ldots, \tau_4\}\).
- HI-criticality tasks: \(\tau_1, \tau_3\). LO-criticality tasks: \(\tau_2, \tau_4\).
Schedulability with mode transitions

- Example: schedule the task set \( \{\tau_1, \ldots, \tau_4\} \).
- HI-criticality tasks: \( \tau_1, \tau_3 \). LO-criticality tasks: \( \tau_2, \tau_4 \).

*Mode transitions: potential deadline misses.*

*Time drifts when tasks are data-dependent...*
Designing safety-critical applications thanks to data-flows

- Models of Computation: data-flow & Directed Acyclic Graphs (DAGs).
  - Deterministic communication patterns.
  - Boundedness in memory, deadlock/starvation freedom...
- Industrial tools based on these model (e.g. Simulink, SCADE).
  - Code generation, automatic deployment into architecture.
Outline

Research Context

Problem Statement

Scheduling MC-DAGs on multi-cores

Case Study

Performance tests

Conclusion and perspectives
Problem statement: scheduling data-dependent MC tasks

- MC scheduling is intractable: \textbf{NP-hard} problem\(^3\).
- Multiple DAG scheduling in multi-core architectures: \textbf{NP-complete} problem\(^4\).

Industrial systems with both: MC task + DAGs

---


Problem statement: scheduling data-dependent MC tasks

- MC scheduling is intractable: **NP-hard** problem\(^3\).
- Multiple DAG scheduling in multi-core architectures: **NP-complete** problem\(^4\).

Industrial systems with **both**: MC task + DAGs

Existing works and current limitations

- For DAGs: List Scheduling efficient heuristic.
  - No variations in execution time in the literature.
  - No mode transitions for the system.
- For MC task sets: many different scheduling policies.
  - Rarely take into account data-dependencies (DAG).
  - When they do, systems are overdimensioned... again!

---

\(^3\) Baruah, “Mixed criticality schedulability analysis is highly intractable”.

\(^4\) Kwok and Ahmad, “Static scheduling algorithms for allocating directed task graphs to multiprocessors”.
Outline

Research Context

Problem Statement

Scheduling MC-DAGs on multi-cores
   MC-correct schedules for MC-DAGs
   Safe mode transition property
   Meta-heuristic for MC-DAGs

Case Study

Performance tests

Conclusion and perspectives
Definition

A **MC-correct** schedule is one which guarantees:

1. **Condition LO-mode**: If no vertex of any MC-DAG executes beyond its $C_i(LO)$ then all the vertices complete execution by their deadlines.

2. **Condition HI-mode**: If no vertex of any MC-DAG executes beyond its $C_i(HI)$ then all the vertices designated as being of HI-criticality complete execution by their deadlines.

---

Safe mode transitions general property

▶ **Intuition**: At any instant $t$, HI task execution time given in LO mode at least equal to the execution time given in HI mode.

▶ $\psi^\chi_i(t_1, t_2)$: cumulative execution time given to task $\tau_i$ in mode $\chi$ from $t_1$ to $t_2$.

Safe Transition Property

$$\psi^LO_i(r_{i,k}, t) < C_i(LO) \implies \psi^LO_i(r_{i,k}, t) \geq \psi^HI_i(r_{i,k}, t). \quad (1)$$
Meta-heuristic for MC-DAGs Scheduling

- Solve the complex scheduling problem off-line: computing **static scheduling tables**.
  - Easier to verify and have certified.
  - Easier to calculate $\psi_i^\chi$, enforce **Safe Transition Property**.

### MH-McDag

1. Compute static scheduling in HI-criticality mode.
2. Compute static scheduling in LO-criticality mode, enforcing **Safe Transition Property**.

Produces **MC-correct** schedulers for MC-DAGs.

- Existing multi-core schedulers can be adapted to produce **MC-DAG schedulers**.
  - Global-Least Laxity First and Global-Earliest Deadline First.
Outline

Research Context

Problem Statement

Scheduling MC-DAGs on multi-cores

Case Study

Unmanned Air Vehicle for field exploration
Efficient implementations of MH-McDAG

Performance tests

Conclusion and perspectives
Case Study: unmanned air vehicle (UAV)

\[ U_{\text{max}} = U_{\text{FCS}} + U_{\text{Montage}} = 1.8 + 1.05 = 2.85. \]
Application of the federated approach

Figure 2: Five cores required for the federated scheduling approach

Limitations

1. Single DAG has exclusive access to a cluster of cores.
2. HI tasks scheduled ASAP in the LO-criticality mode.
   - Respects Safe Trans. Prop. but...
   - LO-criticality task scheduling too constrained.
   - No longer necessary with Safe Trans. Prop.
How to improve resource usage with MC-DAGs?

Two main strategies

▶ Adopt a **global multi-core scheduling**
  → MC-DAGs share cores (better resource usage)
▶ As late as possible (ALAP) policy in the HI mode
  → Relax HI-criticality tasks execution in the LO mode.

**Genericity** of our implementation (**G-ALAP**)

▶ *Deadlines* (based on Global-Earliest Deadline First).
▶ *Laxities* (based on Global-Least Laxity First).
Earliest deadline priority ordering

- Ready task jobs sorted by a “virtual deadline”.
- Virtual deadline for a job $k$ of task $\tau_i$ in mode $\chi$:
  \[ D_{i,k}^\chi = d_{i,k} - CP_i^\chi. \]  
  (2)
- $d_{i,k}$ deadline of the $k$-th activation of the MC-DAG.
- $CP_i^\chi$ critical path to the vertex.
Computed scheduling tables w/ \textit{G-ALAP-EDF}

(a) HI-criticality scheduling w/ ALAP behavior

(b) LO-criticality scheduling

From five cores to \textbf{three cores}
Laxity-based priority ordering

- Ready tasks sorted by their laxities.
- Laxity for a job $k$ of task $\tau_i$:

$$L_{i,k}^\chi(t) = d_{i,k} - t - (C_{P_i}^\chi + R_{i,k}^\chi).$$  \hfill (3)

- $d_{i,k}$ deadline of the $k$-th activation of the MC-DAG.
- $t$ current time slot.
- $C_{P_i}^\chi$ critical path to the vertex.
- $R_{i,k}^\chi$ remaining execution time.
  - Initialized with $C_i(LO)$ or $C_i(HI)$. 
Outline

Research Context

Problem Statement

Scheduling MC-DAGs on multi-cores

Case Study

Performance tests
  - MC-DAG generation
  - Acceptance rate results

Conclusion and perspectives
MC-DAG generation

- Unbiased random generation of MC-DAGs.
  - Avoid particular DAG shapes\(^6\).
  - System’s utilization is uniformly distributed among vertices\(^7\).
- Configurable parameters:
  - Edge probability.
  - Number of vertices.
  - Number of MC-DAGs.
  - Utilization of the system.
  - Ratio HI/LO-criticality tasks.
- Open source framework\(^8\).

---


\(^8\) MC-DAG framework - [https://github.com/robertoxmed/MC-DAG](https://github.com/robertoxmed/MC-DAG)
Experimentation setup

- Generated large number of MC systems (1000 systems/configuration).
- Fixed the number of cores and vertices.
- Vary the utilization of the system.
- Vary the number of MC-DAGs.
- Vary the density of the graph (probability to have an edge).
- Measured the acceptance rate in function of the normalized utilization.
Significant performance increase

- Comparison between our \textit{G-ALAP} implementations and \textit{FedMcDAG}\textsuperscript{5}.

\begin{figure}[h]
\centering
\begin{subfigure}{0.49\textwidth}
\centering
\includegraphics[width=\textwidth]{chart_a.png}
\caption{$e = 20\%, |G| = 2$ and $m = 4$.}
\end{subfigure} \hfill
\begin{subfigure}{0.49\textwidth}
\centering
\includegraphics[width=\textwidth]{chart_b.png}
\caption{$e = 20\%, |G| = 4$ and $m = 4$.}
\end{subfigure}
\end{figure}

- Better schedulability when the number of \textit{MC-DAGs} increases.
Significant performance increase

When MC-DAGs are denser (parameter $e$):

- More difficult to schedule a MC system.
- Still better schedulability than existing approaches.

\[(c)\] $e = 20\%$, $|G| = 2$ and $m = 4$.

\[(d)\] $e = 40\%$, $|G| = 2$ and $m = 4$. 
Outline

Research Context

Problem Statement

Scheduling MC-DAGs on multi-cores

Case Study

Performance tests

Conclusion and perspectives
Conclusion on MC-DAG scheduling

- Designed a meta-heuristic to obtain various schedulers for DAGs on Mixed-Criticality systems.
- Meta-heuristic proven to be correct:
  - Schedulability on both modes (HI & LO).
  - Safe mode transitions to higher criticality mode.
- Our implementations outperform the state of the art.
  - More systems are schedulable considering a given architecture.
  - Good acceptance rate even when the utilization is high.

Perspectives

- Support an arbitrary number of criticality levels.
- Perform benchmarks on number of preemptions.
Entailed number of preemptions

![Graph](image)

(a) $e = 20\%$, $|G| = 2$, $m = 4$.  
(b) $e = 40\%$, $|G| = 2$, $m = 4$.

**Figure 3**: Average number of preemptions per job (log scale)

- Number of preemptions for systems schedulable with all methods.