Scaling Up the Memory Interference Analysis for Hard Real-Time Many-Core Systems

Maximilien Dupont de Dinechin\textsuperscript{1}, Matheus Schuh\textsuperscript{2,3}, Matthieu Moy\textsuperscript{1}, Claire Maiza\textsuperscript{2}

\textsuperscript{1}Univ Lyon, EnsL, UCBL, CNRS, Inria
\textsuperscript{2} Univ. Grenoble Alpes CNRS, Grenoble INP, VERIMAG
\textsuperscript{3} Kalray SA
• Safety-Critical Hard Real-Time Systems
  • *Execution time* of the program **must be predictable**
  • *Timing constraints* are part of the specification
  • Challenge is moving to **multi/many-cores**

• Data-Flow languages, *safe release dates* respecting *precedence*

• Original algorithm in $O(n^4)$: we try to do better!
• Task set: Directed Acyclic Graph (DAG)
  • Obtained from Data-Flow languages
  • Dependencies, order, mapping and Worst-Case Execution Time (WCET) in isolation are given

Mapping = \[
\begin{align*}
  n_0 & \mapsto \text{PE0} \\
  n_1, n_2 & \mapsto \text{PE1} \\
  n_3 & \mapsto \text{PE2} \\
  n_4 & \mapsto \text{PE3}
\end{align*}
\]

WCET = \[
\begin{align*}
  n_0, n_1, n_4 & : 2 \\
  n_2 & : 1 \\
  n_3 & : 3
\end{align*}
\]
• Access to shared resources
  • Controlled by hardware **arbiters**
  • **Huge** impact on execution time
Context: Interference Computation Problem

- Static Time-Triggered Schedule
  - *Release date* for each task
  - Worst-Case Response Time (WCRT) of the program
Original Algorithm

• Global fixed point iterations
  • Response times
  • Adjust release dates

• $O(n^4)$ complexity

• *Interference* and *execution time* are *inter-dependent* problems
Original Algorithm

0. Input (Isolated WCET)

1. Estimate current interference

2. Adjust release dates

3. Check schedulability

16th March 2020
• Schedule computed **incrementally**
  • Task is released when:
    • Dependencies are satisfied
    • A *minimal release date* (if any) was reached
  • Release date is **fixed**
  • Interference added to or received from *alive tasks* is then computed

• $O(n^2)$ complexity
  • *Scales for thousands of tasks*
Our Improved Algorithm

PE0: $n_0 \rightarrow n_1 \rightarrow n_2$

PE1: $n_3 \rightarrow n_4$

PE2: $n_5 \rightarrow n_6 \rightarrow n_7$

PE3: $n_8 \rightarrow n_9 \rightarrow n_{10}$

Dead: $n_3, n_5, n_8, n_6$

Alive: $n_0, n_4, n_7, n_9$

Future: $n_1, n_2, n_{10}$

$t \geq \text{end}_{date}$

$\text{rel}_{date} \leq t < \text{end}_{date}$

$t < \text{rel}_{date}$
• Generation of random DAGs to stress the algorithm
  • Tasks on the same layer are assigned to cores in a cyclic way
  • Random WCET and memory accesses
Final words

• Conclusion
  • **Interference** analysis algorithm in $O(n^2)$ for many-core response time computation, able to quickly schedule *thousands* of tasks

• Future work
  • **Overlap** treatment
    • Tasks with multiple co-runners: treated as a single big task
    • Sum their WCETs and memory accesses
Thanks for your attention!

To read the full paper:
https://hal.archives-ouvertes.fr/hal-02431273

You can find me at:
mschuh@kalray.eu
matheus.schuh@univ-grenoble-alpes.fr