Distributed run-time WCET controller for concurrent critical tasks in mixed-critical systems

Angeliki Kritikakou ‡‡, Claire Pagetti †, Christine Rochange ‡, Matthieu Roy *, Madeleine Faugere , Sylvain Girbal and Daniel Gracia Perez

09/10/2014, RTNS’14, Versailles, France
Outline

• Introduction & Motivation

• Overview

• Design time analysis

• Run-time control

• Evaluation results

• Conclusions & Future directions
Target domain

- **Software**: Mixed Critical applications
  - Hard & soft deadlines
  - Different consequences in failure
  - Criticality Levels: DAL, ASIL, ...
  - Domains: Aeronautics, Automotive, ...
Target domain

- **Software:** Mixed Critical applications
  - Hard & soft deadlines
  - Different consequences in failure
  - Criticality Levels: DAL, ASIL, ...
  - Domains: Aeronautics, Automotive, ...

- **Hardware:** Multi/many-cores
  - Cores with components with dynamic behavior
  - Shared resources
  - COTS: Tilera, Kalray, Texas TMS, ...
Target domain

• **Software:** Mixed Critical applications
  – Hard & soft deadlines
  – Different consequences in failure
  – Criticality Levels: DAL, ASIL, ...
  – Domains: Aeronautics, Automotive, ...

• **Hardware:** Multi/many-cores
  – Cores with components with dynamic behavior
  – Shared resources
  – COTS: Tilera, Kalray, Texas TMS, ...

**Challenge:** Guarantee hard real-time response & improve cores utilization
Motivation

- Safe WCET static analysis for each critical task
Motivation

• Safe WCET static analysis for each critical task

  Many active cores

• Maximum load
Motivation

- Safe WCET static analysis for each critical task
  
  **Many active cores**
  
- Maximum load
- Full congestion
Motivation

- Safe WCET static analysis for each critical task
  
  **Many active cores**
  
- Maximum load
- Full congestion

\[ \text{WCET} > D_c \]
Motivation

- Safe WCET static analysis for each critical task

**Many active cores**
- Maximum load
- Full congestion

**Few active cores**
- Isolation

![Diagram showing many active cores vs. few active cores with WCET > DC]
Motivation

• Safe WCET static analysis for each critical task

Many active cores
• Maximum load
• Full congestion

Few active cores
• Isolation
• Congestion of critical tasks

WCET > D_c
Motivation

- Safe WCET static analysis for each critical task

**Many active cores**
- Maximum load
- Full congestion

**Few active cores**
- Isolation
- Congestion of critical tasks

WCET > $D_c$

WCET ≤ $D_c$
Contribution

Existing approaches: Not directly applicable
Contribution

- Existing approaches: Not directly applicable
- Safe: Only the most critical tasks
- Under-utilization of the system resources
Contribution

Existing approaches: Not directly applicable

Safe: Only the most critical tasks

Under-utilization of the system resources

Proposed: Maximum load if safe, ELSE switch to isolation

Improved system resources utilization

Guarantee critical deadlines
Contribution

- Existing approaches: Not directly applicable
- Safe: Only the most critical tasks
  - Under-utilization of the system resources
- Proposed: Maximum load if safe, ELSE switch to isolation
  - Improved system resources utilization
  - Guarantee critical deadlines

[ECRTS’14] → Approach to **more than** 1 critical task
Implementation on real multicore platform
Methodology overview: Design-time

Tasks 1… N

Partition: Critical Vs less critical

Critical tasks

Master +

Less Critical Tasks

Configured platform

CPU 1  CPU 2  CPU 3  CPU 4

Memory
Methodology overview: Design-time

Tasks 1… N

Partition: Critical Vs less critical

Critical tasks

Instrumentation

Instrumented source code

Less Critical Tasks

Source code

Configured platform

Memory

CPU 1

CPU 2

CPU 3

CPU 4
Methodology overview: Design-time

Tasks 1… N

Partition: Critical Vs less critical

Critical tasks

Instrumentation

Instrumented source code

Master + Less Critical Tasks

Source code

Compiler

Configured platform

Memory

CPU 1
CPU 2
CPU 3
CPU 4
Methodology overview: Design-time

Tasks 1... N

Partition: Critical Vs less critical

Critical tasks

Instrumented source code

Instrumentation

Bin.code

Structure & Timing Analysis

Compiler

Master +

Less Critical Tasks

Source code

Configured platform

Memory

CPU 1

CPU 2

CPU 3

CPU 4
Methodology overview: Design-time

Tasks 1… N

Partition: Critical Vs less critical

Critical tasks

Instrumented source code

Instrumentation

Compiler

Source code

Bin.code

Master +

Structure & Timing Analysis

Bin.code

S & T info

Configured platform

Memory

Less Critical Tasks

Bin.code

Bin.code

Bin.code

CPU 1

CPU 2

CPU 3

CPU 4

A. Kritikakou
Methodology overview: Run-time
Methodology overview: Run-time
Methodology overview: Run-time
Methodology overview: Run-time
Instrumentation

- Model task as a set of functions: \( S = \{F_0, \ldots, F_n\} \)
  - Each function is an Extended Control Flow Graph (ECFG)
Instrumentation

- Model task as a set of functions: \( S = \{F_0, \ldots, F_n\} \)
  - Each function is an Extended Control Flow Graph (ECFG)

- Observation points per node & at the end of ECFG of \( F_0 \)
**Instrumentation**

- Model task as a set of functions: \( S = \{F_0, \ldots, F_n\} \)
  - Each function is an Extended Control Flow Graph (ECFG)

\[
\begin{array}{cccc}
\text{IN} & \downarrow & \text{OUT} \\
\circ & \text{N} & \text{F} & \text{C} \\
\end{array}
\]

- Observation points per node & at the end of ECFG of \( F_0 \)

<table>
<thead>
<tr>
<th>( C_R )</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Instrumentation**

- Model task as a set of functions: $S = \{F_0, \ldots, F_n\}$
  - Each function is an Extended Control Flow Graph (ECFG)

- Observation points per node & at the end of ECFG of $F_0$

<table>
<thead>
<tr>
<th>$C_R$</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Maximum load)</td>
<td>Call $F_R$</td>
</tr>
</tbody>
</table>

$C_R^n$
Instrumentation

- Model task as a set of functions: \( S = \{F_0, \ldots, F_n\} \)
  - Each function is an Extended Control Flow Graph (ECFG)

\[
\begin{array}{ccc}
\text{IN} & \downarrow & \text{OUT} \\
\end{array}
\]

- Observation points per node & at the end of ECFG of \( F_0 \)

<table>
<thead>
<tr>
<th>( C_R )</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Maximum load)</td>
<td>Call ( F_R )</td>
</tr>
<tr>
<td>0 (Isolation)</td>
<td>-</td>
</tr>
</tbody>
</table>
Instrumentation

- Model task as a set of functions: \( S = \{F_0, \ldots, F_n\} \)
  - Each function is an Extended Control Flow Graph (ECFG)

\[
\begin{array}{ccc}
\text{IN} & \overset{\diamond}{\circ} & \text{OUT} \\
\end{array}
\]

- Observation points per node & at the end of ECFG of \( F_0 \)

<table>
<thead>
<tr>
<th>( C_R )</th>
<th>Node</th>
<th>End of ECFG</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Maximum load)</td>
<td>Call ( F_R )</td>
<td></td>
</tr>
<tr>
<td>0 (Isolation)</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
**Instrumentation**

- Model task as a set of functions: \( S = \{F_0, \ldots, F_n\} \)
  - Each function is an Extended Control Flow Graph (ECFG)

- Observation points per node & at the end of ECFG of \( F_0 \)

<table>
<thead>
<tr>
<th>( C_R )</th>
<th>Node</th>
<th>End of ECFG</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Maximum load)</td>
<td>Call ( F_R )</td>
<td>-</td>
</tr>
<tr>
<td>0 (Isolation)</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
**Instrumentation**

- **Model task as a set of functions:** \( S = \{F_0, \ldots, F_n\} \)
  - Each function is an Extended Control Flow Graph (ECFG)

\[
\begin{align*}
\text{IN} & \quad \downarrow \quad \text{OUT} \\
N & \quad F & \quad C
\end{align*}
\]

- **Observation points per node & at the end of ECFG of \( F_0 \)**

<table>
<thead>
<tr>
<th>( C_R )</th>
<th>Node</th>
<th>End of ECFG</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Maximum load)</td>
<td>Call ( F_R )</td>
<td>-</td>
</tr>
<tr>
<td>0 (Isolation)</td>
<td>-</td>
<td>Notify termination</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\text{IN} & \quad \downarrow \quad \text{OUT}_0 \\
N & \quad C_R & \quad F_R \\
\end{align*}
\]
Instrumentation example

\[ F_0 \]

\[ \text{IN}_0 \]

\[ N_1 \]

\[ F_1 \]

\[ N_4 \]

\[ \text{OUT}_0 \]

\[ F_1 \]

\[ \text{IN}_1 \]

\[ N_2 \]

\[ C \]

\[ N_3 \]

\[ \text{OUT}_1 \]
Instrumentation example

\[ F_0 \]

\[ IN_0 \]

n₁

\[ N_1 \]

f₁

\[ F_1 \]

n₄

\[ N_4 \]

out₀

\[ OUT_0 \]

\[ F_1 \]

\[ IN_1 \]

n₂

\[ N_2 \]

c

\[ C \]

\[ N_3 \]

n₃

\[ OUT_1 \]
Instrumentation example

\[ \begin{align*}
C_R &\rightarrow F_R &\rightarrow F_0 &\rightarrow IN_0 &\rightarrow N_1 &\rightarrow C_R &\rightarrow F_R &\rightarrow IN_1 &\rightarrow N_2 \\
C_R &\rightarrow F_R &\rightarrow F_1 &\rightarrow n_1 &\rightarrow F_1 &\rightarrow n_2 &\rightarrow C &\rightarrow N_3 &\rightarrow OUT_0 \\
C_R &\rightarrow F_R &\rightarrow n_4 &\rightarrow N_4 &\rightarrow \overline{C_R} &\rightarrow N_S &\rightarrow out_0 &\rightarrow \overline{C_R} &\rightarrow F_R &\rightarrow C_R
\end{align*} \]
Structure & Timing information

Initialization: $S$

Diagram:

- $IN_0$ connected to $n_1$ to $N_1$
- $n_1$ connected to $f_1$ to $F_1$
- $f_1$ connected to $n_4$ to $N_4$
- $n_4$ connected to $out_0$ to $OUT_0$
- $IN_1$ connected to $n_2$ to $N_2$
- $n_2$ connected to $c$ to $C$
- $c$ connected to $n_3$ to $N_3$
- $N_3$ connected to $OUT_1$

Legend:
- $IN_0$: Input 0
- $IN_1$: Input 1
- $OUT_0$: Output 0
- $OUT_1$: Output 1
Structure & Timing information

Initialization: $S$

- Structure
  - Level

\[ \begin{align*}
\text{Structure} & : & N_1 & \rightarrow & F_1 & \rightarrow & N_4 & \rightarrow & \text{OUT}_0 \\
\text{Timing} & : & \text{IN}_0 & \rightarrow & n_1 & \rightarrow & \text{F}_1 & \rightarrow & n_4 & \rightarrow & \text{OUT}_0 \\
\end{align*} \]
Structure & Timing information

Initialization: $S$

- Structure
  - Level
Structure & Timing information

Initialization: 0 S

- Structure
  - Level

![Diagram of the structure and timing information with nodes and connections labeled with IN0, IN1, N1, N2, N3, OUT0, OUT1, and connections such as n1, f1, n2, c, n3, out0, and in1. The diagram shows a flow from IN0 to OUT0 through intermediate nodes.]
Structure & Timing information

Initialization: $S$

- **Structure**
  - **Level**

---

**IN$_0$**

1. $n_1$

2. $n_4$

---

**IN$_1$**

1. $n_2$

2. $c$

---

**OUT$_0$**

1. $o_{OUT}$

---

**OUT$_1$**

1. $n_3$

---

A. Kritikakou
Structure & Timing information

Initialization: $S$

- Structure
  - Level
  - Type

A. Kritikakou
Structure & Timing information

Initialization: S

- Structure
  - Level
  - Type

![Diagram](image_url)

- Structure & Timing information
- Initialization: S
- Structure
  - Level
  - Type

![Diagram](image_url)
Structure & Timing information

Initialization:  S

- Structure
  - Level
  - Type

F_ENTRY

F_EXIT

A. Kritikakou

10/21
Structure & Timing information

Initialization: $S$

- Structure
  - Level
  - Type
  - Head

![Diagram of structure and timing information]
Structure & Timing information

Initialization: S

• Structure
  – Level
  – Type
  – Head

Structure – Level – Type – Head

A. Kritikakou

10/21
Structure & Timing information

Initialization:

- S

• Structure
  - Level
  - Type
  - Head

Structure & Timing information

Initialization:

- S

• Structure
  - Level
  - Type
  - Head
Structure & Timing information

Initialization: $S$

• Structure
  – Level
  – Type
  – Head

- Structure & Timing information

$S^1_{n_1} \rightarrow N_1 \rightarrow F_1 \rightarrow N_2 \rightarrow C \rightarrow N_3 \rightarrow OUT_1$
Structure & Timing information

Initialization: S

- Structure
  - Level
  - Type
  - Head

\[ S \]

\[ S_{n_1} \] \( N_1 \)

\[ S_{f_1} \] \( F_1 \)

\[ S_{n_4} \] \( N_4 \)

\[ S_{out_0} \] \( OUT_0 \)

\[ IN_0 \]

\[ IN_1 \] \( F_{1n_2} \) \( N_2 \)

\[ F_{1c} \] \( C \)

\[ OUT_1 \] \( N_3 \)

\[ n_3 \]

\[ n_1 \] \( n_2 \) \( n_4 \)
Structure & Timing information

Initialization: $S$

- Structure
  - Level
  - Type
  - Head

- Timing
  - $d(\text{head} \rightarrow \text{point})$

A. Kritikakou
Structure & Timing information

Initialization:

- **Structure**
  - Level
  - Type
  - Head

- **Timing**
  - $d(\text{head} \rightarrow \text{point})$

**Structure**

- Node $N_1$
- Node $N_2$
- Node $N_3$
- Node $N_4$

**Timing**

- Node $C$

Connections:

- $d_{\text{S}-n_1}$
- $d_{\text{S}-f_1}$
- $d_{\text{S}-n_4}$
- $\text{out}_0$
- $\text{out}_1$
- $n_1$
- $n_2$
- $n_3$
- $n_4$
- $f_1$

A. Kritikakou
Structure & Timing information

Initialization: $S$

- Structure
  - Level
  - Type
  - Head

- Timing
  - $d(\text{head} \rightarrow \text{point})$

**Diagram:**

- Nodes: $N_1$, $N_2$, $N_3$, $N_4$
- Edges: $IN_0$, $IN_1$, $OUT_0$, $OUT_1$

- $IN_0$ connected to $N_1$, $N_1$ connected to $F_1$, $F_1$ connected to $N_4$, $N_4$ connected to $OUT_0$
- $IN_1$ connected to $N_2$, $N_2$ connected to $C$, $C$ connected to $N_3$, $N_3$ connected to $OUT_1$
- $d_{f1-n2}$ from $F_1$ to $N_2$
- $d_{f1-c}$ from $F_1$ to $C$
Structure & Timing information

Initialization: S

- Structure
  - Level
  - Type
  - Head

- Timing
  - \( d(\text{head} \rightarrow \text{point}) \)

\begin{itemize}
  \item \( N_1 \)
  \item \( N_2 \)
  \item \( N_3 \)
  \item \( N_4 \)
\end{itemize}
Structure & Timing information

Initialization:

- **Structure**
  - Level
  - Type
  - Head

- **Timing**
  - \(d(\text{head} \rightarrow \text{point})\)
  - \(w\)
Observation point
Observation point

ET

Run-time control

RWCET

Deadline
Observation point

Run-time control

Deadline

ET

RWCET

\( t_{RT} \)

W

RWCET
Observation point

- ET
- RWCET
- t_{RT}
- W
- Continue

Deadline

Run-time control
... later observation point
… later observation point
... later observation point
... later observation point
... later observation point
Switching to isolation: One request
Switching to isolation: One request

- System Bus/Crossbar switch
- Memory

A. Kritikakou
Switching to isolation: One request
Switching to isolation: One request
Switching to isolation: One request
Switching to isolation: One request
Switching to isolation: One request
Switching to isolation: One request
Switching to isolation: Several requests
Switching to isolation: Several requests
Switching to isolation: Several requests
Switching to isolation: Several requests
Switching to isolation: Several requests
Switching to isolation: Several requests
Switching to isolation: Several requests
Switching to isolation: Several requests
Switching to isolation: Several requests
Switching to isolation: Several requests

System Bus/Crossbar switch

Memory
Switching to isolation: Several requests
Switching to isolation: Several requests
Switching to isolation: Several requests

System Bus/Crossbar switch

Memory
Switching to isolation: Several requests
Platform: Texas Instruments TMS320C6678

- 8 TMS320C66x DSP processors
  - clocked at 1 GHz
  - VLIW instruction set architecture
  - 32 KB program memory (L1P)
  - 32 KB data memory (L1D)
  - 512 KB level 2 memory (L2)
Platform: Texas Instruments TMS320C6678

- 8 TMS320C66x DSP processors
  - clocked at 1 GHz
  - VLIW instruction set architecture
  - 32 KB program memory (L1P)
  - 32 KB data memory (L1D)
  - 512 KB level 2 memory (L2)
- Platform configuration
  - Caches → SRAM
  - Data @ DDR, otherwise @ L2 SRAM
  - Partitioned non pre-emptive schedule
Platform: Texas Instruments TMS320C6678

- 8 TMS320C66x DSP processors
  - clocked at 1 GHz
  - VLIW instruction set architecture
  - 32 KB program memory (L1P)
  - 32 KB data memory (L1D)
  - 512 KB level 2 memory (L2)

- Platform configuration
  - Caches → SRAM
  - Data @ DDR, otherwise @ L2 SRAM
  - Partitioned non pre-emptive schedule

- Bare-metal library
  - Time management
  - Message passing
  - Events & Interrupts
Experimental setup

- Number of parallel tasks
  - 1 - 6 cores running less critical
Experimental setup

- Number of parallel tasks
  - 1-6 cores running less critical

- Size of critical application
  - Several loop bounds
Experimental setup

• Number of parallel tasks
  – 1 - 6 cores running less critical

• Size of critical application
  – Several loop bounds

• Critical deadlines
  – Tight (close to WCET_{iso})
  – More relaxed
Experimental setup

- Number of parallel tasks
  - 1 - 6 cores running less critical

- Size of critical application
  - Several loop bounds

- Critical deadlines
  - Tight (close to $WCET_{iso}$)
  - More relaxed

- Observation points
  - From external nested level
  - To all nested levels
Number of Parallel tasks

WCET (ms) vs Number of parallel tasks

Number of tasks: 2 to 8

WCET (ms) increases with the number of parallel tasks.

At 8 tasks, the WCET is approximately x6.28.
Application size

- Isolated execution
- Max load (6LC)

WCET(ms) vs Loop bounds

- Application size

A. Kritikakou
Gain (application size 32)

\[ Gain = \frac{t - t_{iso}}{t_{iso}} \]
Gain (application size 32)

Gain = \frac{t - t_{iso}}{t_{iso}}

298%
Gain (application size 32)

Gain = \frac{t - t_{iso}}{t_{iso}}

- HP1
- HP2
- HP3

298%

65%
Gain (application size 32)

Gain = \frac{t - t_{iso}}{t_{iso}}

- HP1: 298%
- HP2: 65%
- HP3: 37%
Gain (application size 128)

- HP1: ~480% gain
- HP2: 506% gain
- HP3: 24% gain

Critical Deadline (ms)

Gain vs. Critical Deadline plot showing the performance of different HPs.
Conclusions & Future directions

• **Conclusions:**
  – Design-time:
    • Instrumentation of critical tasks
    • Pre-computation of structure & timing information
  – Run-time:
    • Locally use info to decide the suspension of less critical tasks
    • Globally manage the execution of the less critical tasks
  – Implementation on Texas TMS platform

• **Future directions:**
  – Methodology for position & sampling of observation points
  – Extension to:
    • Several criticality levels
    • Time & Space partitioning
THANK YOU

angeliki.kritikakou@irisa.fr
www.kritikakou.com

QUESTIONS?