Run-time adaptation of task execution in time-critical systems: Challenges and Solutions

Angeliki Kritikakou

Univ Rennes, IRISA, Inria, CNRS

June 2021
Time-critical systems

- **Application domains**
  - Avionics (Fly-by-wire)
  - Automotive (Airbag)
  - Medical (X-Ray)

- **Common characteristics**
  - **Criticality Level**: dual-criticality model
  - **Real-Time constraint**: Hard or soft (no)
  - **Priority**

<table>
<thead>
<tr>
<th>CL</th>
<th>RT</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>HI</td>
<td>5 ms</td>
<td>0</td>
</tr>
<tr>
<td>LO</td>
<td>7 ms</td>
<td>2</td>
</tr>
<tr>
<td>HI</td>
<td>3 ms</td>
<td>1</td>
</tr>
</tbody>
</table>

High criticality tasks require timing-guarantees
Time guarantees

- **Worst-Case Execution Time (WCET)**
  - Variations in the execution time

- **Application**
  - Several execution paths
  - Data-dependent

- **Platform**
  - Dynamic behavior
    - Caches, branch predictors
  - Shared resources
    - Timing interference

*Static WCET is safe, but pessimistic*
Actual execution is typically better than WCET estimation

- The actual execution path is not the worst
- Some memory accesses were actually cache hits
- Less interference occurred in shared resources

Can we reduce WCET pessimism?
Reducing WCET pessimism

### Typical approaches

<table>
<thead>
<tr>
<th>Isolation</th>
<th>Interference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Free</td>
<td>Maintain schedule</td>
</tr>
</tbody>
</table>

**PREM**
e.g. RTS’12

C₀

- Memory phase
- Execution phase
- Idle

C₁

Reducing WCET pessimism

<table>
<thead>
<tr>
<th>Typical approaches</th>
<th>Interference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isolation</td>
<td>Free</td>
<td>Maintain schedule</td>
</tr>
<tr>
<td>Interference-sensitive WCET</td>
<td>Controlled</td>
<td>Maintain schedule or bounds</td>
</tr>
</tbody>
</table>

S. Skalistis and A. Simalatsar, “Near-optimal deployment of dataflow applications on many-core platforms with real-time guarantees”, in DATE, 2017
Reducing WCET pessimism

Typical approaches | Interference | Requirement
---|---|---
Isolation | Free | Maintain schedule
Interference-sensitive WCET | Controlled | Maintain schedule or bounds
Mode switch | Tolerant | Maintain bounds

**Mixed-Critical model**
e.g. RTSS’07, EMSOFT’13

**Requirements for safety impose limitations**
Maintaining schedule: Limitations

- **Time-triggered execution**
  - Fixed start time, computed statically

<table>
<thead>
<tr>
<th>Core</th>
<th>Task</th>
<th>Time</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_0</td>
<td>τ_0</td>
<td>t_0</td>
<td>τ_2</td>
</tr>
<tr>
<td>C_1</td>
<td>τ_1</td>
<td>t_1</td>
<td>τ_3</td>
</tr>
<tr>
<td>C_0</td>
<td>τ_2</td>
<td>t_2</td>
<td>-</td>
</tr>
<tr>
<td>C_1</td>
<td>τ_3</td>
<td>t_3</td>
<td>-</td>
</tr>
</tbody>
</table>

- **Cannot exploit: Early task termination**
  - Idle time

*Run-time adaptation is required*
Run-time adaptation (RA)

- **Control mechanism**
  - Software
  - Hardware

- **Safe adaptation**
  - No deadline miss
  - No additional/Bounded interference

- **Low overhead**
  - Not to negate adaptation gain
RA: isWCET schedule

- Key idea: Preserve partial order of tasks
- Safe: No additional interference

Implementation
- Insert scheduling dependencies
- Encoded with bit vectors
  - Task: Notification, Ready
  - Core: Status

RA: isWCET schedule

- **Key idea:** Preserve partial order of tasks
- **Safe:** No additional interference

**Implementation**
- Insert scheduling dependencies
- Encoded with bit vectors
  - Task: Notification, Ready
  - Core: Status
- Concurrency: Status
  - Protection mechanisms

Evaluation

- **Methods:**
  - isTT: Time-triggered
  - isRA: Run-time Adaptation
    - GLO: Global protection mechanism
    - FG: Distributed protection mechanism

- **TI TMS3206678**
  - 8 DSPs @ 1GHz

Can we do better?
Exploit run-time information

- Available only during actual execution
  - Execution progress
  - Current state of hardware components

- Dynamically improve bounds computed statically
  - Allowed interference from co-runner tasks
  - Upper bounds in resource usage
  - WCET estimations
Exploit run-time information

- Available only during actual execution
  - Execution progress
  - Current state of hardware components

- Dynamically improve bounds computed statically
  - Allowed interference from co-runner tasks
  - Upper bounds in resource usage
  - WCET estimations
Progress: isWCET schedule

- **Key idea:** Relax partial order of tasks
- **Safe:** If extra interference is sustained
- **Implementation**
  - As before: Bit vectors (sch. dependencies)
  - Time Slack: Minimum speed-up in task execution among all cores
  - Relax: WCET of extra interferences ≤ Slack
    - Remove scheduling dependencies

---

Evaluation

- **Methods**
  - isRA-FG
  - isRA-DYN

- **Timing Variability (≤70%)**
  - Paths
  - Cache

- **TI TMS3206678**
  - 8 DSPs @ 1GHz
Progress: WCET estimation

- Key idea: Compute the Remaining WCET (RWCET)
  - WCET of the code that has not executed yet

- Safe: Removing WCET of executed part

- Implementation
  - Insert monitoring points
  - Compute partial WCET
    - Static analysis
    - Measurement-based
  - Compute RWCET based on partial WCET

Instrument source code

```c
even = 0;
odd = 0;
if (iso==0) RTC(a);
for (i=0 ; i<N ; i++) {
   if (i%2 == 0)
       even++;
   else
       odd++;
}
```

Assembly

```
add r3,r0,r0 ...
```

**RWCET: Mode switch**

- **Max load mode:**
  - HC tasks
  - LC tasks
    - Interference
    - WCET>D

- **Isolation mode:**
  - Only HC tasks
  - If time, LC tasks

- **Mode switch:**
  - Max load mode
  - If risk for HC tasks
    - Isolation mode

**Safety condition**
RWCET: Safety condition

Monitoring point

Run-time control

ET

RWCET

Deadline

Continue
RWCET: Safety condition

Monitoring point

Run-time control

ET

RWCET

ET

Deadline

Switch NOW
Results

- **Methods:**
  - isolation: Only HC tasks, if time LC tasks
  - Mode-switch: RWCET

- **Workload:**
  - 2 cores: HC tasks
  - 6 cores: LC tasks

- **TI TMS3206678**
  - 8 DSPs @ 1GHz
Conclusions & Further opportunities

- **WCET pessimism**

- **Run-time adaptation approaches:**
  - Execution progress
    - Interference-sensitive schedule
    - WCET estimation
  - Software

- **Hardware mechanisms for run-time adaptation**

- **Approaches**
  - Take into account state of hardware components
  - Combine with scheduling techniques
Thank You

Questions?

In any case, feel free to contact me: angeliki.kritikakou@irisa.fr