Memory Bandwidth Regulation for Multiframe Task Sets

Muhammad Ali Awan, Pedro F. Souto, Konstantinos Bletsas, Benny Akesson, and Eduardo Tovar
Outline

- Motivation and system model
- Background
- Our approach
- Computation time
- Heuristics
- Evaluation
- Conclusions
Motivation

- Resource Utilization
- Timing Analysis

Multi-core Platform

- App 1
- App 2
- App 3
- App 4

- Computation
- Energy
- Weight
- Cost
- Scalability

CISTER – Research Centre in Real-Time & Embedded Computing Systems
Task Model

- Multiframe task-model
- Preemptive fixed priority scheduling
- Independent sporadic tasks
- No-migration
- Deadline monotonic

Super frame

Frame 1 Frame 2 Frame 3 Frame 4
Platform Model

- Identical multicore platform
- Partitioned/private last-level cache
- Prefetchers and speculative units are disabled
- Round-robin bus arbitration (interconnect + memory controller)
- Performance monitoring counters
- Constant memory access time
- Multiple outstanding memory requests
- CPU computation and memory accesses do not overlap in time

Core 1 → Cache → Bus Arbiter → Main Memory
Core 2 → Cache → Bus Arbiter → Main Memory
Core 3 → Cache → Bus Arbiter → Main Memory
Core 4 → Cache → Bus Arbiter → Main Memory

WCET

CISTER – Research Centre in Real-Time & Embedded Computing Systems
Memory Access Regulation Model

Memory accesses regulated by MemGuard

- Memory budget assigned
- Budget exhausted
- Contention stall
- Regulation period
- Regulation stall
- Uneven memory bandwidth across cores
Memory-aware schedulability analysis

Schedulability analysis (Yao’s approach)

Stall analysis

- $C^e$
- $C^m$
- Memory access budget
- Upper bound on Stall

Stall Analysis

WCRT = Stall + Interference of higher priority tasks + ($C^e + C^m$)

Composite Task

- Jobs of higher or equal priority tasks that can preempt a task under analysis
- Task under analysis

Standard response time analysis demand in previous iteration

WCET of a task under analysis
Schedulability analysis of Multiframe tasks

Maximum cumulative execution requirement (Baruah’s approach)

For two frames = Max (a, b, c, d)

For three frames = Max (w, x, y, z)

Compute number of jobs for each higher priority task

Sum their maximum cumulative execution requirement
Why SOTA solutions do not work

Task parameters
- Period of 10
- Three frames

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Total WCET</th>
<th>Total memory accesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S^1$</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>$S^2$</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>$S^3$</td>
<td>9</td>
<td>4</td>
</tr>
</tbody>
</table>

WCRT = Stall + Interference of higher priority tasks + ($C^e + C^m$)

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Total WCET</th>
<th>Total memory accesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S^1$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$S^2$</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$S^3$</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>
Main contributions

- Worst-case memory stall for multiframe task-model
- Stall-aware schedulability analysis for multiframe task-model
- Five memory bandwidth and task-to-core allocation heuristics
Our proposed schedulability analysis

WCRT^{F_1} = \text{Stall} + \text{Interference of higher priority tasks} + (C_{e,F_1} + C_{m,F_1})

WCRT^{F_2} = \text{Stall} + \text{Interference of higher priority tasks} + (C_{e,F_2} + C_{m,F_2})

WCRT^{F_n} = \text{Stall} + \text{Interference of higher priority tasks} + (C_{e,F_n} + C_{m,F_n})

\text{WCRT} = \text{Max}(WCRT^{F_1}, WCRT^{F_2}, \ldots, WCRT^{F_n})
**Example**

### Task A
(3 frames, Period = 20)

<table>
<thead>
<tr>
<th>S1,A</th>
<th>S2,A</th>
<th>S3,A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ t = 25 \]

- \( S^{1,A} = \{F^{1,A}, F^{2,A}\} \)
- \( S^{2,A} = \{F^{2,A}, F^{3,A}\} \)
- \( S^{3,A} = \{F^{3,A}, F^{1,A}\} \)

**Cartesian product of sequences =**

\[ \{ (S^{1,A}, S^{1,B}), (S^{1,A}, S^{2,B}), (S^{1,A}, S^{3,B}), (S^{1,A}, S^{4,B}), (S^{2,A}, S^{1,B}), (S^{2,A}, S^{2,B}), (S^{2,A}, S^{3,B}), (S^{2,A}, S^{4,B}), (S^{3,A}, S^{1,B}), (S^{3,A}, S^{2,B}), (S^{3,A}, S^{3,B}), (S^{3,A}, S^{4,B}) \} \]

Or

\[ \{ \Theta^1, \Theta^2, \Theta^3, \Theta^4, \Theta^5, \Theta^6, \Theta^7, \Theta^8, \Theta^9, \Theta^{10}, \Theta^{11}, \Theta^{12} \} \]

In \((k+1)\)th iteration, for each tuple \(\Theta^x\) compute stall and interference

\[ WCRT^{Fx}_{(k+1)} = \text{Stall} (\Theta^x) + \text{Interference of higher priority tasks} (\Theta^x) + (C^{e,Fx} + C^{m,Fx}), \text{ where } x = \{1, 2, ..., 12\} \]

Select tuple that gives the maximum memory regulation stall and interference.
Reducing computational cost

**No. of Frames**
For any two frames
\[ F^i \geq F^j, \text{ iff } \]
\[ C^{e,i} \geq C^{e,j} \quad \text{and} \quad C^{m,i} \geq C^{m,j} \]

Or
\[ (C^{e,i}, C^{m,i}) \geq (C^{e,j}, C^{m,j}) \]

It is sufficient to check only WCRT of \( F^i \)

**No. of Sequences**
For any two sequences
\[ S^x \geq S^y, \text{ iff } \]
\[ C^{e,x} \geq C^{e,y} \quad \text{and} \quad C^{m,x} \geq C^{m,y} \]

Or
\[ (C^{e,x}, C^{m,x}) \geq (C^{e,y}, C^{m,y}) \]

It is sufficient to check only sequence \( S^x \)

**No. of tuples**
For any two tuple
\[ \Theta^x \geq \Theta^y, \text{ iff } \]
\[ C^{e,\Theta^x} \geq C^{e,\Theta^y} \quad \text{and} \quad C^{m,\Theta^x} \geq C^{m,\Theta^y} \]

Or
\[ (C^{e,\Theta^x}, C^{m,\Theta^x}) \geq (C^{e,\Theta^y}, C^{m,\Theta^y}) \]

It is sufficient to check only tuple \( \Theta^x \)
Tightness vs computational cost

Task parameters
- Period of 10
- Three frames

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Total WCET</th>
<th>Total memory accesses</th>
<th>Total CPU Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>S^1</td>
<td>7</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>S^2</td>
<td>8</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>S^3</td>
<td>9</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Sequence: S^{123}
Total WCET: 9
Total memory accesses: 5
Total CPU Computation: 5
## Implementation details

For each task

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1,1</td>
<td>1,2</td>
<td>1,3</td>
<td>...</td>
<td>1,N</td>
<td></td>
</tr>
<tr>
<td>2,1</td>
<td>2,2</td>
<td>2,3</td>
<td>...</td>
<td>2,N</td>
<td></td>
</tr>
<tr>
<td>3,1</td>
<td>3,2</td>
<td>3,3</td>
<td>...</td>
<td>3,N</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N,1</td>
<td>N,2</td>
<td>N,3</td>
<td>...</td>
<td>N,N</td>
<td></td>
</tr>
</tbody>
</table>

N frames

(2,3) → 2 jobs starting from 3rd frame
Task-to-core and Memory Bandwidth allocation

Five Heuristics

Even First-Fit
- Each Core has equal memory bandwidth share
- First-fit bin packing for task-to-core analysis

Uneven First-Fit
- Initially each Core has equal memory bandwidth share
- Trim-off memory bandwidth, if tasks are not schedulable with equal memory bandwidth
- Use this trimmed bandwidth to schedule remaining tasks

Memory-fit
- Assign a task to a core that requires minimum memory bandwidth

Priority assignment: Deadline monotonic

Memory density worst-fit
- Sort cores in non-increasing order of energy density
- Assign task to a core with that gives minimum increase in memory density

Total density worst-fit
- Similar to memory density worst-fit, except it uses total density instead of memory density
Experimental Setup

- Utilization: UUnifast-discard algorithm
- Inter-arrival time: Log-uniform distribution (10 ms - 1 s)
- WCET of first frame = inter-arrival time × Utilization
- Implicit deadlines (though algorithm works for constrained deadlines)
- Number of frames: Selected randomly
- WCET of other frames
  - Randomly selected with log-uniform distribution
  - Between user define value and WCET of first frame
- Memory accesses of each frame
  - Selected randomly
  - Memory access time is 40 nsec
  - Regulation period length is 100 msec
- 1000 random task-sets per set point
Results

MF-TA achieves up to 85% improvements in terms of schedulability success ratio.

MF-FA is 11x times faster than MF-TA with 3.7% decreases in terms of schedulability success ratio.
Conclusions

- Stall-aware schedulability analysis for multiframe task sets on multicore platforms
- Provided techniques to reduce the computation time
- Has possibility to trade-off tightness vs computation time
- Improved schedulability success ratio up to 85% when compared to frame-agonistic stall-aware analysis
- Achieved 11-fold speed up with 3.7% loss in schedulability
- Proposed five memory bandwidth and task-to-core allocation heuristics
Questions ?
Main idea

Multiframe Task-model  Multicore platforms  Memory regulation

Our work