Analysis of Dynamic Memory Bandwidth Regulation in Multi-core Real-Time Systems

Ankit Agrawal¹, Renato Mancuso², Rodolfo Pellizzoni³ & Gerhard Fohler¹
Certified single-core avionics application

Level 1: Workload

Workload’s mean mem. BW demand over time
(measured)
[Agrawal-ECRTS’17]
Application’s Memory BW demand varies over time

Certified single-core avionics application

Level 1: Workload

Workload’s mean mem. BW demand over time (measured) [Agrawal-ECRTS’17]

Level 2: Intra-Workload

Workload’s instantaneous mem. BW demand over time (illustration)
Application’s Memory BW demand varies over time

Certified single-core avionics application

Level 1: Workload

Workload’s mean mem. BW demand over time
(measured)
[Agrawal-ECRTS’17]

Level 2: Intra-Workload

Workload’s instantaneous mem. BW demand over time
(illustration)

Start: Lots of reads
Application’s Memory BW demand varies over time

Certified single-core avionics application

**Level 1: Workload**

Workload’s mean mem. BW demand over time
(measured)
[Agrawal-ECRTS’17]

**Level 2: Intra-Workload**

Workload’s instantaneous mem. BW demand over time
(illustration)
Application’s Memory BW demand varies over time

Level 1: Workload

Workload’s **mean** mem. BW demand over time
(measured) [Agrawal-ECRTS’17]

Level 2: Intra-Workload

Workload’s **instantaneous** mem. BW demand over time
(illustration)
State-of-the-Art mainly CPU-level scheduling
State-of-the-Art mainly CPU-level scheduling

Static memory BW partitioning

Unsuited to match assigned BW to workload-level BW demand

[Mancuso-ECRTS’17]
[Mancuso-ECRTS’15]
[Yao-TC’16]
State-of-the-Art mainly CPU-level scheduling

Static memory BW partitioning

[ statically partitioned memory to balance BW demand across different workload levels]

Unsuited to match assigned BW to workload-level BW demand

Synchronized workload-level scheduling

[ synchronously scheduled tasks to meet BW demand]

Unsuited to match assigned BW to intra-workload level BW demand

References:

[Mancuso-ECRTS’17]
[Mancuso-ECRTS’15]
[Yao-TC’16]

[Nowotsch-RTNS’13]
[Nowotsch-ECRTS’14]
State-of-the-Art mainly CPU-level scheduling

- Well-known: memory BW is performance bottleneck in multi-cores
- State-of-the-art: mainly limited to CPU-level scheduling
State-of-the-Art mainly CPU-level scheduling

Static memory BW partitioning
- [Mancuso-ECRTS’17]
- [Mancuso-ECRTS’15]
- [Yao-TC’16]

Synchronized workload-level scheduling
- [Nowotsch-RTNS’13]
- [Nowotsch-ECRTS’14]

- Well–known: memory BW is performance bottleneck in multi-cores
- State-of-the-art: mainly limited to CPU-level scheduling

Why not schedule memory BW over time
i.e., assign memory BW to cores over time?
Why schedule memory BW over time?

- **dynamic management**; static partitioning known to be sub-optimal
Why schedule memory BW over time?

- **dynamic management**; static partitioning known to be sub-optimal
- enables **tailoring BW supply** to application’s needs

... can **increase** system **schedulability**
Why schedule memory BW over time?
- dynamic management; static partitioning known to be sub-optimal
- enables tailoring BW supply to application’s needs
  ... can increase system schedulability

Fundamental Challenge?
- Changing memory BW assigned to a workload
Why schedule memory BW over time?
- **dynamic management**; static partitioning known to be sub-optimal
- enables **tailoring BW supply** to application’s needs
  ... can **increase** system **schedulability**

**Fundamental Challenge?**
- **Changing memory BW** assigned to a workload
  - **affects** its **WCET** as its worst-case memory request pattern can change
Why schedule memory BW over time?

- **dynamic management**; static partitioning known to be sub-optimal
- enables **tailoring BW supply** to application’s needs
  ... can increase system **schedulability**

Fundamental Challenge?

- **Changing memory BW** assigned to a workload
  - affects its **WCET** as its worst-case memory request pattern can change
  - affects **WCET** of **co-executing workloads** as memory BW is shared
Vision and This Work

Classical Scheduling

- CPUs

Vision: Co-Scheduling

- CPUs
- Memory

Multi-core
Vision and This Work

This work

WCET / Response Time Analysis when...
This work

WCET / Response Time Analysis when

- memory BW is subject to scheduling and
Vision and This Work

**Classical Scheduling**

- CPUs

**Vision: Co-Scheduling**

- CPUs
- Memory

---

**This work**

WCET / Response Time Analysis when

- memory BW is subject to scheduling and
- memory BW allocation over time to cores is known
Vision and This Work

This work

**WCET / Response Time Analysis when**
- memory BW is subject to scheduling and
- memory BW allocation over time to cores is known
- workload to CPU allocation and workload priority order is known
Multi-core Model

\[ \text{Core } c_1, \text{ Core } c_2, \ldots, \text{ Core } c_m \]

Number of cores: \( m \)
Multi-core Model

Core $c_1$  Core $c_2$  ...  Core $c_m$

$m$ Number of cores

Memory
Multi-core Model

Core $c_1$  Core $c_2$  ...  Core $c_m$

Number of cores $m$

Memory

Round-Robin
Multi-core Model

- Core $c_1$
- Core $c_2$
- ... $\ldots$
- Core $c_m$

Number of cores $m$

Max. size of memory request $L_{size}$
Model & Assumptions

Multi-core Model

- Core $c_1$
- Core $c_2$
- ... (x)... (y)
- Core $c_m$

- $m$: Number of cores
- $L_{\text{size}}$: Max. size of memory request
- $L_{\max}$: Max. time for memory request & max. delay from any other core
Memory Regulation Model
Memory Regulation Model

\[ P = \text{regulation period} \]
Memory Regulation Model

\[ P = \text{regulation period} \]

Max. cumulative memory requests in \( P \)

\[ \frac{P}{L_{\text{max}}} = Q_{\text{tot}} \]
Model & Assumptions

Memory Regulation Model

\[ P = \text{regulation period} \]

Max. cumulative memory requests in \( P \)

\[ \frac{P}{L_{\text{max}}} = Q_{\text{tot}} \]

Memory Budget Assignment

\[ \sum q_i \leq Q_{\text{tot}} \]
Memory Regulation Model

\[ P = \text{regulation period} \]

Max. cumulative memory requests in \( P \)

\[ \frac{P}{L_{\text{max}}} = Q_{\text{tot}} \]

Memory Budget Assignment

\[ Q = \{ q_1, q_2, q_3, q_4 \} \]
Model & Assumptions

Memory Regulation Model

\[ Q_{tot} \]

Memory Regulation Model

MemGuard *

BW Regulators

\[ q_1 + q_2 + q_3 + q_4 \leq Q_{tot} \]

**Model & Assumptions**

**Memory Regulation Model**

*MemGuard*  Periodically regulate each core’s memory bandwidth

*BW Regulators*  *from [Yun-RTAS’13]*

\[
q_1 + q_2 + q_3 + q_4 \leq Q_{tot}
\]

Model & Assumptions

Memory Regulation Model

*MemGuard* Periodically regulate each core’s memory bandwidth

*BW Regulators*  

\[ Q = q_1, q_2, q_3, q_4 \leq Q_{tot} \]

* from [Yun-RTAS’13]

\[ q_2 = 2 \]

Example:

\[ Q = \{1,2,3,4\} \]

**Model & Assumptions**

### Memory Regulation Model

**MemGuard**

*Periodically regulate each core’s memory bandwidth*

*from [Yun-RTAS’13]*

**BW Regulators**

Execution =  
DRAM request =

\[ P = \text{regulation period} \]

\[ Q = \{1, 2, 3, 4\} \]

**e.g.:**

\[ q_2 = 2 \]

Memory Regulation Model

**MemGuard** *


**BW Regulators** *from [Yun-RTAS’13]*

Periodically regulate each core’s memory bandwidth

Execution = ![Yellow bar]

DRAM request = ![Red bar]

Budget tracked with performance counters

<table>
<thead>
<tr>
<th>Core</th>
<th>Budget</th>
<th>DRAM request</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>q1</td>
<td>MG</td>
</tr>
<tr>
<td>c2</td>
<td>q2</td>
<td>MG</td>
</tr>
<tr>
<td>c3</td>
<td>q3</td>
<td>MG</td>
</tr>
<tr>
<td>c4</td>
<td>q4</td>
<td>MG</td>
</tr>
</tbody>
</table>

\[ Q_{tot} \leq Q \]

e.g.:

\[ Q = \{1, 2, 3, 4\} \]

\[ q_2 = 2 \]

Memory Regulation Model

**MemGuard**

*BW Regulators*

Periodically regulate each core’s memory bandwidth

*from [Yun-RTAS’13]*

Budget tracked with performance counters

\[ Q = \{1,2,3,4\} \]

\[ q_2 = 2 \]

Memory Regulation Model

*MemGuard* from [Yun-RTAS’13]

Periodically regulate each core’s memory bandwidth

Execution = DRAM request = Budget tracked with performance counters

\[ P = \text{regulation period} \]

\[ Q = \{1, 2, 3, 4\} \]

\[ q_2 = 2 \]

Model & Assumptions

Workload Model
Workload Model

Application comprises 1 or more workloads
Workload Model

Application comprises 1 or more workloads

- partitioned across m cores
Workload Model

Application comprises 1 or more workloads

- partitioned across m cores
- Workload characterization:
Workload Model

Application comprises 1 or more workloads
  ▪ partitioned across m cores
  ▪ Workload characterization:

\[ E \quad \text{Pure execution time} \]
in units of \( L_{\text{max}} \)
Workload Model

Application comprises 1 or more workloads
- partitioned across m cores
- Workload characterization:

\[ E \text{ Pure execution time in units of } L_{\text{max}} \]

\[ \mu \text{ Max. no. of memory requests} \]
Workload Model

Application comprises 1 or more workloads
- partitioned across m cores
- Workload characterization:
  - Pure execution time $E$
in units of $L_{max}$
  - Max. no. of memory requests $\mu$
**Model & Assumptions**

**Workload Model**

Application comprises 1 or more workloads
- partitioned across m cores
- Workload characterization:

- Pure execution time \( E \) in units of \( L_{\text{max}} \)
- Max. no. of memory requests \( \mu \)

Span: Computed Maximum time required to meet a workload’s \( E \) and \( \mu \) demand.
Stall: Worst-case Patterns -- core $c_4$

Recall:

$$\vec{Q} = \{q_1, q_2, q_3, q_4\}$$

$$P = \frac{Q_{tot}}{L_{max}}$$
Stall: Worst-case Patterns -- core \( c_4 \)

\[
\mathbf{Q} = \{q_1, q_2, q_3, q_4\}
\]

\[
P = \frac{Q}{L_{\text{max}}} = \frac{Q_{\text{tot}}}{10} = \frac{1 + 2 + 3 + 4}{10}
\]

\( e.g.: \vec{Q} = \{1, 2, 3, 4\} \)
Stall: Worst-case Patterns -- core \( c_4 \)

\[
\overrightarrow{Q} = \{q_1, q_2, q_3, q_4\}
\]

\[
\overrightarrow{L_{\text{max}}} = \overrightarrow{Q_{\text{tot}}} = 10 = 1 + 2 + 3 + 4
\]

\[ e.g.: \overrightarrow{Q} = \{1, 2, 3, 4\} \]

- Core \( c_1 \) stalled after 1 memory request
- Core \( c_2 \) stalled after 2 memory requests
- Core \( c_3 \) stalled after 3 memory requests
- Core \( c_4 \) stalled after 4 memory requests
Stall: Worst-case Patterns -- core $c_4$

\[ Q = \{q_1, q_2, q_3, q_4\} \]

\[ P \]

\[ L_{max} = Q_{tot} = 10 = 1 + 2 + 3 + 4 \]

e.g.: \[ Q = \{1, 2, 3, 4\} \]

- Core $c_1$: stalled after 1 memory request
- Core $c_2$: stalled after 2 memory requests
- Core $c_3$: stalled after 3 memory requests
- Core $c_4$: stalled after 4 memory requests
Stall: Worst-case Patterns -- core $c_4$

\[ \overrightarrow{Q} = \{q_1, q_2, q_3, q_4\} \]

\[ \overrightarrow{L_{max}} = \overrightarrow{Q_{tot}} = 10 = 1 + 2 + 3 + 4 \]

\[ e.g.: \overrightarrow{Q} = \{1, 2, 3, 4\} \]

- Core $c_1$ stalled after 1 memory request
- Core $c_2$ stalled after 2 memory requests
- Core $c_3$ stalled after 3 memory requests
- Core $c_4$ stalled after 4 memory requests
Stall: Worst-case Patterns -- core $c_4$

\[
\begin{align*}
\mathbf{Q} &= \{q_1, q_2, q_3, q_4\} \\
\mathbf{P} &= \mathbf{Q}_{\text{tot}} = \frac{10}{L_{\text{max}}} = 1 + 2 + 3 + 4 \\
\text{e.g.: } \mathbf{Q} &= \{1, 2, 3, 4\}
\end{align*}
\]

- Core $c_1$ stalled after 1 memory request
- Core $c_2$ stalled after 2 memory requests
- Core $c_3$ stalled after 3 memory requests
- Core $c_4$ stalled after 4 memory requests
Stall: Worst-case Patterns -- core $c_4$

\[ \mathbf{Q} = \{q_1, q_2, q_3, q_4\} \]

\[ \mathbf{P} = \frac{\mathbf{Q}_{tot}}{L_{max}} = \frac{10 = 1 + 2 + 3 + 4}{L_{max}} \]

\[ \mathbf{Q} = \{1, 2, 3, 4\} \]

Core $c_1$ stalled after 1 memory request
Core $c_2$ stalled after 2 memory requests
Core $c_3$ stalled after 3 memory requests
Core $c_4$ stalled after 4 memory requests
Stall: Worst-case Patterns -- core $c_4$

\[
\begin{align*}
\overrightarrow{Q} &= \{q_1, q_2, q_3, q_4\} \\
\overrightarrow{P} &= \frac{\overrightarrow{Q}}{L_{\text{max}}} = \overrightarrow{Q_{\text{tot}}} = 10 = 1 + 2 + 3 + 4
\end{align*}
\]

**Example:** $Q = \{1,2,3,4\}$

- Core $c_1$ stalled after 1 memory request
- Core $c_2$ stalled after 2 memory requests
- Core $c_3$ stalled after 3 memory requests
- Core $c_4$ stalled after 4 memory requests
Stall: Worst-case Patterns -- core $c_4$

\[ \vec{Q} = \{q_1, q_2, q_3, q_4\} \]

\[ \frac{P}{L_{\text{max}}} = Q_{\text{tot}} = 10 = 1 + 2 + 3 + 4 \]

\[ \text{e.g.: } Q = \{1, 2, 3, 4\} \]

- Core $c_1$: stalled after 1 memory request
- Core $c_2$: stalled after 2 memory requests
- Core $c_3$: stalled after 3 memory requests
- Core $c_4$: stalled after 4 memory requests
Curve: Max. Memory Request Time -- core $c_4$

$Q = \{1,2,3,4\}$

Core $c_4$ stalled after 4 memory requests

$e.g.: Q = \{1,2,3,4\}$
Curve: Max. Memory Request Time -- core $c_4$

**Example:** $Q = \{1,2,3,4\}$

Core $c_4$ stalled after 4 memory requests.
e.g.: $Q = \{1,2,3,4\}$

Core $c_4$ stalled after 4 memory requests
Curve: Max. Memory Request Time -- core $c_4$

*e.g.:* $Q = \{1, 2, 3, 4\}$

Core $c_4$ stalled after 4 memory requests
**Curve: Max. Memory Request Time -- core c₄**

**Example:** $Q = \{1,2,3,4\}$

Core c₄ stalled after 4 memory requests
**Curve: Max. Memory Request Time -- core $c_4$**

**e.g.:** $Q = \{1,2,3,4\}$

Core $c_4$ stalled after 4 memory requests

- Memory request = $L_{\text{max}}$
- Core # memory = $L_{\text{max}}$
- Execution = $L_{\text{max}}$

$P$ is the set of core states:
- $c_1$, $c_2$, $c_3$, $c_2$, $c_3$, $c_3$, $c_3$

$r \in [0, q_4] = [0,4]$

Time units (in $L_{\text{max}}$)

Stall = 0
Stall = 1
Stall = 2
Stall = 3
Upcoming

- Span Analysis under static uneven budget
WCET / Span Analyses

Upcoming

- Span Analysis under static uneven budget

\[ Q = \{q_1, q_2, q_3, q_4\} \]

\[ \overrightarrow{Q} = \{q_1, q_2, q_3, q_4\} \]

\[ \text{e.g.: } Q = \{1, 2, 3, 4\} \]

\[ L_{\max} = 1; \ P = Q_{\text{tot}} = 10 \]
Upcoming

- Span Analysis under static uneven budget

\[ \overrightarrow{Q} = \{q_1, q_2, q_3, q_4\} \]

\[ e.g.: \overrightarrow{Q} = \{1, 2, 3, 4\} \]

\[ L_{\text{max}} = 1; \quad P = Q_{\text{tot}} = 10 \]

Consider

Workload on core \( c_4 \)

\[ E = 10 \quad \mu = 4 \]

\[ q_4 = 4 \]
Upcoming

- Span Analysis under static uneven budget

\[
\overrightarrow{Q} = \{q_1, q_2, q_3, q_4\}
\]

\text{e.g.: } \overrightarrow{Q} = \{1, 2, 3, 4\}

\[L_{\text{max}} = 1; \quad P = Q_{\text{tot}} = 10\]

Consider

Workload on core \(c_4\)

\(E = 10\) \quad \(\mu = 4\)

\(q_4 = 4\)

\text{Ques.: Calculate span } W \text{ of a workload with } E = 10 \text{ and } \mu = 4 \text{ on core } c_4 \text{ with } q_4 = 4 \text{ under static } \overrightarrow{Q} = \{1, 2, 3, 4\}\]
Ques.: Calculate Span $W$ of workload $E = 10$ and $\mu = 4$ on core $c_4$ under $Q = \{1,2,3,4\}$ and $P = 10$.
Ques.: Calculate Span $W$ of workload $E = 10$ and $\mu = 4$ on core $c_4$ under $Q = \{1, 2, 3, 4\}$ and $P = 10$

$W_{\text{min}} = E + \mu$

$= 10 + 4$

$= 14$
Ques.: Calculate Span $W$ of workload $E = 10$ and $\mu = 4$ on core $c_4$ under $Q = \{1, 2, 3, 4\}$ and $P = 10$. 

$W_{\text{min}} = 14$
Ques.: Calculate Span $W$ of workload $E = 10$ and $\mu = 4$ on core $c_4$ under $Q = \{1,2,3,4\}$ and $P = 10$.

$W_{min} = 14$

Time units (in $L_{max}$)
Ques.: Calculate **Span** $W$ of workload $E = 10$ and $\mu = 4$ on core $c_4$ under $Q = \{1,2,3,4\}$ and $P = 10$

$W_{min} = 14$

**Case:** All mem. requests at begin
Ques.: Calculate Span $W$ of workload $E = 10$ and $\mu = 4$ on core $c_4$ under $Q = \{1,2,3,4\}$ and $P = 10$

$W_{min} = 14$

All mem. requests at begin

Span $W = 20$
Span Analysis under Static Budget

**Ques.:** Calculate **Span W** of workload $E = 10$ and $\mu = 4$ on core $c_4$ under $Q = \{1,2,3,4\}$ and $P = 10$

$W_{\min} = 14$

All mem. requests at begin

Span $W = 20$

Span is the maximum time to met $E$ and $\mu$ demand
**Ques.:** Calculate Span $W$ of workload $E = 10$ and $\mu = 4$ on core $c_4$ under $Q = \{1,2,3,4\}$ and $P = 10$.

$W_{min} = 14$

Span $W = 20$
Ques.: Calculate Span \( W \) of workload \( E = 10 \) and \( \mu = 4 \) on core \( c_4 \) under \( Q = \{1,2,3,4\} \) and \( P = 10 \)

\[ W_{\text{min}} = 14 \]

Span \( W = 20 \times \)

Span \( W = ? \)
Ques.: Calculate Span \( W \) of workload \( E = 10 \) and \( \mu = 4 \) on core \( c_4 \) under \( Q = \{1,2,3,4\} \) and \( P = 10 \)

\[ W_{\text{min}} = 14 \]

Span \( W = 2 \) \( \times \)

Span \( W = ? \)

**Key intuition:** Distribute requests along curve in decreasing stall order.
Span Analysis under Static Budget

**Ques.:** Calculate Span $W$ of workload $E = 10$ and $\mu = 4$ on core $c_4$ under $W_{min} = 14$ and $Q = \{1,2,3,4\}$ and $P = 10$.

**Span:**
- $W = 2$ (marked with an 'X')
- $W = ?$

**Time units (in $L_{max}$):**

**Ques.:** Calculate Span $W$ of workload $E = 10$ and $\mu = 4$ on core $c_4$ under $Q = \{1,2,3,4\}$ and $P = 10$.

**Span:**
- $W = 2$ (marked with an 'X')
- $W = ?$
Ques.: Calculate Span $W$ of workload $E = 10$ and $\mu = 4$ on core $c_4$ under $Q = \{1, 2, 3, 4\}$ and $P = 10$

$W_{\text{min}} = 14$

Span $W = 20$

Span $W = ?$
Ques.: Calculate Span $W$ of workload $E = 10$ and $\mu = 4$ on core $c_4$ under $Q = \{1,2,3,4\}$ and $P = 10$

$W_{min} = 14$

$W = 2\times$

$W = ?$

$Span = 2$
Ques.: Calculate Span \( W \) of workload \( E = 10 \) and \( \mu = 4 \) on core \( c_4 \) under \( Q = \{1,2,3,4\} \) and \( P = 10 \).
Ques.: Calculate Span $W$ of workload $E = 10$ and $\mu = 4$ on core $c_4$ under $Q = \{1,2,3,4\}$ and $P = 10$

$W_{\text{min}} = 14$

Span
$W = 2$

Span
$W = ?$

$P$

Memory request = $L_{\text{max}}$

Core # memory = $L_{\text{max}}$

Execution = $L_{\text{max}}$
Ques.: Calculate Span $W$ of workload $E = 10$ and $\mu = 4$ on core $c_4$ under $Q = \{1, 2, 3, 4\}$ and $P = 10$.

$W_{min} = 14$

Span $W = 20$ ✗

Span $W = 25$ ✔

$Span W = 25$ ✔
Span Analysis under Dynamic Budgets

system undergoes, multiple modes
Span Analysis under Dynamic Budgets

system undergoes, multiple modes

\[
\begin{align*}
Q_1 &= \{1,2,3,4\} \\
\end{align*}
\]
Span Analysis under Dynamic Budgets

The system undergoes multiple modes.

Initial mode:
- $Q_1 = \{1, 2, 3, 4\}$

Mode change:
- $Q_2 = \{1, 4, 3, 2\}$
Span Analysis under Dynamic Budgets

The system undergoes multiple modes of operation. The core configurations are as follows:

- **Initial mode**:
  - Core \( c_1 \)
  - Core \( c_2 \)
  - Core \( c_3 \)
  - Core \( c_4 \)

- **Mode change**:
  - Core \( c_1 \) becomes active
  - Core \( c_2 \) becomes active
  - Core \( c_3 \) becomes active
  - Core \( c_4 \) becomes active

- **Mode change**:
  - Core \( c_1 \) becomes active
  - Core \( c_2 \) becomes active
  - Core \( c_3 \) becomes active
  - Core \( c_4 \) becomes active

The core configurations are represented by vectors:

- \( \tilde{Q}_1 = \{1,2,3,4\} \)
- \( \tilde{Q}_2 = \{1,4,3,2\} \)
- \( \tilde{Q}_3 = \{1,2,3,4\} \)
Span Analysis under Dynamic Budgets

The system undergoes multiple modes.

- **Initial mode**: \( \overrightarrow{Q_1} = \{1, 2, 3, 4\} \)
- **Mode change**: \( \overrightarrow{Q_2} = \{1, 4, 3, 2\} \)
- **Mode change**: \( \overrightarrow{Q_3} = \{1, 2, 3, 4\} \)

Span: \( W(\overrightarrow{Q_1} \rightarrow \overrightarrow{Q_2} \rightarrow \overrightarrow{Q_3} \rightarrow \ldots) = ? \)
Span Analysis under Dynamic Budgets

Core # memory = $L_{max}$

Memory request = $L_{max}$

Core # memory = $L_{max}$

Execution = $L_{max}$
Span Analysis under Dynamic Budgets

Core # memory = $L_{\text{max}}$

Execution = $L_{\text{max}}$

Memory request = $L_{\text{max}}$

Regulation stall = $L_{\text{max}}$
Ques.: Calculate Span $W$ of workload $E = 10$ and $\mu = 4$ on core $c_4$ and $P = 10$ under $\vec{Q}_1, \vec{Q}_2$ and $\vec{Q}_3$.
Ques.: Calculate Span $W$ of workload $E = 10$ and $\mu = 4$ on core $c_4$ and $P = 10$ under $\vec{Q}_1, \vec{Q}_2$ and $\vec{Q}_3$. 

**Core # memory** $= L_{\text{max}}$

**Execution** $= L_{\text{max}}$

**Memory request** $= L_{\text{max}}$

**Regulation stall** $= L_{\text{max}}$
Ques.: Calculate **Span W** of workload $E = 10$ and $\mu = 4$ on core $c_4$ and $P = 10$ under $\vec{Q}_1, \vec{Q}_2$ and $\vec{Q}_3$ under $\vec{Q}_1 = \{1, 2, 3, 4\}$, $\vec{Q}_2 = \{1, 2, 3, 4\}$ and $\vec{Q}_3 = \{1, 2, 3, 4\}$. 

Core # memory = $L_{\text{max}}$ 

Execution = $L_{\text{max}}$ 

Regulation stall = $L_{\text{max}}$ 

Memory request = $L_{\text{max}}$
Ques.: Calculate **Span W** of workload $E = 10$ and $\mu = 4$ on core $c_4$ and $P = 10$ under $\overrightarrow{Q_1}, \overrightarrow{Q_2}$ and $\overrightarrow{Q_3}$

$\overrightarrow{Q_1} = \{1,2,3,4\}$  
$\overrightarrow{Q_2} = \{1,4,3,2\}$  
$\overrightarrow{Q_3} = \{1,2,3,4\}$
Ques.: Calculate Span $W$ of workload $E = 10$ and $\mu = 4$ on core $c_4$ and $P = 10$ under $\overrightarrow{Q_1}, \overrightarrow{Q_2}$ and $\overrightarrow{Q_3}$

$\overrightarrow{Q_1} = \{1,2,3,4\}$
$\overrightarrow{Q_2} = \{1,4,3,2\}$
$\overrightarrow{Q_3} = \{1,2,3,4\}$
Ques.: Calculate Span $W$ of workload $E = 10$ and $\mu = 4$ on core $c_4$ and $P = 10$ under $\overrightarrow{Q_1}, \overrightarrow{Q_2}$ and $\overrightarrow{Q_3}$ under $\overrightarrow{Q_1}$, $\overrightarrow{Q_2}$ and $\overrightarrow{Q_3}$.
Ques.: Calculate Span $W$ of workload $E = 10$ and $\mu = 4$ on core $c_4$ and $P = 10$ under $\vec{Q}_1, \vec{Q}_2$ and $\vec{Q}_3$.

$\vec{Q}_1 = \{1,2,3,4\}$

$\vec{Q}_2 = \{1,4,3,2\}$

$\vec{Q}_3 = \{1,2,3,4\}$
Ques.: Calculate Span W of workload $E = 10$ and $\mu = 4$ on core $c_4$ and $P = 10$ under $\vec{Q}_1, \vec{Q}_2$ and $\vec{Q}_3$ under $\vec{Q}_1, \vec{Q}_2$ and $\vec{Q}_3$.
Ques.: Calculate Span $W$ of workload $\mathbf{E} = 10$ and $\mu = 4$ on core $c_4$ and $P = 10$ under $\mathbf{Q}_1, \mathbf{Q}_2$ and $\mathbf{Q}_3$ under $\mathbf{Q}_1, \mathbf{Q}_2$ and $\mathbf{Q}_3$.

$\overline{\mathbf{Q}}_1 = \{1,2,3,4\}$
$\overline{\mathbf{Q}}_2 = \{1,4,3,2\}$
$\overline{\mathbf{Q}}_3 = \{1,2,3,4\}$

Span $W = 28$
Ques.: Span $W$ of workload $E = 10$ and $\mu = 4$ on core $c_4$ and $P = Q_{tot} = 10$ under static and dynamic budgets?
Ques.: Span $W$ of workload $E = 10$ and $\mu = 4$ on core $c_4$ and $P = Q_{tot} = 10$ under static and dynamic budgets?
Ques.: Span $W$ of workload $E = 10$ and $\mu = 4$ on core $c_4$ and $P = Q_{tot} = 10$ under static and dynamic budgets?
Evaluation: Avionics Scenario

- $= 41666$

- $\mu / (E + \mu) \in [0.5, 0.99]$
Evaluation: Avionics Scenario

Goal

- Compare schedulability ratios under three budget assignment policies: Dynamic (DY) vs. Static Even (SE), Static Uneven (SU)

- $\frac{\mu}{(E + \mu)} \in [0.5, 0.99]$
Evaluation: Avionics Scenario

41666

Goal

- Compare schedulability ratios under three budget assignment policies: Dynamic (DY) vs. Static Even (SE), Static Uneven (SU)

Realistic multi-core platform values

- $L_{max} = 2.4 \times 10^{-8}$ s
- $P = 1ms; \quad Q_{tot} = tot t tot = 41666$

- $\mu / (E + \mu) \in [0.5,0.99]$
Evaluation: Avionics Scenario

41666

Goal

- Compare **schedulability ratios** under three budget assignment policies: Dynamic (DY) vs. Static Even (SE), Static Uneven (SU)

Realistic multi-core platform values

- \( L_{\text{max}} = 2.4 \times 10^{-8} \text{ s} \)

Workload Set Generation

- Total workloads: \( m \times 4 \)

\[ \mu / (E + \mu) \in [0.5, 0.99] \]
Goal

- Compare **schedulability ratios** under three budget assignment policies: Dynamic (DY) vs. Static Even (SE), Static Uneven (SU)

Realistic multi-core platform values

- $L_{\text{max}} = 2.4 \times 10^{-8} \text{ s}$

Workload Set Generation

- Total workloads: $m \times 4$
- Allocation to cores: Random 4 workloads per core

\[ \mu / (E + \mu) \in [0.5, 0.99] \]
Goal

- Compare *schedulability ratios* under three budget assignment policies: Dynamic (DY) vs. Static Even (SE), Static Uneven (SU)

**Realistic multi-core platform values**

- \( L_{\text{max}} = 2.4 \times 10^{-8} \) s

**Workload Set Generation**

- Total workloads: \( m \times 4 \)
- Allocation to cores: Random 4 workloads per core
- \( rl = 0; H=D = 128ms \)

\[ \frac{\mu}{(E+\mu)} \in [0.5,0.99] \]
Evaluation: Avionics Scenario

$\mu / (E + \mu) \in [0.5, 0.99]$  

41666

Goal

- Compare schedulability ratios under three budget assignment policies: Dynamic (DY) vs. Static Even (SE), Static Uneven (SU)

Realistic multi-core platform values

- $L_{max} = 2.4 * 10^{-8}$ s

Workload Set Generation

- Total workloads: $m * 4$
- Allocation to cores: Random 4 workloads per core
- $rl = 0; H = D = 128ms$
- Memory-intensive workload = $\left( \mu / (E + \mu) \right) \in [0.5, 0.99]$
Evaluation: Avionics Scenario

\[ \frac{\mu}{(E_E + \mu)} \in [0.5, 0.99] \]

41666

Goal

- Compare **schedulability ratios** under three budget assignment policies: Dynamic (**DY**) vs. Static Even (**SE**), Static Uneven (**SU**)

Realistic multi-core platform values

- \( L_{max} = 2.4 \times 10^{-8} \) s

Workload Set Generation

- Total workloads: \( m \times 4 \)
- Allocation to cores: Random 4 workloads per core
- \( r_l = 0; H = D = 128ms \)

Core Scheduling: lower index first

\[ \frac{\mu}{(E_E + \mu)} \in [0.5, 0.99] \]
Evaluation: Effect of Varying no. of Cores

\[ m = \{4, 8, 12\} \]
Evaluation: Effect of Varying no. of Cores

$m = \{4, 8, 12\}$

x-axis: $U$: [0.1, 0.9], step size 0.01

DY policy significantly outperforms static policies: SU and SE
Evaluation: Effect of Varying no. of Cores

\[ m = \{4, 8, 12\} \]

x-axis: \( U: [0.1, 0.9] \), step size 0.01

**DY policy significantly outperforms** static policies: SU and SE
Evaluation: Effect of Varying no. of Cores

$m = \{4, 8, 12\}$

x-axis: $U$: [0.1, 0.9],
step size 0.01

DY policy significantly outperforms static policies: SU and SE
Evaluation: Effect of Varying no. of Cores

$m = \{4, 8, 12\}$

x-axis: $U$: [0.1, 0.9], step size 0.01

DY policy significantly outperforms static policies: SU and SE
Evaluation: Sensitivity to Memory Intensity ratio
Evaluation: Sensitivity to Memory Intensity ratio

Memory Intensity ratio ($Mlr$) varies the no. of memory-intensive workloads
Memory Intensity ratio ($M_{Ir}$) varies the no. of memory-intensive workloads

$M_{Ir}$: {0.15, 0.25, 0.50}
Evaluation: Sensitivity to Memory Intensity ratio

Memory Intensity ratio ($Mlr$) varies the no. of memory-intensive workloads

$Mlr$: \{0.15, 0.25, 0.50\}

x-axis: $U=[0.1, 0.9]$, step size 0.01

DY policy significantly outperforms static policies: SU and SE
Memory Intensity ratio ($MIr$) varies the no. of memory-intensive workloads

$MIr$: \{0.15, 0.25, 0.50\}

x-axis: $U$=[0.1, 0.9], step size 0.01

DY policy significantly outperforms static policies: SU and SE
Evaluation: Sensitivity to Memory Intensity ratio

Memory Intensity ratio ($Mlr$) varies the no. of memory-intensive workloads

$Mlr$: $\{0.15, 0.25, 0.50\}$

x-axis: $U=[0.1, 0.9]$, step size 0.01

DY policy significantly outperforms static policies: SU and SE
Evaluation: Sensitivity to Memory Intensity ratio

Memory Intensity ratio ($Mlr$) varies the no. of memory-intensive workloads

$Mlr$: \{0.15, 0.25, 0.50\}

x-axis: $U=[0.1, 0.9]$, step size 0.01

DY policy significantly outperforms static policies: SU and SE
Stall Curves:

- **Regulation case**: (Convex)

Span Analysis

- for Static Budget
  - Fixed-point iterative algorithm
- for Dynamic Budget
  - Stall Maximization algorithm

Heuristic for dynamic budget assignment
Although in multi-cores, performance bottleneck is memory BW, state-of-the-art mainly limited to CPU-level scheduling
Conclusion

Although in multi-cores, performance bottleneck is memory BW, state-of-the-art mainly limited to CPU-level scheduling.

We proposed to schedule memory BW over time to increase schedulability.
Conclusion

Although in multi-cores, performance bottleneck is memory BW, state-of-the-art mainly limited to CPU-level scheduling.

We proposed to schedule memory BW over time to increase schedulability.

We proposed a method to analyse WCET / Response Time.
Although in multi-cores, performance bottleneck is memory BW, state-of-the-art mainly limited to CPU-level scheduling

We proposed to schedule memory BW over time to increase schedulability

We proposed a method to analyse WCET / Response Time

- when memory BW over time and workload priority order are known
Conclusion

Although in multi-cores, performance bottleneck is memory BW, state-of-the-art mainly limited to CPU-level scheduling

We proposed to schedule memory BW over time to increase schedulability

We proposed a method to analyse WCET / Response Time
- when memory BW over time and workload priority order are known
- covers both: static case and dynamic
Although in multi-cores, performance bottleneck is memory BW, state-of-the-art mainly limited to CPU-level scheduling.

We proposed to schedule memory BW over time to increase schedulability.

We proposed a **method to analyse WCET / Response Time**
- when memory BW over time and workload priority order are known
- covers both: static case and dynamic
- works with event-driven (in paper) and time-driven systems
Although in multi-cores, performance bottleneck is memory BW, state-of-the-art mainly limited to CPU-level scheduling

We proposed to schedule memory BW over time to increase schedulability.

We proposed a method to analyse WCET / Response Time

- when memory BW over time and workload priority order are known
- covers both: static case and dynamic
- works with event-driven (in paper) and time-driven systems

Evaluation: Dynamic policy dominates static even and static uneven policies
Although in multi-cores, **performance bottleneck is memory BW, state-of-the-art mainly limited to CPU-level scheduling**

We proposed to **schedule memory BW over time to increase schedulability**

We proposed a **method to analyse WCET / Response Time**
- when memory BW over time and workload priority order are known
- covers both: static case and dynamic
- works with event-driven (in paper) and time-driven systems

**Evaluation**: Dynamic policy dominates static even and static uneven policies
- when cores are varied from 4 to 12
Although in multi-cores, performance bottleneck is memory BW, state-of-the-art mainly limited to CPU-level scheduling

We proposed to schedule memory BW over time to increase schedulability

We proposed a method to analyse WCET / Response Time

- when memory BW over time and workload priority order are known
- covers both: static case and dynamic
- works with event-driven (in paper) and time-driven systems

Evaluation: Dynamic policy dominates static even and static uneven policies

- when cores are varied from 4 to 12
- when the ratio of memory-intensive workloads is varied from 15% to 50%
Thank You