DASH: Deadline-Aware High-Performance Memory Scheduler for Heterogeneous Systems with Hardware Accelerators

Hiroyuki Usui, Lavanya Subramanian
Kevin Chang, Onur Mutlu

DASH source code is available at GitHub
https://github.com/CMU-SAFARI/HWASim
Current SoC Architectures

- Heterogeneous agents: CPUs and HWAs
  - HWA: Hardware Accelerator
- Main memory is shared by CPUs and HWAs → Interference

How to schedule memory requests from CPUs and HWAs to mitigate interference?
DASH Scheduler: Executive Summary

- **Problem**: Hardware accelerators (HWAs) and CPUs share the same memory subsystem and interfere with each other in main memory.
- **Goal**: Design a memory scheduler that improves CPU performance while meeting HWAs’ deadlines.
- **Challenge**: Different HWAs have different memory access characteristics and different deadlines, which current schedulers do not smoothly handle.
  - Memory-intensive and long-deadline HWAs significantly degrade CPU performance *when they become high priority* (due to slow progress).
  - Short-deadline HWAs sometimes miss their deadlines *despite high priority*.
- **Solution**: DASH Memory Scheduler
  - Prioritize HWAs over CPU anytime when the HWA is not making good progress.
  - Application-aware scheduling for CPUs and HWAs.
- **Key Results**:
  1) Improves CPU performance for a wide variety of workloads by 9.5%.
  2) Meets 100% deadline met ratio for HWAs.
- DASH source code freely available on the GitHub.
Outline

- Introduction
- Problem with Existing Memory Schedulers for Heterogeneous Systems
- DASH: Key Ideas
- DASH: Scheduling Policy
- Evaluation and Results
- Conclusion
Outline

- Introduction
- Problem with Existing Memory Schedulers for Heterogeneous Systems
- DASH: Key Ideas
- DASH: Scheduling Policy
- Evaluation and Results
- Conclusion
Existing QoS-Aware Scheduling Scheme

- **Dynamic Prioritization for a CPU-GPU System** [Jeong et al., DAC 2012]
  - Dynamically adjust GPU priority based on its progress
  - Lower GPU priority if GPU is making a good progress to achieve its target frame rate

- We apply this scheme for a wide variety of HWAs
  - Compare HWA’s current progress against expected progress
    - **Current Progress**: \( \frac{\text{The number of finished memory requests for a period}}{\text{The number of total memory requests for a period}} \)
    - **Expected Progress**: \( \frac{\text{Elapsed cycles in a period}}{\text{Total cycles in a period}} \)
  - Every scheduling unit, dynamically adjust HWA priority
    - If **Expected Progress** > EmergentThreshold (=0.9): HWA > CPU
    - If \((\text{Current Progress}) > (\text{Expected Progress})\): HWA < CPU
    - If \((\text{Current Progress}) \leq (\text{Expected Progress})\): HWA = CPU
Problems in Dynamic Prioritization

- **Dynamic Prioritization for a CPU-HWA system**
  - Compares HWA’s *current progress* against *expected progress*
    - **Current Progress**: (The number of finished memory requests for a period) / (The number of total memory requests for a period)
    - **Expected Progress**: (Elapsed cycles in a period) / (Total cycles in a period)
  - Every scheduling unit, dynamically adjust HWA priority
    - If *Expected Progress* > EmergentThreshold (=0.9): HWA > CPU
    - If (Current Progress) > (Expected Progress): HWA < CPU
    - If (Current Progress) <= (Expected Progress): HWA = CPU

1. An HWA is prioritized over CPU cores *only when* it is closed to HWA’s deadline
   - The HWA often misses deadlines
2. This scheme does not consider the diverse memory access characteristics of CPUs and HWAs
   - It treats each CPU and each HWA equally
   - Missing opportunities to improve system performance
Outline

- Introduction
- Problem with Existing Memory Schedulers for Heterogeneous Systems
- DASH: Key Ideas
- DASH: Scheduling Policy
- Evaluation and Results
- Conclusion
Key Idea 1: Distributed Priority

- **Problem 1:** An HWA is prioritized over CPU cores only when it is close to HWA’s deadline

- **Key Idea 1:** Distributed Prioritization for a CPU-HWA system
  - Compares HWA’s current progress against expected progress
    - **Current Progress:** (The number of finished memory requests for a period) / (The number of total memory requests for a period)
    - **Expected Progress:** (Elapsed cycles in a period) / (Total cycles in a period)
  - Dynamically adjust HWA priority based on its progress every scheduling unit
    - If **Expected Progress** > EmergentThreshold (=0.9) : **HWA > CPU**
    - If (Current Progress) > (Expected Progress) : **HWA < CPU**
    - If (Current Progress) <= (Expected Progress) : **HWA > CPU**

Prioritize HWAs over CPU anytime when the HWA is not making good progress
Example: Scheduling HWA and CPU Requests

- Scheduling requests from 2 CPU applications and a HWA
  - CPU-A: memory non-intensive application
  - CPU-B: memory intensive application

**Alone Execution Timeline**

- Period = 20T

**Deadline for 10 Requests**
DASH: Distributed Priority

- **Distributed Priority (Scheduling unit = 4T)**

CPU-A

- COMP.
- STALL
- Req x1

CPU-B

- COMP.
- STALL
- Req x7

HWA

- COMP.
- COMP.
- Req x10

DRAM

- H H H H

HWA > CPU
Current: 0 / 10
Expected: 0 / 20
DASH: Distributed Priority

- Distributed Priority (Scheduling unit = 4T)

CPU-A

- COMP.
- STALL
- COMP.

CPU-B

- COMP.
- STALL

HWA

- COMP.
- COMPUTATION

DRAM

- H H H H A B B B

HWA < CPU
Current: 4 / 10
Expected: 5 / 20
**DASH: Distributed Priority**

- **Distributed Priority (Scheduling unit = 4T)**

<table>
<thead>
<tr>
<th>CPU-A</th>
<th>COMP.</th>
<th>STALL</th>
<th>COMP.</th>
<th>STALL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Req x1</td>
<td></td>
<td>Req x1</td>
<td></td>
</tr>
<tr>
<td>CPU-B</td>
<td>COMP.</td>
<td>STALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Req x7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HWA</td>
<td>COMP.</td>
<td>COMPUTATION</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Req x10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRAM</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>A</td>
</tr>
</tbody>
</table>

HWA > CPU
Current: 4 / 10
Expected: 8 / 20
DASH: Distributed Priority

- **Distributed Priority (Scheduling unit = 4T)**

**Diagram Description:**
- **CPU-A:**
  - **COMP.**
  - **STALL**
  - **COMP.**
  - **STALL**
  - **COMP.**

- **CPU-B:**
  - **COMP.**
  - **STALL**

- **HWA:**
  - **COMP.**
  - **COMPUTATION**

- **DRAM:**
  - **H**
  - **H**
  - **H**
  - **H**
  - **A**
  - **B**
  - **B**
  - **B**
  - **H**
  - **H**
  - **H**
  - **H**
  - **A**
  - **B**
  - **B**
  - **B**

**Legend:**
- **Req x1:** Request 1
- **Req x7:** Request 7
- **Req x10:** Request 10

**Status:**
- **HWA < CPU**
- Current: 8 / 10
- Expected: 12 / 20
DASH: Distributed Priority

- **Distributed Priority (Scheduling unit = 4T)**

![Diagram of CPU-A, CPU-B, HWA, and DRAM with requests, completion, and stalls]

- **CPU-A:**
  - Req x1
  - Req x1
  - Req x1

- **CPU-B:**
  - Req x7

- **HWA:**
  - Req x10

- **DRAM:**
  - H H H H H A B B B B H H H H A B B B B H H A B

**Notes:**
- HWA > CPU
- Current: 8 / 10
- Expected: 16 / 20
Problem 2: Application-unawareness

- **Problem 2 (Application-unawareness):** Existing memory schedulers for heterogeneous systems do not consider the diverse memory access characteristics of CPUs and HWAs.

- Application-unawareness causes two problems:
  - **Problem 2.1:** When a HWA has high priority (i.e., not measuring up to its expected progress), it interferes with all CPU cores for a long time.
  - **Problem 2.2:** A HWA with a short period misses its deadlines due to fluctuations in available memory bandwidth (due to priority changes of other HWAs).
Problem 2.1 and Its Solution

- **Problem 2.1 Restated:** When HWA is low priority, it is deprioritized too much → It becomes high priority as a result and destroys CPU progress.

- **Goal:** Avoid making the HWA high priority as much as possible.

![Diagram showing CPU and HWA interactions]

HWA delays both A and B.
Key Idea 2.1: Application-aware Scheduling for CPUs

Key Idea 2.1: HWA priority over CPUs should depend on CPU memory intensity

- **Not all CPUs are equal**
  - Memory-intensive cores are much less vulnerable to memory access latency
  - Memory-non-intensive cores are much more vulnerable to latency

- **While HWA has low priority, HWA is prioritized over memory-intensive cores**

  A: Memory-non-intensive, B: Memory-intensive
DASH: Application-aware Scheduling

- **Distributed Priority (Scheduling unit = 4T)**

  - CPU-A
    - COMP.
    - STALL
    - COMP.
    - STALL
    - COMP.
    - COMP.
    - Req x1
    - Req x1
    - Req x1

  - CPU-B
    - COMP.
    - STALL
    - COMP.
    - Req x7

  - HWA
    - COMP.
    - High Priority
    - Low Priority
    - High Priority
    - Low Priority
    - High Priority
    - Req x10

  - DRAM
DASH: Application-aware Scheduling

- **Distributed Priority (Scheduling unit = 4T)**

- **Application-aware Scheduling (Scheduling unit = 4T)**

HWA > CPU-A&B

Current : 0 / 10  Expected : 0 / 20
DASH: Application-aware Scheduling

- **Distributed Priority (Scheduling unit = 4T)**

<table>
<thead>
<tr>
<th>CPU A</th>
<th>COMP.</th>
<th>STALL</th>
<th>COMP.</th>
<th>STALL</th>
<th>COMP.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req x1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CPU B</th>
<th>COMP.</th>
<th>STALL</th>
<th>COMP.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req x7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HWA</th>
<th>COMP.</th>
<th>High Priority</th>
<th>Low Priority</th>
<th>High Priority</th>
<th>Low Priority</th>
<th>High Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req x10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DRAM</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>A</th>
<th>B</th>
<th>B</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req x10</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
</tbody>
</table>

- **Application-aware Scheduling (Scheduling unit = 4T)**

<table>
<thead>
<tr>
<th>CPU A</th>
<th>COMP.</th>
<th>STALL</th>
<th>COMP.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req x1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CPU B</th>
<th>COMP.</th>
<th>STALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req x7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HWA</th>
<th>COMP.</th>
<th>High Priority</th>
<th>Low Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req x10</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DRAM</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>A</th>
<th>H</th>
<th>H</th>
<th>H</th>
</tr>
</thead>
</table>

CPU-A > HWA > CPU-B
Current: 4 / 10  Expected: 4 / 20
DASH: Application-aware Scheduling

- **Distributed Priority (Scheduling unit = 4T)**

<table>
<thead>
<tr>
<th>CPU-A</th>
<th>COMP.</th>
<th>STALL</th>
<th>COMP.</th>
<th>STALL</th>
<th>COMP.</th>
<th>COMP.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Req x1</td>
<td></td>
<td>Req x1</td>
<td></td>
<td>Req x1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CPU-B</th>
<th>COMP.</th>
<th>STALL</th>
<th>COMP.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Req x7</td>
<td></td>
<td>COMP.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HWA</th>
<th>COMP.</th>
<th>High Priority</th>
<th>Low Priority</th>
<th>High Priority</th>
<th>Low Priority</th>
<th>High Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Req x10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DRAM</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>A</th>
<th>B</th>
<th>B</th>
<th>B</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>A</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>A</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td></td>
</tr>
</tbody>
</table>

- **Application-aware Scheduling (Scheduling unit = 4T)**

<table>
<thead>
<tr>
<th>CPU-A</th>
<th>COMP.</th>
<th>STALL</th>
<th>COMP.</th>
<th>COMP.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Req x1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CPU-B</th>
<th>COMP.</th>
<th>STALL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Req x7</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HWA</th>
<th>COMP.</th>
<th>High Priority</th>
<th>Low Priority</th>
<th>Low Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Req x10</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DRAM</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>A</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>A</th>
<th>H</th>
<th>H</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>A</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>A</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
</tbody>
</table>

**CPU-A > HWA > CPU-B**
Current: 7/10  Expected: 8/20
DASH: Application-aware Scheduling

- **Distributed Priority (Scheduling unit = 4T)**

<table>
<thead>
<tr>
<th>CPU-A</th>
<th>COMP.</th>
<th>STALL</th>
<th>COMP.</th>
<th>STALL</th>
<th>COMP.</th>
<th>COMP.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Req x1</td>
<td></td>
<td></td>
<td>Req x1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CPU-B</th>
<th>COMP.</th>
<th></th>
<th></th>
<th>COMP.</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Req x7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HWA</th>
<th>COMP.</th>
<th>High Priority</th>
<th>Low Priority</th>
<th>High Priority</th>
<th>Low Priority</th>
<th>High Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Req x10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

  | DRAM  | H H H H A B B B H H H H A B B B H H A B B |
  |-------|-------|-------|-------|-------|-------|
  |       |       |       |       |       |       |       |

- **Application-aware Scheduling (Scheduling unit = 4T)**

<table>
<thead>
<tr>
<th>CPU-A</th>
<th>COMP.</th>
<th>STALL</th>
<th>COMP.</th>
<th>COMP.</th>
<th>COMP.</th>
<th>COMP.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Req x1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CPU-B</th>
<th>COMP.</th>
<th></th>
<th></th>
<th>COMP.</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Req x7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HWA</th>
<th>COMP.</th>
<th>High Priority</th>
<th>Low Priority</th>
<th>Low Priority</th>
<th>Low Priority</th>
<th>Low Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Req x10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

  | DRAM  | H H H H A H H H A H H H A B B B B |
  |-------|-------|-------|-------|-------|
  |       |       |       |       |       |       |       |

CPU-A > HWA > CPU-B
Current: 10 / 10  Expected: 12 / 20
DASH: Application-aware Scheduling

- **Distributed Priority (Scheduling unit = 4T)**
  - CPU-A
    - Req x1
    - Comp.
    - Stall
    - Comp.
    - Stall
    - Comp.
    - Comp.
  - CPU-B
    - Req x7
    - Comp.
    - Stall
    - Comp.
  - HWA
    - Req x10
    - High Priority
    - Low Priority
  - DRAM
    - Req x10
    - H
    - H
    - H
    - A
    - B
    - B
    - B
    - H
    - H
    - A
    - B

- **Application-aware Scheduling (Scheduling unit = 4T)**
  - CPU-A
    - Req x1
    - Comp.
    - Stall
    - Comp.
    - Comp.
    - Comp.
    - Comp.
  - CPU-B
    - Req x7
    - Comp.
    - Stall
    - Comp.
  - HWA
    - Req x10
    - High Priority
    - Low Priority
    - Low Priority
    - Low Priority
  - DRAM
    - Req x10
    - H
    - H
    - H
    - A
    - H
    - H
    - H
    - A
    - H
    - H
    - A
    - B
    - B
    - B
    - B

**Saved Cycles**
Problem 2.2 and Its Solution

- **Problem 2.2:** A HWA with a short-deadline-period misses its deadlines due to fluctuations in available memory bandwidth (due to priority changes of other HWAs)

- **Key Idea 2.2:** Estimate the worst-case memory access latency and give a short-deadline-period HWA the highest priority for \((WorstCaseLatency) \times (NumberOfRequests)\) cycles close to its deadline
  - \(WorstCaseLatency = tRC\) : the minimum time between two DRAM row ACTIVATE commands

<table>
<thead>
<tr>
<th></th>
<th>HWA-A</th>
<th>HWA-B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period</td>
<td>63,041 Cycles</td>
<td>5,447 Cycles</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>8.32 GB/s</td>
<td>475 MB/s</td>
</tr>
</tbody>
</table>

- **HWA-A:** meets all its deadlines
- **HWA-B:** misses a deadline every 2000 periods
DASH: Summary of Key Ideas

1. Distributed priority

2. Application-aware scheduling

3. Worst-case memory access latency based prioritization
Outline

- Introduction
- Problem with Existing Memory Schedulers for Heterogeneous Systems
- DASH: Key Ideas
- DASH: Scheduling Policy
- Evaluation and Results
- Conclusion
DASH: Scheduling Policy

- DASH scheduling policy
  1. Short-deadline-period HWAs with high priority
  2. Long-deadline-period HWAs with high priority
  3. Memory non-intensive CPU applications
  4. Long-deadline-period HWAs with low priority
  5. Memory-intensive CPU applications
  6. Short-deadline-period HWAs with low priority
DASH: Scheduling Policy

- DASH scheduling policy
  1. Short-deadline-period HWAs with high priority
  2. Long-deadline-period HWAs with high priority
  3. Memory non-intensive CPU applications
  4. Long-deadline-period HWAs with low priority
  5. Memory-intensive CPU applications
  6. Short-deadline-period HWAs with low priority

Switch probabilistically
Outline

- Introduction
- Problem with Existing Memory Schedulers for Heterogeneous Systems
- DASH: Key Ideas
- DASH: Scheduling Policy
- Evaluation and Results
- Conclusion
Experimental Methodology (1/2)

- **New Heterogeneous System Simulator**
  - We have released this at GitHub (https://github.com/CMU-SAFARI/HWASim)

- **Configurations**
  - 8 CPUs (2.66GHz), 32KB/L1, 4MB Shared/L2
  - 4 HWAs
  - DDR3 1333 DRAM x 2 channels

- **Workloads**
  - CPUs: 80 multi-programmed workloads
    - SPEC CPU2006, TPC, NAS parallel benchmark
  - HWAs:
    - Image processing
    - Image recognition [Lee+ ICCD 2009] [Viola and Jones CVPR 2001]

- **Metrics**
  - CPUs: Weighted Speedup
  - HWAs: Deadline met ratio (%)
Experimental Methodology (2/2)

- Parameters of the HWAs

<table>
<thead>
<tr>
<th></th>
<th>Period</th>
<th>Bandwidth</th>
<th>Deadline Group</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IMG</strong>: Image Processing</td>
<td>33 ms</td>
<td>360MB/s</td>
<td>Long</td>
</tr>
<tr>
<td><strong>HES</strong>: Hessian</td>
<td>2 us</td>
<td>478MB/s</td>
<td>Short</td>
</tr>
<tr>
<td><strong>MAT</strong>: Matching (1) 20fps</td>
<td>35.4 us</td>
<td>8.32 GB/s</td>
<td>Long</td>
</tr>
<tr>
<td><strong>MAT</strong>: Matching (2) 30fps</td>
<td>23.6 us</td>
<td>5.55 GB/s</td>
<td>Long</td>
</tr>
<tr>
<td><strong>RSZ</strong>: Resize</td>
<td>46.5 – 5183 us</td>
<td>2.07 – 3.33 GB/s</td>
<td>Long</td>
</tr>
<tr>
<td><strong>DET</strong>: Detect</td>
<td>0.8 – 9.6 us</td>
<td>1.60 – 1.86 GB/s</td>
<td>Short</td>
</tr>
</tbody>
</table>

- Configurations of 4 HWAs

<table>
<thead>
<tr>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config-A</strong></td>
</tr>
<tr>
<td>IMG x 2, HES, MAT(2)</td>
</tr>
<tr>
<td><strong>Config-B</strong></td>
</tr>
<tr>
<td>HES, MAT(1), RSZ, DET</td>
</tr>
</tbody>
</table>
Evaluated Memory Schedulers

- **FRFCFS-St, TCM-St:** FRFCFS or TCM with *static priority* for HWAs
  - HWAs *always* have higher priority than CPUs
  - **FRFCFS-St:** FRFCFS [Zuravleff and Robinson US Patent 1997, Rixner et al. ISCA 2000] for CPUs
    - Prioritizes row-buffer hits and older requests
  - **TCM-St:** TCM [Kim+ MICRO 2010] for CPUs
    - Always prioritizes memory-non-intensive applications
    - Shuffles thread ranks of memory-intensive applications

- **FRFCFS-Dyn:** FRFCFS with *dynamic priority* for HWAs [Jeong et al., DAC 2012]
  - HWA’s priority is dynamically adjusted based on its progress
    - **FRFCFS-Dyn0.9:** EmergentThreshold = 0.9 for all HWAs (Only after 90% of the HWA’s period elapsed, the HWA has higher priority than CPUs)
    - **FRFCFS-DynOpt:** Each HWA has different EmergentThreshold to meet its deadline

- **DASH:** *Distributed Priority* + Application-aware scheduling for CPUs + HWAs
  - TCM is used for CPUs to classify memory intensity of CPUs
  - EmergentThreshold = 0.8 for all HWAs

<table>
<thead>
<tr>
<th>Config-A</th>
<th>Config-B</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMG</td>
<td>HES</td>
</tr>
<tr>
<td>0.9</td>
<td>0.2</td>
</tr>
</tbody>
</table>
Performance and Deadline Met Ratio

- **Weighted Speedup for CPUs**

  ![](chart.png)

- **Deadline Met Ratio (%) for HWAs**

<table>
<thead>
<tr>
<th></th>
<th>IMG</th>
<th>HES</th>
<th>MAT</th>
<th>RSZ</th>
<th>DET</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRFCFS-St</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>TCM-St</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>FRFCFS-Dyn0.9</td>
<td>100</td>
<td>99.4</td>
<td>46.01</td>
<td>97.98</td>
<td>97.14</td>
</tr>
<tr>
<td>FRFCFS-DynOpt</td>
<td>100</td>
<td>100</td>
<td>99.997</td>
<td>100</td>
<td>99.99</td>
</tr>
<tr>
<td>DASH</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>
Performance and Deadline Met Ratio

- **Weighted Speedup for CPUs**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>IMG</th>
<th>HES</th>
<th>MAT</th>
<th>RSZ</th>
<th>DET</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRFCFS-St</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>TCM-St</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>FRFCFS-Dyn0.9</td>
<td>100</td>
<td>99.4</td>
<td>46.01</td>
<td>97.98</td>
<td>97.14</td>
</tr>
<tr>
<td>FRFCFS-DynOpt</td>
<td>100</td>
<td>100</td>
<td>99.997</td>
<td>100</td>
<td>99.99</td>
</tr>
<tr>
<td>DASH</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

1. DASH achieves 100% deadline met ratio

- **Deadline Met Ratio (%) for HWAs**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>IMG</th>
<th>HES</th>
<th>MAT</th>
<th>RSZ</th>
<th>DET</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRFCFS-St</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>TCM-St</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>FRFCFS-Dyn0.9</td>
<td>100</td>
<td>99.4</td>
<td>46.01</td>
<td>97.98</td>
<td>97.14</td>
</tr>
<tr>
<td>FRFCFS-DynOpt</td>
<td>100</td>
<td>100</td>
<td>99.997</td>
<td>100</td>
<td>99.99</td>
</tr>
<tr>
<td>DASH</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>
1. DASH achieves 100% deadline met ratio
2. DASH achieves better performance (+9.5%) than FRFCFS-DynOpt that meets the most of HWAs’ deadlines (Optimized for HWAs)
Performance and Deadline Met Ratio

- **Weighted Speedup for CPUs**

<table>
<thead>
<tr>
<th>Method</th>
<th>0</th>
<th>0.5</th>
<th>1</th>
<th>1.5</th>
<th>2</th>
<th>2.5</th>
<th>3</th>
<th>3.5</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRFCFS-St</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCM-St</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRFCFS-Dyn0.9</td>
<td>100</td>
<td>99.4</td>
<td>46.01</td>
<td>97.98</td>
<td>97.14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRFCFS-DynOpt</td>
<td>100</td>
<td>100</td>
<td>99.997</td>
<td>100</td>
<td>99.99</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DASH</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. DASH achieves 100% deadline met ratio
2. DASH achieves better performance (+9.5%) than FRFCFS-DynOpt that meets the most of HWAs’ deadlines (Optimized for HWAs)
3. DASH achieves comparable performance to FRFCFS-Dyn0.9 that frequently misses HWAs’ deadlines (Optimized for CPUs)
DASH Scheduler: Summary

- **Problem**: Hardware accelerators (HWAs) and CPUs share the same memory subsystem and interfere with each other in main memory.

- **Goal**: Design a memory scheduler that improves CPU performance while meeting HWAs’ deadlines.

- **Challenge**: Different HWAs have different memory access characteristics and different deadlines, which current schedulers do not smoothly handle.
  - Memory-intensive and long-deadline HWAs significantly degrade CPU performance *when they become high priority* (due to slow progress).
  - Short-deadline HWAs sometimes miss their deadlines *despite high priority*.

- **Solution**: DASH Memory Scheduler
  - Prioritize HWAs over CPU anytime when the HWA is not making good progress.
  - Application-aware scheduling for CPUs and HWAs.

- **Key Results**:
  1. Improves CPU performance for a wide variety of workloads by 9.5%.
  2. Meets 100% deadline met ratio for HWAs.

- DASH source code freely available on the GitHub.
DASH: Deadline-Aware High-Performance Memory Scheduler for Heterogeneous Systems with Hardware Accelerators

Hiroyuki Usui, Lavanya Subramanian
Kevin Chang, Onur Mutlu

DASH source code is available at GitHub
https://github.com/CMU-SAFARI/HWASim
Backup Slides
Probabilistic Switching of Priorities

- Each Long-deadline-period HWA $x$ has probability $P_b(x)$

- Scheduling using $P_b(x)$
  - With a probability $P_b(x)$
    - Memory-intensive applications > Long-deadline-period HWA $x$
  - With a probability $1 - P_b(x)$
    - Memory-intensive applications < Long-deadline-period HWA $x$

- Controlling $P_b(x)$
  - Initial: $P_b(x) = 0$
  - Every SwitchingUnit:
    - If CurrentProgress > ExpectedProgress: $P_b(x) += 1\%$
    - If CurrentProgress < ExpectedProgress: $P_b(x) -= 5\%$
Priorities for Multiple Short-deadline-period HWAs

- A HWA with shorter deadline period is given higher priority (HWA-a > HWA-b)
- UPL = Urgent Period Length: \( tRC \times \text{NumberOfRequests} + a \)
- During UPL(b), HWA-a will interfere HWA-b for \( (UPL(a) \times 2) \) cycles at maximum
  - \( [UPL(b)/\text{Period(a)}] = 2 \)
  - HWA(b) might fail the deadline due to the interference from HWA-a
Priorities for Multiple Short-deadline-period HWAs

A HWA with shorter deadline period is given higher priority (HWA-a > HWA-b)

- UPL = Urgent Period Length: \( tRC \times \text{NumberOfRequests} + \alpha \)
- During UPL(b), HWA-a will interfere HWA-b for (UPL(a) x 2) cycles at maximum
  - \( \left\lceil \frac{UPL(b)}{Period(a)} \right\rceil = 2 \)
  - HWA(b) might fail the deadline due to the interference from HWA-a
- HWA-b is prioritized when the time remaining in the period is
  \( UPL(b) + UPL(a) \times 2 \) cycles
### Storage required for DASH

- **20 bytes** for each long-deadline-period HWA
- **12 bytes** for each short-deadline-period HWA

<table>
<thead>
<tr>
<th>For long-deadline-period HWA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Curr-Req</td>
</tr>
<tr>
<td>Total-Req</td>
</tr>
<tr>
<td>Curr-Cyc</td>
</tr>
<tr>
<td>Total-Cyc</td>
</tr>
<tr>
<td>Pb</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>For a short-deadline-period HWA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Priority-Cyc</td>
</tr>
<tr>
<td>Curr-Cyc</td>
</tr>
<tr>
<td>Total-Cyc</td>
</tr>
</tbody>
</table>
Simulation Parameter Details

- SchedulingUnit : 1000 CPU cycles
- SwitchingUnit : 500 CPU cycles
- ClusterFactor : 0.15
  - Fraction of total memory bandwidth allocated to memory-non-intensive CPU applications
Performance breakdown of DASH

- DA-D : Distributed Priority
- DA-D+L : DA-D + application-aware priority for CPUs
- DA-D+L+S : DA-D+L + worst-case latency based priority for short-deadline HWAs
- DA-D+L+S+P (DASH) : DA-D+L+S + probabilistic prioritization

SAFARI
Performance breakdown of DASH

- DA-D: Distributed Priority
- DA-D+L: DA-D + application-aware priority for CPUs
- DA-D+L+S: DA-D+L + worst-case latency based priority for short-deadline HWAs
- DA-D+L+S+P (DASH): DA-D+L+S + probabilistic prioritization

Distributed priority improves performance (Max +9.5%)
Performance breakdown of DASH

- DA-D: Distributed Priority
- DA-D+L: DA-D + application-aware priority for CPUs
- DA-D+L+S: DA-D+L + worst-case latency based priority for short deadline HWAs
- DA-D+L+S+P (DASH): DA-D+L+S + probabilistic prioritization

Application-aware priority for CPUs improves performance especially as the memory intensity increases (Max +7.6%)
Performance breakdown of DASH

- DA-D: Distributed Priority
- DA-D+L: DA-D + application-aware priority for CPUs
- DA-D+L+S: DA-D+L + worst-case latency based priority for short-deadline HWAs
- DA-D+L+S+P (DASH): DA-D+L+S + probabilistic prioritization

Probabilistic prioritization achieves good balance between performance and fairness

SAFARI
## Performance breakdown of DASH

### Deadline Met Ratio

<table>
<thead>
<tr>
<th>Name</th>
<th>IMG</th>
<th>HES</th>
<th>MAT</th>
<th>RSZ</th>
<th>DET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deadline group</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Long</td>
<td>Short</td>
<td>Long</td>
<td>Long</td>
<td>Short</td>
<td></td>
</tr>
<tr>
<td>FRFCFS-DynOpt</td>
<td>100</td>
<td>100</td>
<td>99.997</td>
<td>100</td>
<td>99.99</td>
</tr>
<tr>
<td>DA-D</td>
<td>100</td>
<td><strong>99.999</strong></td>
<td>100</td>
<td>100</td>
<td><strong>99.88</strong></td>
</tr>
<tr>
<td>DA-D+L</td>
<td>100</td>
<td><strong>99.999</strong></td>
<td>100</td>
<td>100</td>
<td><strong>99.87</strong></td>
</tr>
<tr>
<td>DA-D+L+S</td>
<td>100</td>
<td><strong>100</strong></td>
<td>100</td>
<td>100</td>
<td><strong>100</strong></td>
</tr>
<tr>
<td>DA-D+L+S+P</td>
<td>100</td>
<td><strong>100</strong></td>
<td>100</td>
<td>100</td>
<td><strong>100</strong></td>
</tr>
</tbody>
</table>

1. Short-deadline HWAs (HES and DET) misses deadlines on distributed priority (DA-D) and application-aware priority for CPU (DA-D+L).
2. Worst-case latency based priority (DA-D+L+S) enables short-deadline HWAs to meet their deadline.
Impact of EmergentThreshold

**CPU performance sensitivity to EmergentThreshold**

- FRFCFS-Dyn
- DASH

FRFCFS-Dyn meets all HWA deadlines

DASH meets all HWA deadlines

DASH can meet all deadlines with a high EmergentThreshold value (=0.8)
## Impact of EmergentThreshold

### Deadline-met ratio(%) of FRFCFS-Dyn

<table>
<thead>
<tr>
<th>Emergent Threshold</th>
<th>Config-A</th>
<th></th>
<th>Config-B</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HES</td>
<td>MAT</td>
<td>HES</td>
<td>MAT</td>
<td>RSZ</td>
</tr>
<tr>
<td>0–0.1</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>0.2</td>
<td>100</td>
<td>99.987</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>0.3</td>
<td>99.992</td>
<td>93.74</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>0.4</td>
<td>99.971</td>
<td>73.179</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>0.5</td>
<td>99.945</td>
<td>55.76</td>
<td>99.9996</td>
<td>99.751</td>
<td>100</td>
</tr>
<tr>
<td>0.6</td>
<td>99.905</td>
<td>44.691</td>
<td>99.989</td>
<td>94.697</td>
<td>100</td>
</tr>
<tr>
<td>0.7</td>
<td>99.875</td>
<td>38.097</td>
<td>99.957</td>
<td>86.366</td>
<td>100</td>
</tr>
<tr>
<td>0.8</td>
<td>99.831</td>
<td>34.098</td>
<td>99.906</td>
<td>74.69</td>
<td>99.886</td>
</tr>
<tr>
<td>0.9</td>
<td>99.487</td>
<td>31.385</td>
<td>99.319</td>
<td>60.641</td>
<td>97.977</td>
</tr>
<tr>
<td>1</td>
<td>96.653</td>
<td>27.32</td>
<td>95.798</td>
<td>33.449</td>
<td>55.773</td>
</tr>
</tbody>
</table>
## Impact of Emergent Threshold

### Deadline-met ratio(%) of DASH

<table>
<thead>
<tr>
<th>Emergent Threshold</th>
<th>Config-A</th>
<th>Config-B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HES</td>
<td>MAT</td>
</tr>
<tr>
<td>0–0.8</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>0.9</td>
<td>100</td>
<td>99.997</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>68.44</td>
</tr>
</tbody>
</table>
Impact of ClusterFactor

ClusterFactor is an effective knob for trading off CPU performance and fairness.
Evaluations with GPUs

- 8 CPUs + 4 HWA (Config-A) + GPU
  - 6 GPU traces: 3D mark and game
As the number of agents increases, DASH achieves greater performance improvement.

8HWAs: IMG x 2, MAT x 2, HES x 2, RSZ x 1, DET x 1
Sensitivity to Number of Agents

**Deadline Met Ratio of HES-HWA**

- FRFCFS-DynOpt
- DASH

<table>
<thead>
<tr>
<th>Number of Agents (# of CPUs)+(# of HWAs)</th>
<th>8+4</th>
<th>16+4</th>
<th>24+4</th>
<th>8+8</th>
<th>16+8</th>
<th>24+8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deadline Met Ratio(%)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>99.5</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>97.5</td>
<td>97.5</td>
<td>97.5</td>
<td>97.5</td>
<td>97.5</td>
<td>97.5</td>
</tr>
</tbody>
</table>

**Deadline Met Ratio of MAT-HWA**

- FRFCFS-DynOpt
- DASH

<table>
<thead>
<tr>
<th>Number of Agents (# of CPUs)+(# of HWAs)</th>
<th>8+4</th>
<th>16+4</th>
<th>24+4</th>
<th>8+8</th>
<th>16+8</th>
<th>24+8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deadline Met Ratio(%)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>99</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>97</td>
<td>97</td>
<td>97</td>
<td>97</td>
<td>97</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>96</td>
<td>96</td>
<td>96</td>
<td>96</td>
<td>96</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>95</td>
<td>95</td>
<td>95</td>
<td>95</td>
<td>95</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td>94</td>
<td>94</td>
<td>94</td>
<td>94</td>
<td>94</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>93</td>
<td>93</td>
<td>93</td>
<td>93</td>
<td>93</td>
<td>93</td>
</tr>
</tbody>
</table>
Sensitivity to Number of Channels

![Bar Chart]

Weighted Speedup

Number of Memory Channels

- FRFCFS-DynOpt
- DASH

- 2 channels: DASH > FRFCFS-DynOpt
- 4 channels: DASH > FRFCFS-DynOpt
- 8 channels: DASH significantly higher than FRFCFS-DynOpt
DASH: Application-aware scheduling for HWAs

- Categorize HWAs as long-deadline-period vs. short-deadline-period statically

- Adjust the priorities of each dynamically
  - Short-deadline-period HWA: becomes high priority if \( \text{time remaining in period} = tRC \times \text{NumberOfRequests} + a \)
  - Long-deadline-period HWA: becomes high priority if \( \text{Current progress} \leq \text{Expected progress} \)