Slack-Aware Shared Bandwidth Management in GPUs

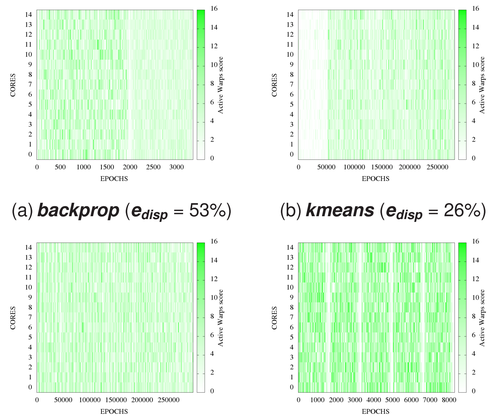
Saumay Dublish
saumay.dublish@ed.ac.uk

THE UNIVERSITY of EDINBURGH

informatics

Motivation

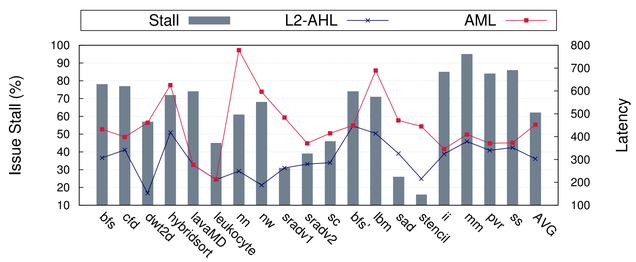
- **Warp Scheduling.** When a warp encounters a long latency memory operation, it is de-scheduled and a new warp is scheduled from a pool of active warps.
- This allows memory latency to be overlapped with execution.
- In memory-intensive applications, cores often exhaust all the active warps thereby exposing the memory latencies.
- **Disparate Bandwidth Criticality.** When cores with varying active warps are present in the same epoch, the shared bandwidth appears more performance critical to cores with few or no active warps.



(a) ***backprop*** (e_{disp} = 53%) (b) ***kmeans*** (e_{disp} = 26%)

(c) ***lud*** (e_{disp} = 49%) (d) ***pathfinder*** (e_{disp} = 72%)

- **Greedy Bandwidth Acquisition.** Each core employs a greedy policy to acquire the shared bandwidth in order to maximise its own bandwidth utilisation, thereby excessively depleting the shared bandwidth and causing congestion.
- As a result of congestion, memory latencies often exceed the normal memory latencies by a factor of 2-3×.



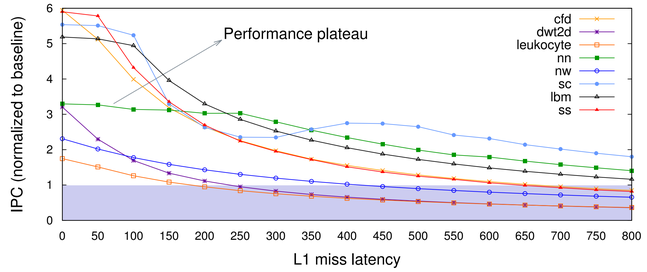
- It is analogous to the *Tragedy of the Commons*, a problem in economics where a strategy best for an individual in using an unregulated common resource may not yield the most optimal outcome for the group.

Problem Statement

The existing policy of allocating shared on-chip and off-chip memory bandwidth presents two major shortcomings–

❶ allocating bandwidth to cores without regard to their criticality,

❷ excessive depletion of shared bandwidth leading to congestion and therefore high memory latencies.



More evidence on scope for improvement. The baseline performance is far from saturation with respect to memory latencies. Therefore, there lies a significant opportunity to improve performance by regulating the bandwidth allocation and reducing the performance critical latencies.

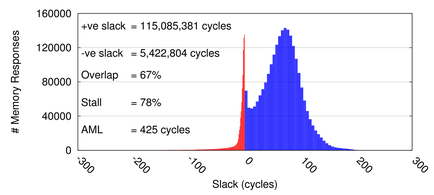
Measuring Core Criticality

- **Slack Metric.** To measure the difference in criticality of shared memory bandwidth on different cores, we record the slack in utilising a cache line that is newly fetched from the lower levels of the memory.
- We refer to the slack as positive if the memory response arrives sooner than it is required to prevent the core from stalling, and negative if the memory response arrives after the core has stalled.

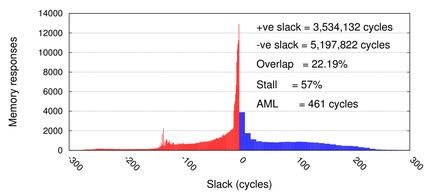
$$slack_n(t) = \begin{cases} k \times W_{n|active}(t) & \text{if } W_{n|active}(t) > 0 \\ -t_{n|stall} & \text{otherwise} \end{cases} \quad (1)$$

where:

$slack_n(t)$ = Slack of memory response received on core n at time t
$W_{n|active}(t)$ = No. of active warps on core n at time t
k = Average core busy cycles per warp between consecutive stalls
$t_{n|stall}$ = No. of cycles for which core n has been stalled since last issue



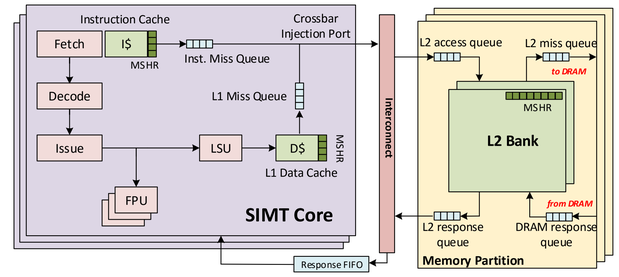
(a) bfs



(b) dwt2D

In dwt2D, considerable overlap of negative and positive slack presents an opportunity to trade-off latter with former.

Shared Bandwidth Management



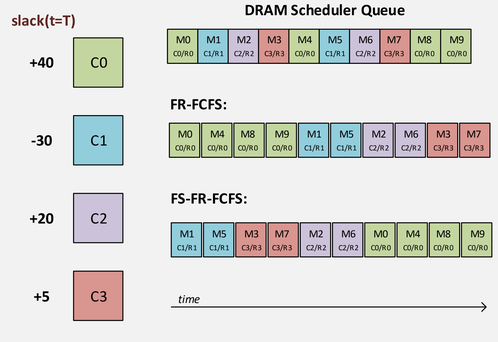
- ❶ **Slack-Aware DRAM Scheduling.** In epochs with significant disparity in slack, the proposed scheduler prioritises requests from cores with negative slack, ahead of row-buffer hits.
- It reduces the memory latencies that lie in the critical path (reducing negative slack) at the cost of increasing the latencies that can be hidden by multithreading (reducing positive slack).

Off-chip bandwidth regulation

if *disparity* && $\Delta_{slack(T)} \geq \Delta_{threshold}$:
FS-FR-FCFS
else:
FR-FCFS

FS-FR-FCFS:
(First Stalled-First Ready FCFS)
1. Minimum slack first
2. Row-hit first
3. Oldest first

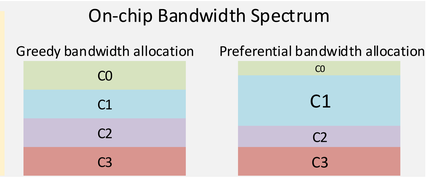
FR-FCFS:
(First Ready FCFS)
1. Row-hit first
2. Oldest first



- ❷ **Request Throttling.** To regulate on-chip bandwidth, we throttle the rate of sending requests to the shared level for cores that contain high number of active warps, resulting in reduced congestion in the memory system.

On-chip bandwidth regulation

if *disparity* && $\Delta_{slack(T)} \geq \Delta_{threshold}$:
preferential bandwidth allocation
else:
greedy bandwidth allocation



Results and Future Work. Preliminary results for dwt2D suggest a 7% reduction in negative slack and a 5% reduction in AML. The current challenge is to devise cost-efficient hardware implementation to communicate slack across the memory system without aggravating the bandwidth bottleneck.

Advisers: Prof. Nigel Topham, Dr. Vijay Nagarajan

icsa | Institute for Computing Systems Architecture