Juan FANG, Sheng LIN, Huijing YANG, Yixiang XU, Xing SU, 2023.
A perceptual and predictive batch-processing memory scheduling strategy for a CPU-GPU heterogeneous system. *Frontiers of Information Technology & Electronic Engineering*, 24(7):994-1006.
https://doi.org/10.1631/FITEE.2200449

A perceptual and predictive batch-processing memory scheduling strategy for a CPU-GPU heterogeneous system

Key words: CPU-GPU heterogeneous; Multi-core; Unified memory; Access scheduling

Corresponding author: Juan FANG
E-mail: fangjuan@bjut.edu.cn
ORCID: https://orcid.org/0000-0002-4542-8727

Motivation

1. In a CPU-GPU shared memory architecture, memory access requests from CPU and GPU applications are interleaved in the memory request buffer, impacting the principle of memory access locality and reducing the memory access efficiency.
2. Applications with a large number of memory access requests, especially GPU applications, significantly impact the execution of memory-sensitive applications.
3. As the number of workloads increases on CPU and GPU, the memory access interference between CPU and GPU cores becomes more severe, adversely affecting the performance of both CPU and GPU.

Method

Analyzing the impact of memory access interference on CPU and GPU performance in a simulation environment:

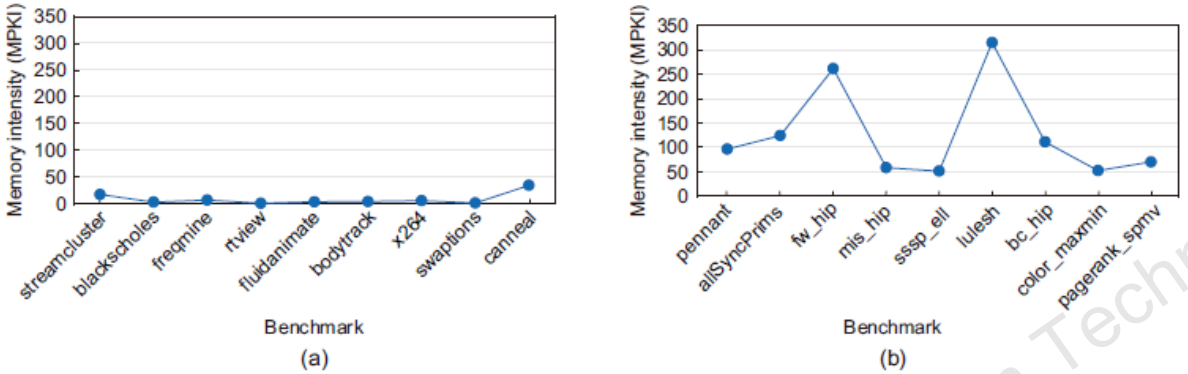


Fig. 2 Comparison of memory intensity between CPU (a) and GPU (b) applications (MPKI: misses per thousand instructions)

Table 2 Configuration of the simulation system

Component	Configuration
CPU	8 cores, 2.3 GHz, X86, out-of-order
GPU	16 computing units, 800 MHz, X86, out-of-order
LLC	512 KB shared, 128-bit line, 8-way, LRU
DRAM	8 GB; channel/rank/bank: 1/2/8; row buffer size: 2 KB

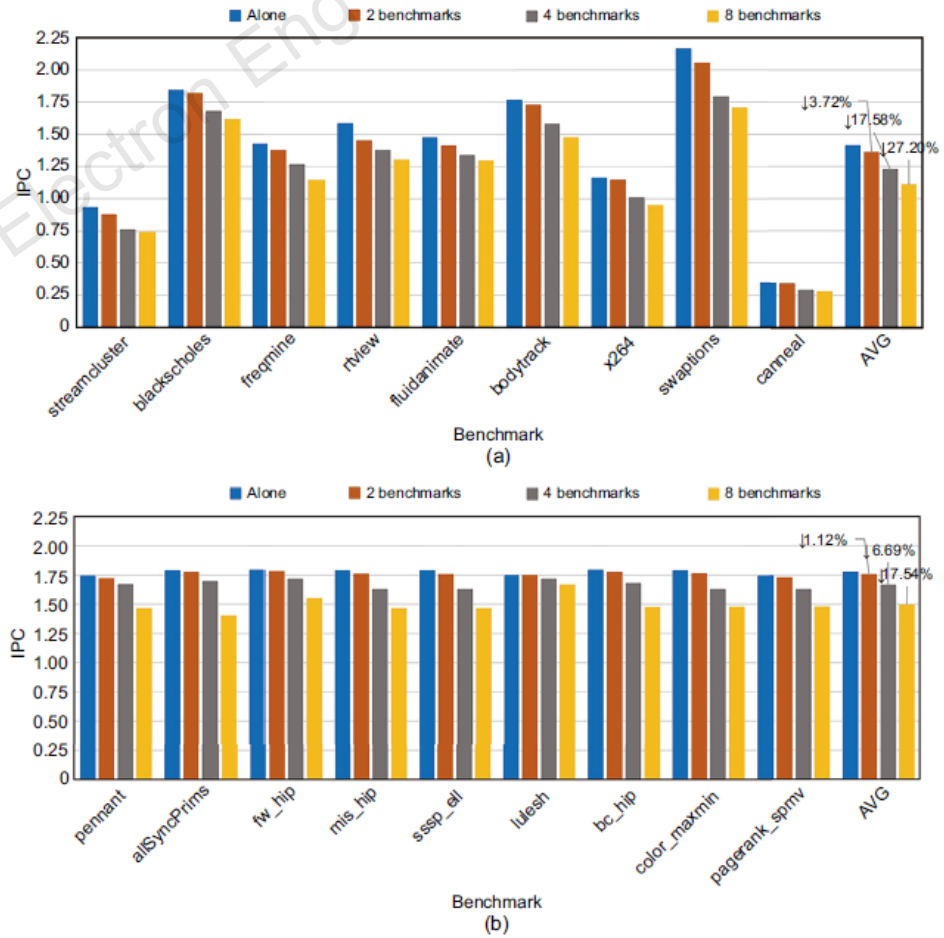


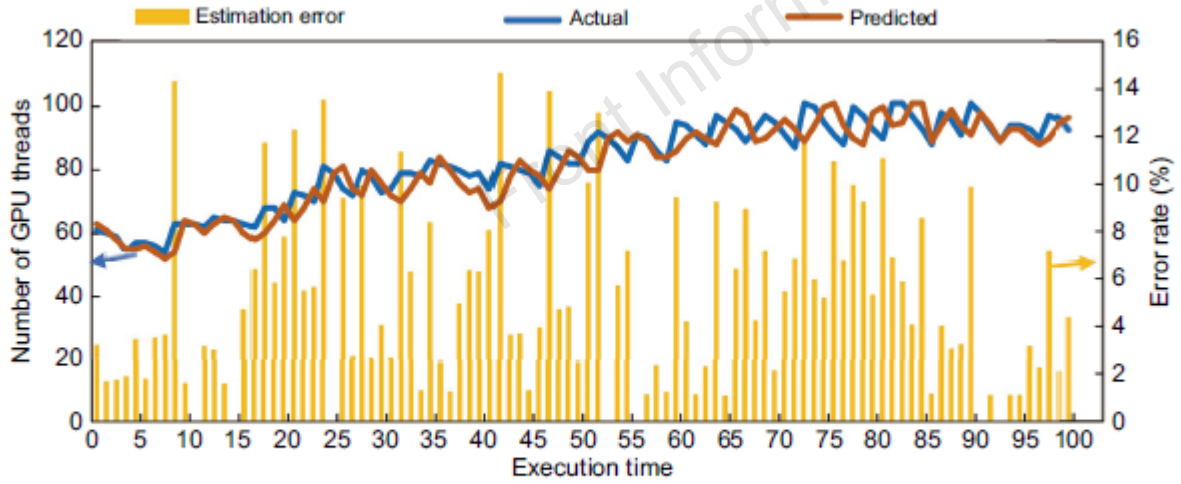
Fig. 4 CPU (a) and GPU (b) throughput when executing different numbers of benchmarks simultaneously (IPC: instructions per cycle)

Method

A simple yet effective scheme to predict GPU latency tolerance:

$$\text{LatencyTolerance} = \frac{\text{GPUThread}_{\text{cycle}}}{\text{GPUThread}_{\text{quantum}}}. \quad (1)$$

$$\text{GPU}_{\text{PRO}} = G' \frac{\text{numREQ}_{\text{GPU}}}{\text{numREQ}_{\text{CPU}} + \text{numREQ}_{\text{GPU}}} \cdot \left(1 + \frac{1}{\text{LatencyTolerance}}\right). \quad (2)$$



Algorithm 1 Perceptual and predictive batch-processing (PPBP)

Input: $\text{numREQ}_{\text{CPU}}$, $\text{numREQ}_{\text{GPU}}$, and LatencyTolerance

Output: Schedulability of the memory requests

1: **if** $\text{numREQ}_{\text{CPU}} \neq 0$ and $\text{numREQ}_{\text{GPU}} \neq 0$ **then**
2: compute the GPU proportion GPU_{PRO}
3: **else if** $\text{numREQ}_{\text{CPU}} = 0$ **then**
4: $\text{GPU}_{\text{PRO}} = 1$
5: **else**
6: $\text{GPU}_{\text{PRO}} = 0$
7: **end if**
8: nextQuantum = Quantum
9: **if** isCPU and $\text{GPU}_{\text{PRO}} \neq 0$ **then**
10: isCPU = false
11: nextQuantum* = GPU_{PRO}
12: **else if** !isCPU and $\text{GPU}_{\text{PRO}} \neq 1.0$ **then**
13: isCPU = true
14: nextQuantum* = $1 - \text{GPU}_{\text{PRO}}$
15: **end if**

Fig. 5 Estimated number of GPU threads (Data are collected at an interval of five quanta, and one scale of the execution time spans 5×10^5 ticks)

Method

A new memory access scheduling strategy, PPBP, and its performance evaluation:

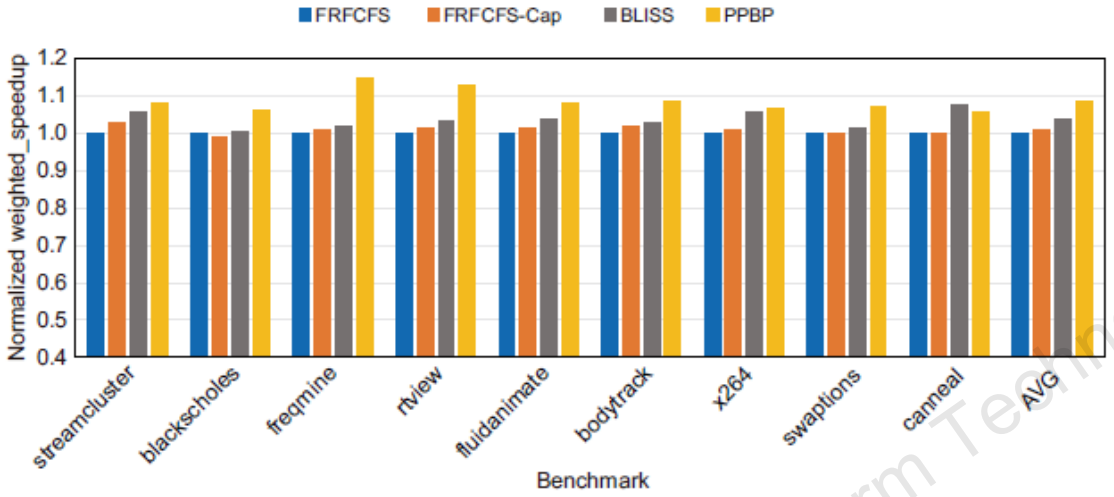


Fig. 6 CPU performance of PPBP and three other algorithms

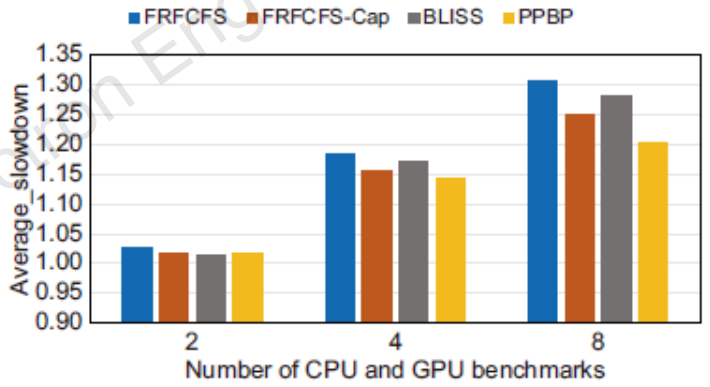


Fig. 8 Average_slowdown at different numbers of CPU and GPU applications

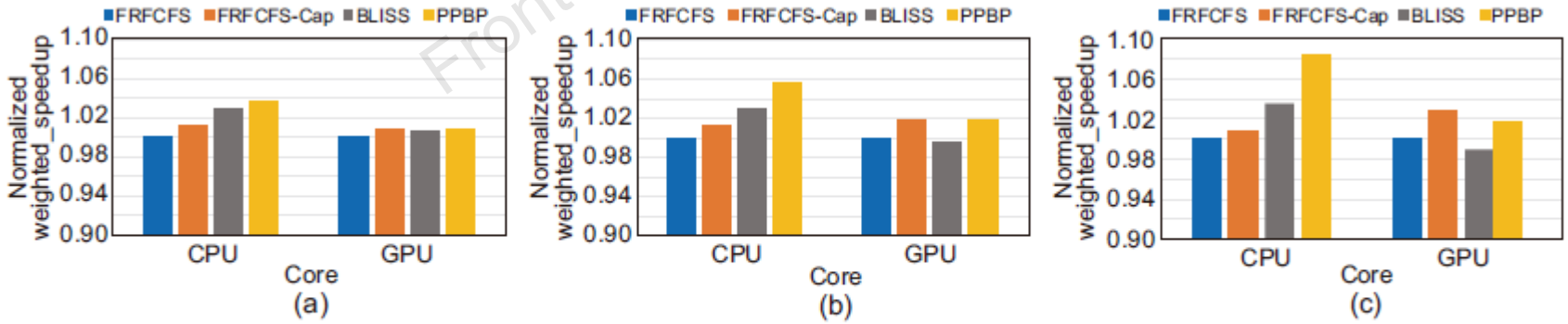


Fig. 7 Performance of PPBP and three other algorithms with one (a), two (b), and four (c) CPU and GPU benchmarks

Method

A new memory access scheduling strategy, PPBP, and its performance evaluation:

Table 1 Additional state (over FRFCFS) required for a possible PPBP implementation

Component	Description/Purpose	Size (number of additional bits)
$\text{numREQ}_{\text{cpu}}$ (register)	Number of CPU requests in the request buffer	$\log_2 \text{numREQ}_{\text{CPU}}$ (6)
$\text{numREQ}_{\text{gpu}}$ (register)	Number of GPU requests in the request buffer	$\log_2 \text{numREQ}_{\text{GPU}}$ (12)
$\text{GPUThread}_{\text{cycle}}$ (register)	Number of GPU memory access threads in a cycle	$\log_2 \text{GPUThread}_{\text{cycle}}$ (8)
$\text{GPUThread}_{\text{quantum}}$ (register)	Number of GPU memory access threads in a span	$\log_2 \text{GPUThread}_{\text{quantum}}$ (8)
Time counter of a quantum (time counter)	Count the execution time and update the range	

Table 3 Characteristics of CPU and GPU benchmarks running on the baseline

No.	Benchmark		MPKI		Row buffer hit rate (%)	
	CPU	GPU	CPU	GPU	CPU	GPU
1	streamcluster	pennant	16.83	96.35	68.00	75.94
2	blackscholes	allSyncPrims	2.68	123.98	64.97	77.82
3	freqmine	fw_hip	6.47	261.43	68.82	45.54
4	rtview	mis_hip	0.54	58.17	64.48	68.81
5	fluidanimate	sssp_ell	3.31	51.27	60.62	80.29
6	bodytrack	lulesh	3.76	314.77	73.11	84.08
7	x264	bc_hip	5.29	111.17	56.72	65.68
8	swaptions	color_maxmin	1.08	52.27	80.04	60.78
9	canneal	pagerank_spmv	34.02	70.00	38.52	71.90
10	facesim	DNNMark	13.64	143.53	70.10	75.64
11	ferret	ForceTreeTest	1.31	88.57	62.27	69.56
12	vips	square	4.28	63.45	66.52	75.59

MPKI: misses per thousand instructions

Conclusions

1. We investigated the impact of interference between CPU and GPU memory requests on performance in the CPU-GPU heterogeneous shared-memory architecture.
2. We proposed a simple yet effective scheme to predict GPU latency tolerance.
3. We introduced a new memory access scheduling strategy, PPBP, specifically for CPU-GPU heterogeneous shared-memory architecture. It has reasonable hardware complexity and improves CPU performance by 8.53% without affecting GPU performance.

Juan FANG, received her MS degree from Jilin University of Technology, Changchun, China, in 1997, and her PhD degree from the College of Computer Science, Beijing University of Technology, Beijing, China, in 2005. In 1997, she joined the College of Computer Science, Beijing University of Technology. From 2015, she is a professor of Beijing University of Technology. Her research interests include high performance computing, edge computing, and big data technology.