Overcoming the Gap Between Compute and Memory Bandwidth in Modern GPUs

Lingqi Zhang[1], **Mohamed Wahib**[2], **Toshio Endo**[1], **Satoshi Matsuoka**[2,1]
1. Tokyo Institute of Technology, Japan 2. RIKEN Center for Computational Science, Japan

Observations

Easier Device Saturation

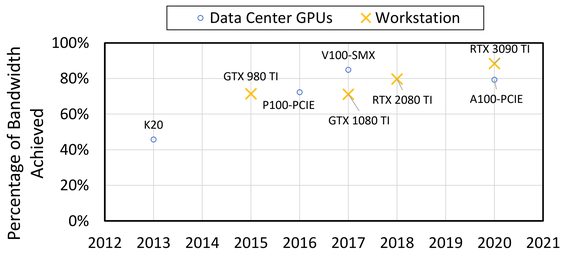
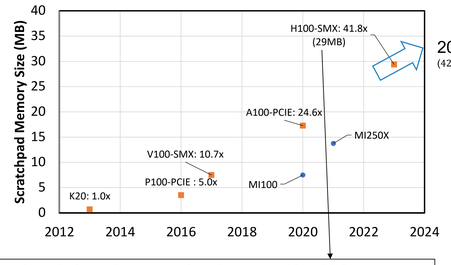


Figure 1: The percentage of performance STREAM kernel achieved with **low occupancy** (256 threads, ILP=4, 8 Byte per memory access).

Larger On-chip Resources

Scratchpad memory as an example:



The size of shared memory has increased by 41.8x over a decade (peak performance has increased by ~22x)

Figure 2: The capacity trend of scratchpad memory.

Mature Device Synchronization [3]

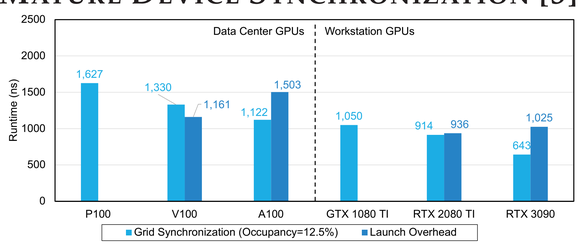


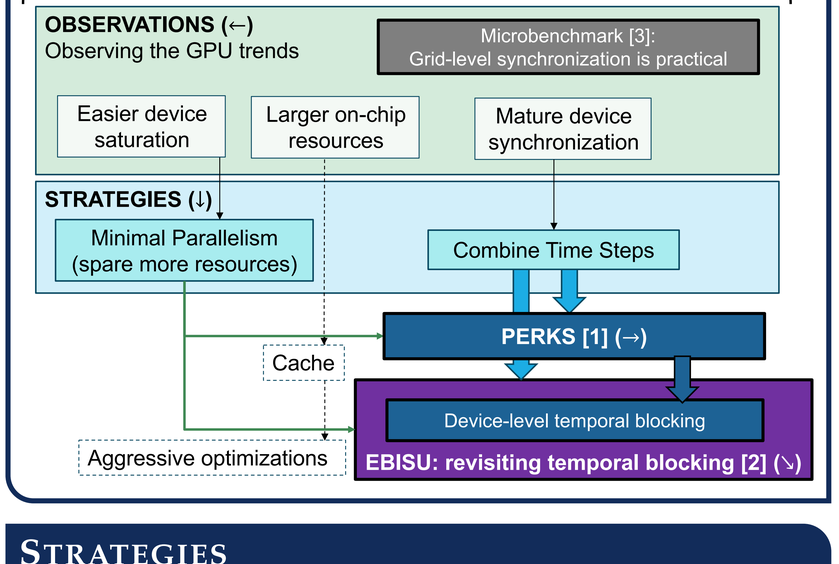
Figure 3: The overhead of device-wide synchronization (grid synchronization as explicit device-wide synchronization; kernel launch as implicit device-wide synchronization in the latest GPUs).

Overview

MOTIVATION

- Compute is faster than memory bandwidth.
- The gap is still widening.
- As GPU architecture evolves, **can we leverage the new features to better overcome the gap?**
- We showcase our methodology with memory-bound kernels (e.g., stencils).

Platform	Launched (Year)	Memory (GB/s)	Compute (TFLOPS/s)	Balance Flops/Bytes
Fujistu (A64FX)	2019	1024	3.4	3.32
Intel (Platinum 8368)	2021	204.8	1.094	5.34
AMD (7773X)	2022	204.8	2.253	11
Nvidia (A100)	2020	1555	9.7	6.24
Nvidia (H100-SMX)	2022	3000	30	10
AMD (MI250X)	2021	3200	47.9	14.97



Strategies

Minimal Parallelism

Orchestrating parallelism.

Little's Law (Hardware)	
$\mathbb{C}$	Concurrency;
$\mathbb{L}$	Latency;
$\mathbb{THR}$	Throughput;
Parallelisms (Software)	
$\mathbb{PAR}$	Parallelism of a program;
ILP	Instruction Level Parallelism;
TLP	Thread Level Parallelism (thread per Stream Multiprocessor);

$$\begin{aligned} \underset{TLP,ILP}{\text{minimize}} \quad & \mathbb{PAR}(TLP, ILP) \\ \text{subject to} \quad & \mathbb{PAR} \geq \mathbb{C} \\ & \mathbb{PAR} = TLP \times ILP \\ & \mathbb{C} = \mathbb{L} \times \mathbb{THR} \end{aligned} \tag{1}$$

Combine Time Steps

Reducing/Eliminating memory traffic.

Roofline	
I	Operation Intensity;
W	Number of works;
Q	Number of bytes of memory traffic;
Additional Parameters	
t	Combined time steps;
$\mathbb{R}$	Percentage of memory traffic reduced in between time steps;

$$I = \frac{W \times t}{Q + (t-1) \times (1-\mathbb{R}) \times Q} \tag{2}$$

Increasing combined time steps t and percent of memory traffic cached $\mathbb{R}$ to increase Operation Intensity I. As such, the memory-bound kernel becomes closer to compute-bound.
The kernel becomes temporal blocking if $\mathbb{R} = 1$.

PERKS [1]

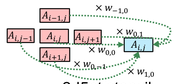
BACKGROUND

Stencil:
Machine Learning
PDE applications:
- Weather modeling
- Fluid dynamics simulation

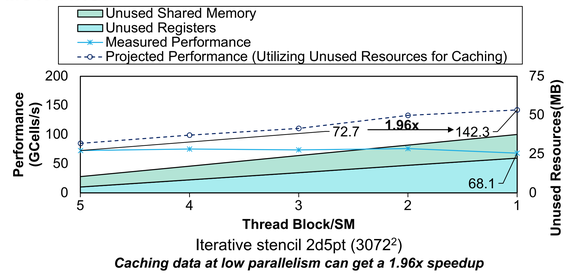
For each grid cell, until $t = T$:

$$A^t(s) = \sum_{a \in N} w_a \times A^{t-1}(s+a) + c$$

s : address, N : concerned neighbours
w : weight, c : constant

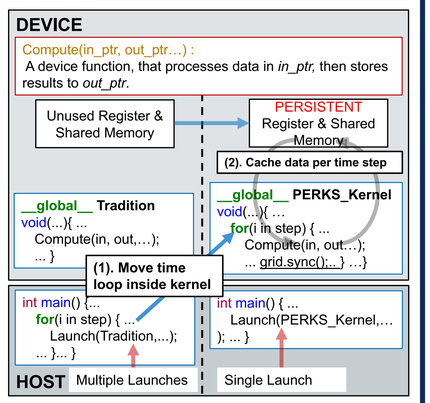


MOTIVATION



Caching data at low parallelism can get a 1.96x speedup

OVERVIEW



EVALUATION

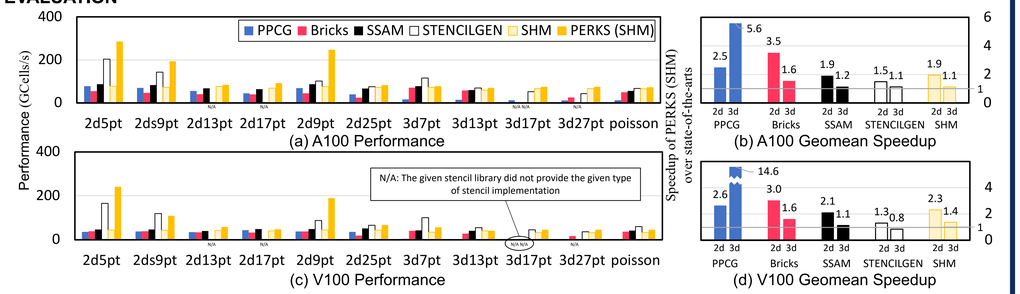
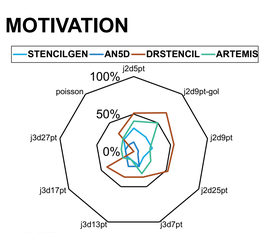


Figure 4: Comparison of PERKS(SHM) over a wide range of stencil libraries.

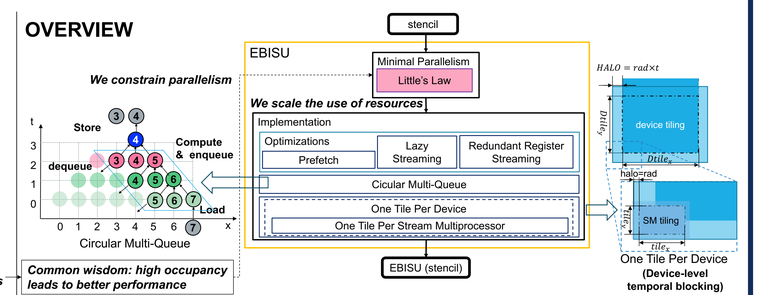
EBISU [2]

MOTIVATION



SOTAs are conservative in using scratchpad memory:
SOTAs constrain resources for parallelisms

OVERVIEW



Common wisdom: high occupancy leads to better performance

EVALUATION

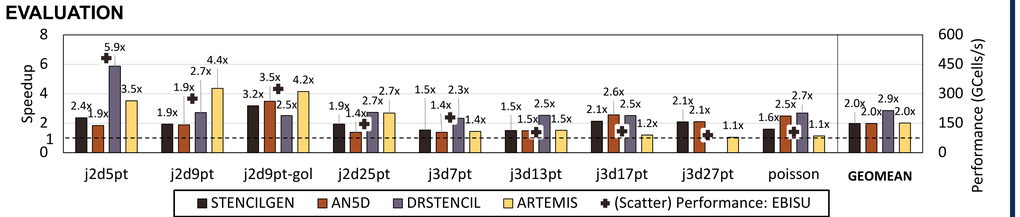


Figure 5: Speedup of EBISU over the state-of-the-art temporal blocking implementations. We also plot the performance of EBISU (right Y-axis plotted as '+' ticks).

References

[1] L. Zhang, M. Wahib, P. Chen, J. Meng, X. Wang, T. Endo, and S. Matsuoka. Perks: A locality-optimized execution model for iterative memory-bound gpu applications. In *Proceedings of the 37th International Conference on Supercomputing*, ICS '23, page 167–179, New York, NY, USA, 2023. Association for Computing Machinery.

[2] L. Zhang, M. Wahib, P. Chen, J. Meng, X. Wang, T. Endo, and S. Matsuoka. Revisiting temporal blocking stencil optimizations. In *Proceedings of the 37th International Conference on Supercomputing*, ICS '23, page 251–263, New York, NY, USA, 2023. Association for Computing Machinery.

[3] L. Zhang, M. Wahib, H. Zhang, and S. Matsuoka. A study of single and multi-device synchronization methods in nvidia gpus. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 483–493, 2020.