Load Balancing in GPU

Masoumeh (Azin) Ebrahimi

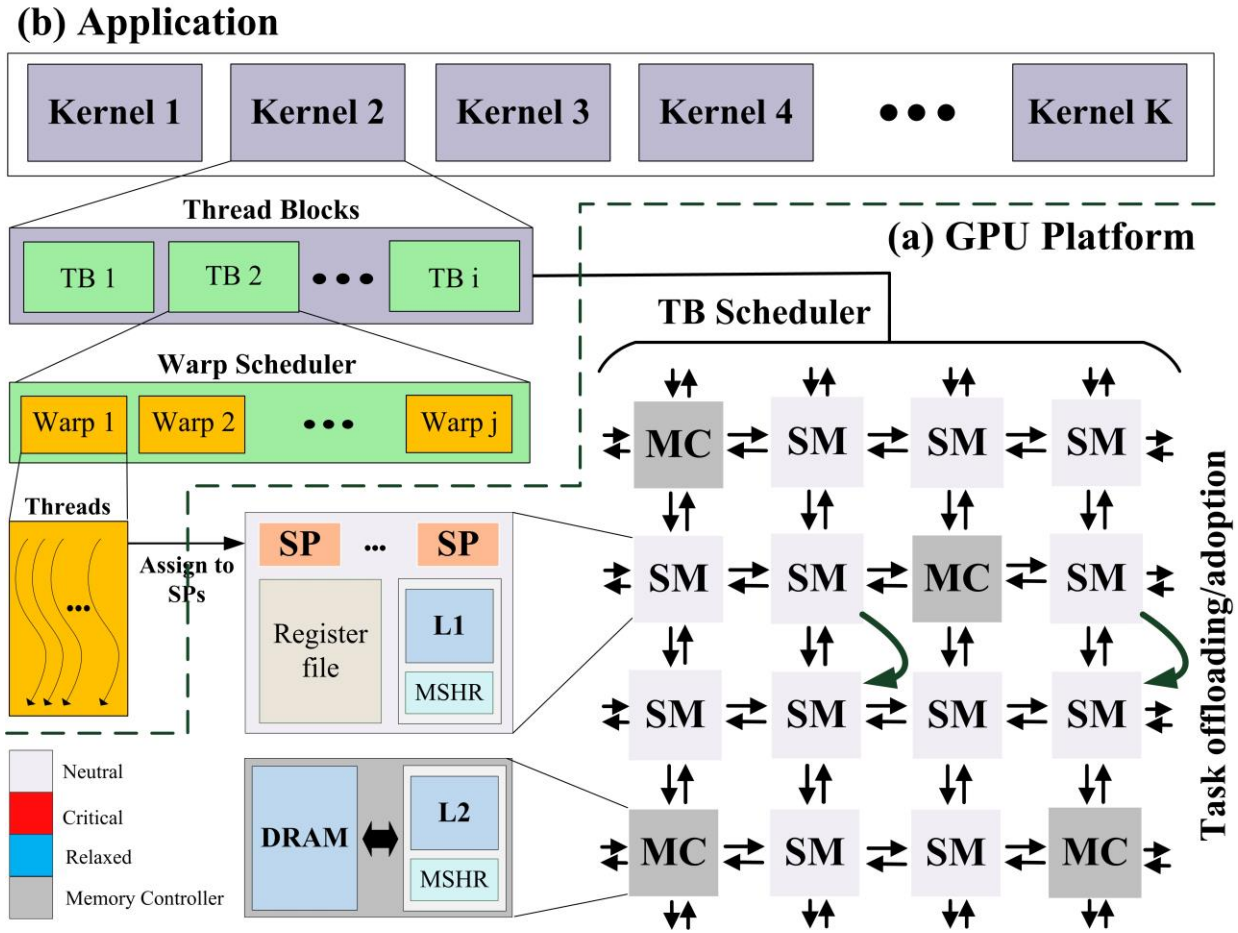
Associate Professor

KTH Royal Institute of Technology, Sweden

Outline

- Background on GPU
- Load balancing
  - Near data processing
  - Processing cores
  - Memory controllers
  - Interconnection networks
- Conclusion

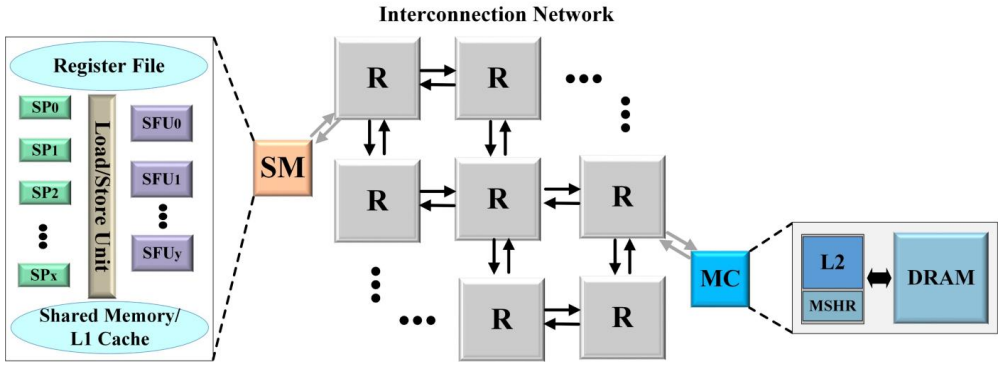
Background on GPU

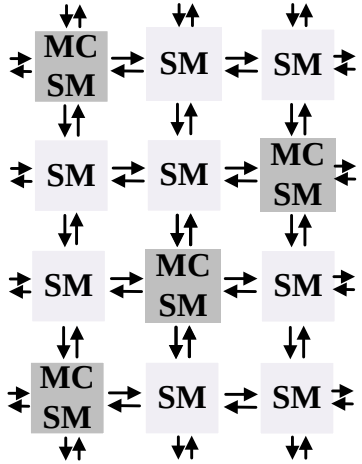


SM: Streaming Multiprocessor
SP: Streaming Processor
TB: Thread Block
MC: Memory Controller

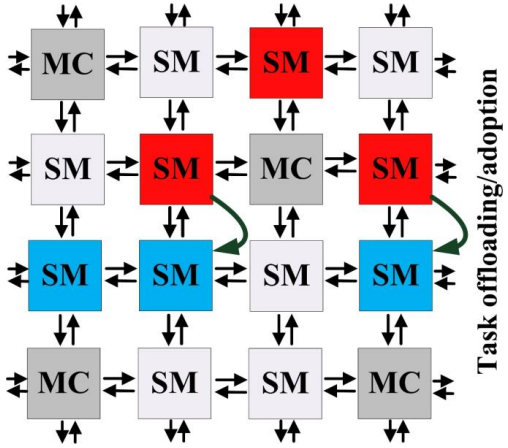
An overall scheme (a) GPU platform; (b) Application running on GPU

Load balancing

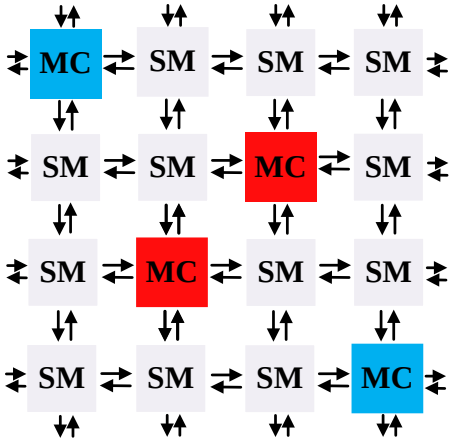




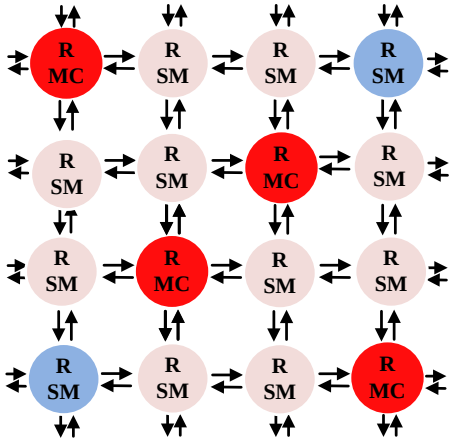
Load balancing through near memory processing



Load balancing among SMs



Load balancing among MCs

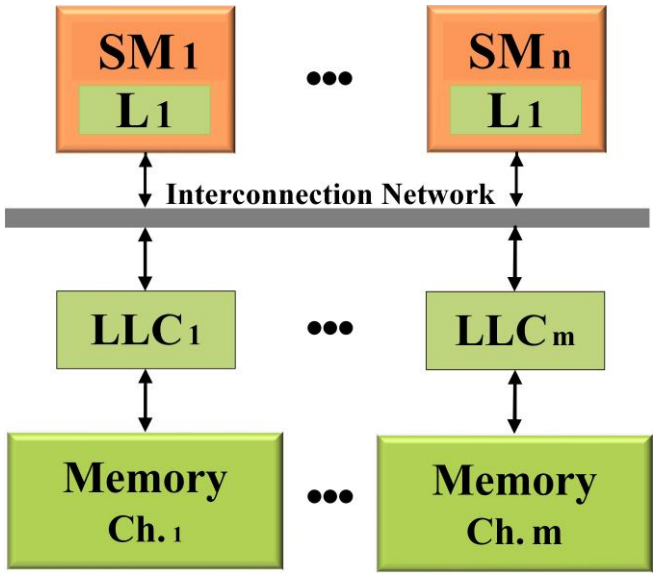


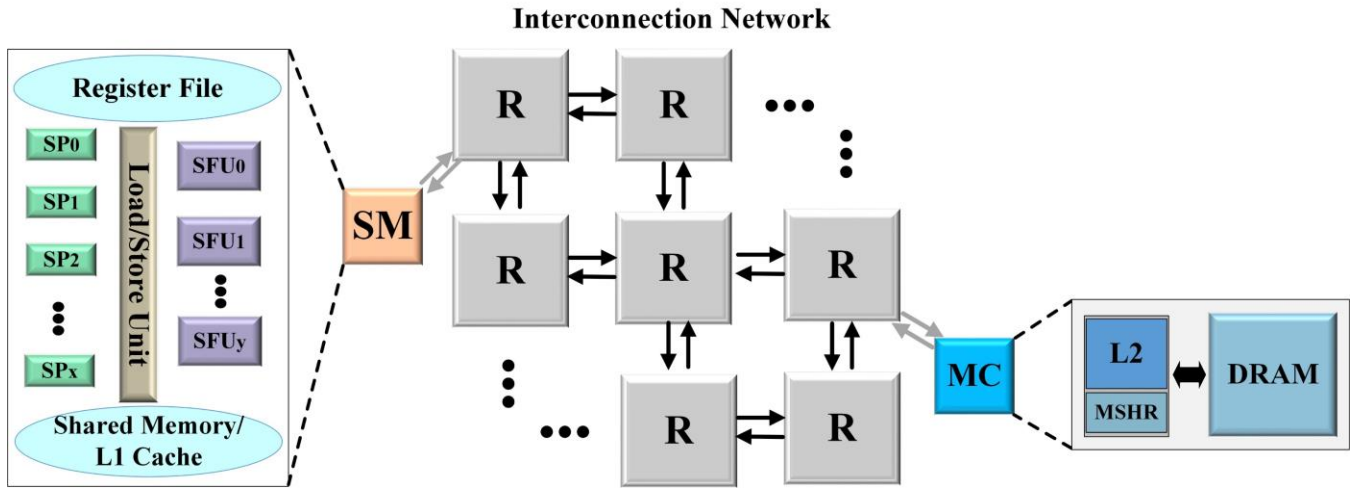
Load balancing in interconnection network

Load balancing through near memory processing

Memory hierarchy

- Each processing core (SM) is equipped with its own private L1 cache.
- L2 or Last level cache (LLC) and main memory are shared among all SMs, and they are accessible through an interconnection network.

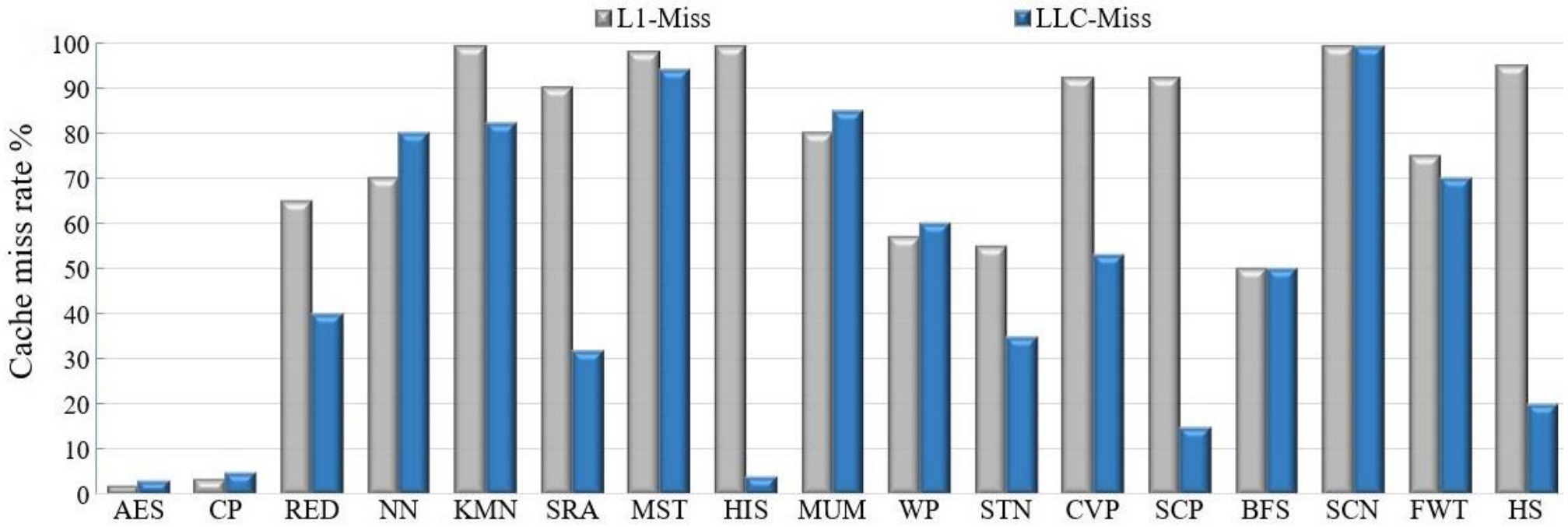




Conventional GPU structure

H. Bitalebi, V. Geraeinejad, and M. Ebrahimi, "Near LLC versus near main memory processing", In Proc. of GPGPU, 2022.

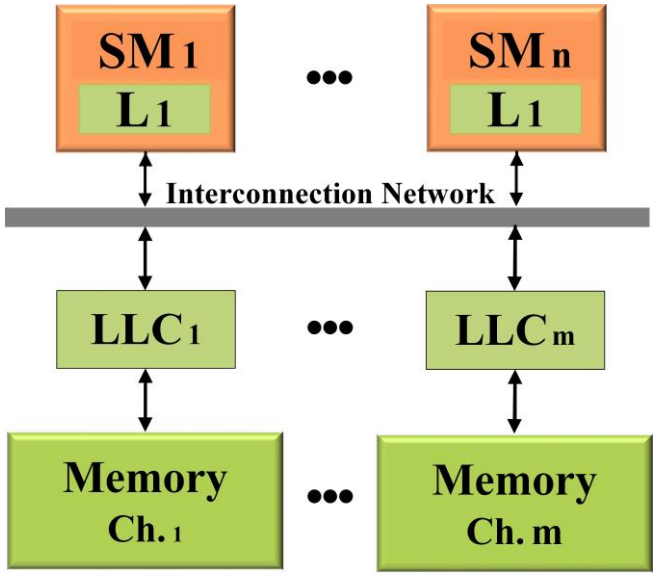
Characterizing applications



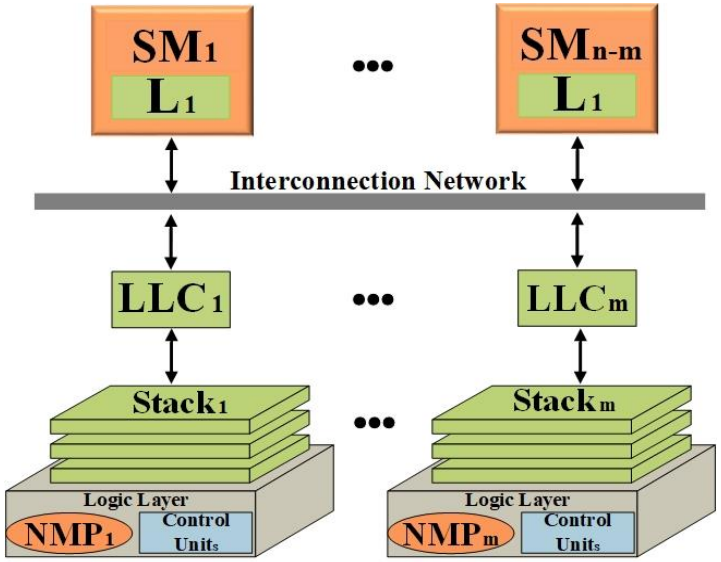
- High L1 and LLC Miss Rate (Avg. L1 Miss: 82%; Avg. LLC Miss: 53%)
- A clear difference between L1 and LLC miss rates in some applications
- Relatively high L1 miss rate and low LLC miss rate in applications such as SRA, HIS, SCP, and HS

H. Bitalebi, V. Geraeinejad, and M. Ebrahimi, "Near LLC versus near main memory processing", In Proc. of GPGPU, 2022.

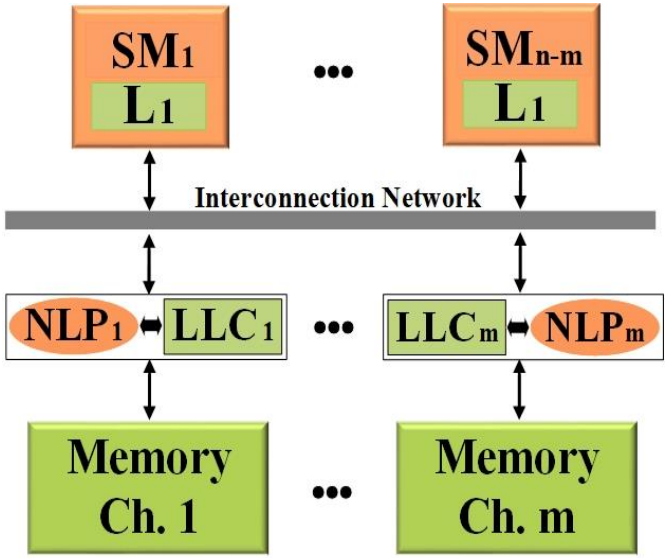
Distribute cores to different memory hierarchy levels



Conventional



Near Memory Processing (NMP)

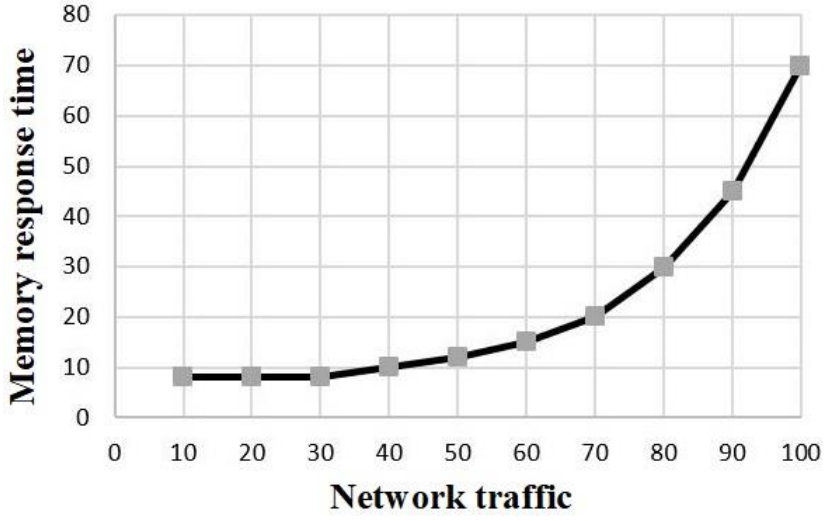


Near LLC Processing (NLP)

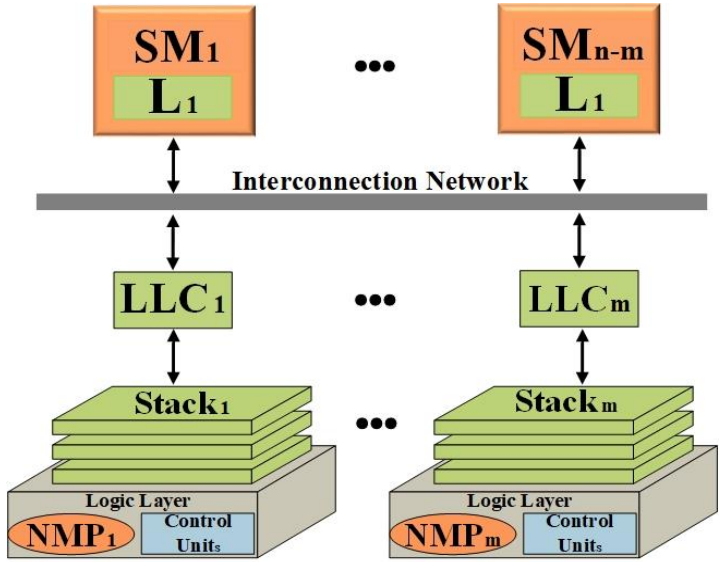
H. Bitalebi, V. Geraeinejad, and M. Ebrahimi, "Near LLC versus near main memory processing", In Proc. of GPGPU, 2022.

Near Data Processing (NDP)

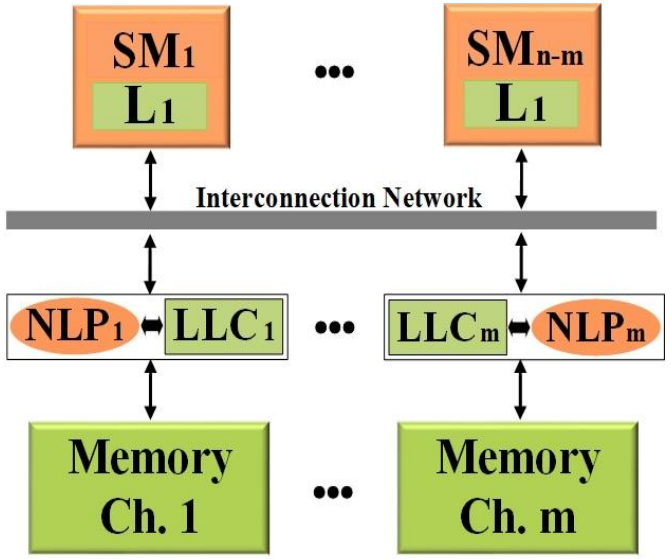
- GPU is a popular platform for compute-intensive applications due to their massive degree of parallel processing
  - How about memory-intensive applications such as DNNs?
- Near data processing (NDP) is usually referred to near main memory processing (NMP)



Impact of interconnection network traffic on GPU memory response time



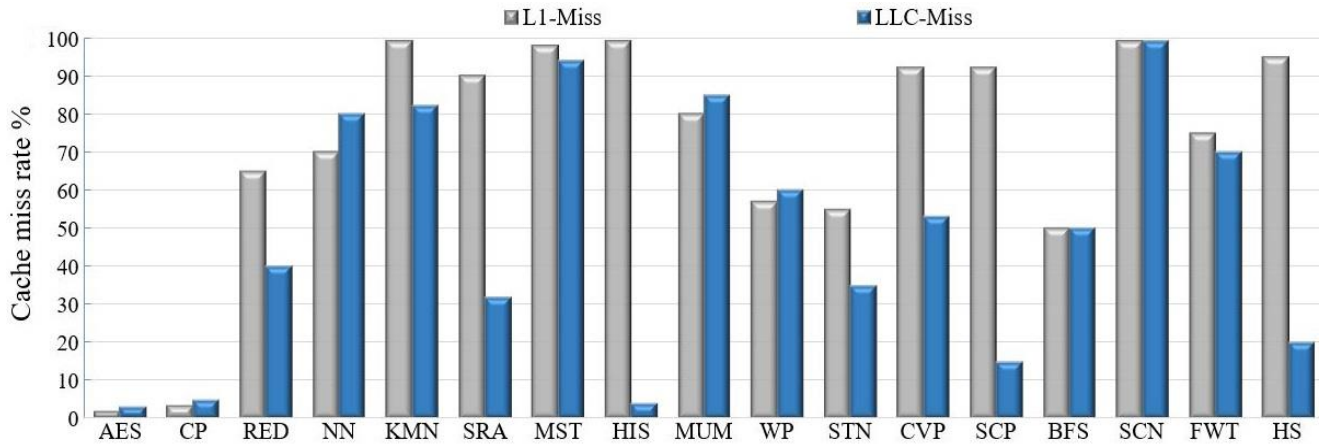
Near Memory Processing (NMP)



Near LLC Processing (NLP)

Classification based on cache access pattern

Miss Access rate	L1	LLC
Class 1	Low	Low
Class 2	High	High
Class 3	High	Low
Class 4	Low	High
Class 5	Moderate	Moderate



- **Class 1 and 4:** the L1 miss rate is low, and data is likely to be hit
  - More suitable for processing in SM cores (SM)
- **Class 2:** high LLC miss rate means a high number of accesses to the main memory
  - More suitable for near main memory processing (NMP)
- **Class 3:** low LLC miss rate, and data is likely to be hit here
  - More suitable for near LLC processing, thus eliminating the long service time of the main memory and shortening the data path (NLP)

Configuration setup

Baseline

Parameter	Value
Total cores	56 Streaming Multiprocessors (SM)
Per core	48 warps, 32 warp width, 8 CTAs, 1536 threads, 32768 registers, 48 KB scratchpad memory
L1 data cache	32 KB, 4-way, 128B block size
LLC (L2) cache	8 x 128K, 16-way
Memory	FR-FCFS scheduler, DDR3-1333H, 8 memory channel
Core, L2 clock	700 MHz, 700 MHz
Interconnect	2D mesh, XY routing, 1 core/node, 4 VCs, 4 routing latency, 1 channel latency

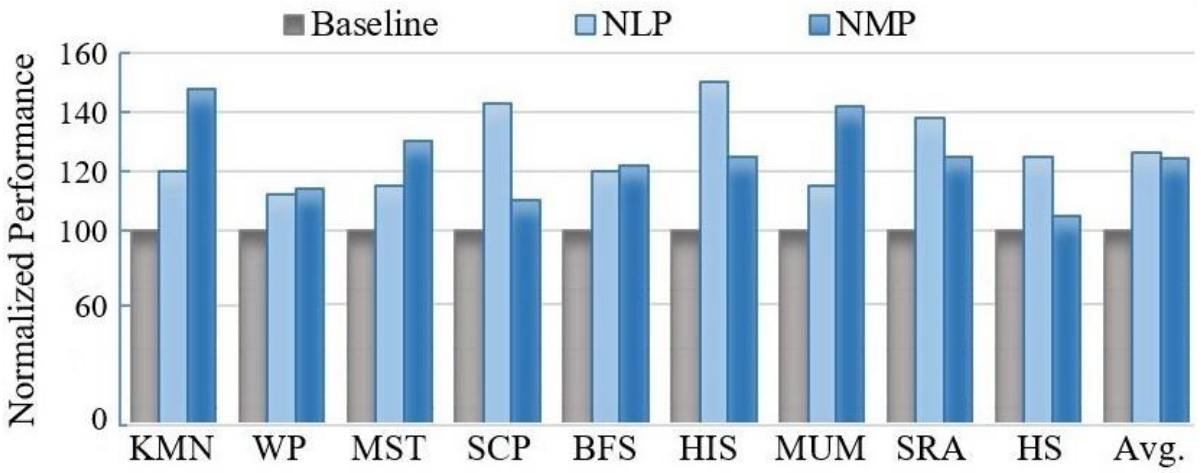
NMP

Parameter	Value
Total cores	56
Main cores	48 SMs
Near main memory cores	8
Memory stack Configuration	8 memory stacks, 16 vaults/stack, 16 banks/vault, 64 TSVs/vault
Intra-stack BW	160 GB/s per stack
Inter-stack BW	40 GB/s per link, fully connected
GPU to memory BW	80 GB/s per link

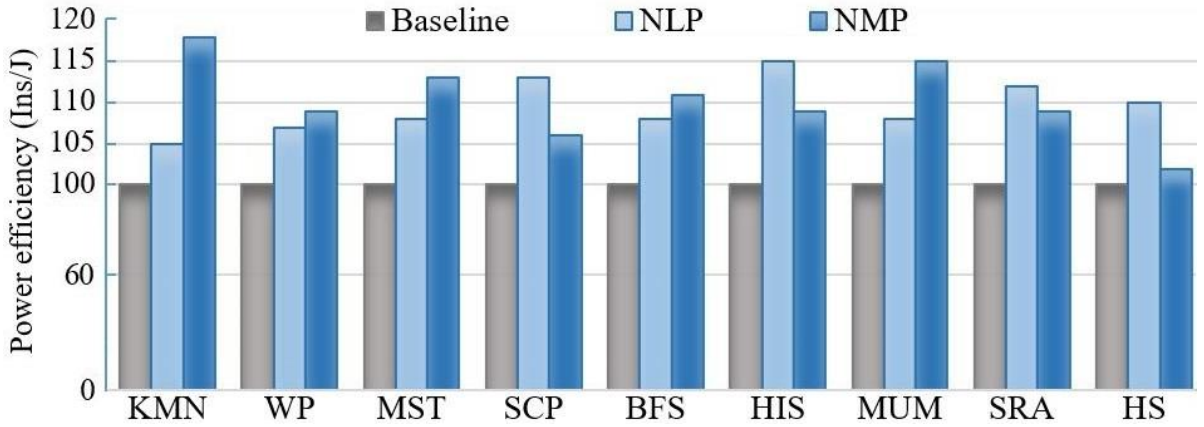
NLP

Parameter	Value
Total cores	56
Main cores	48 SMs
Near LLC cores	8

Performance and power analysis

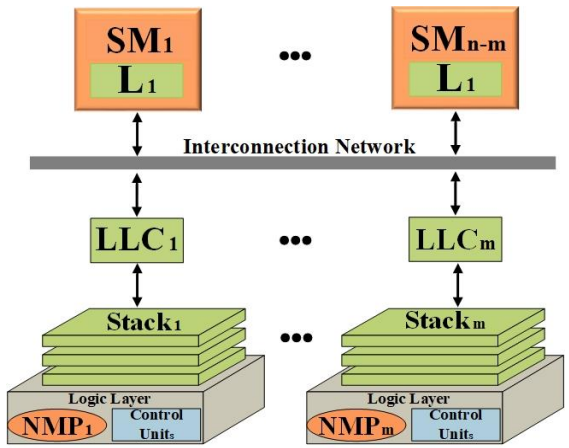


Performance evaluation

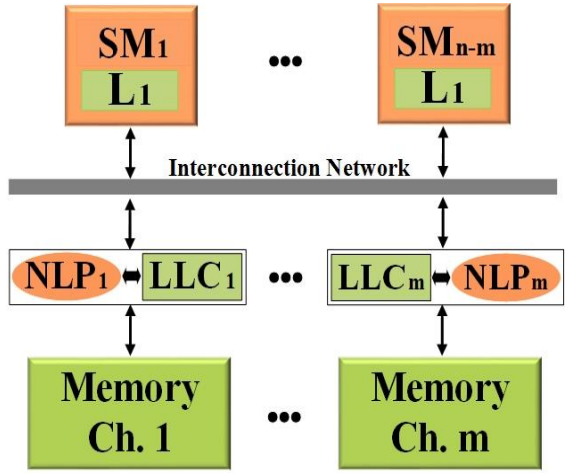


Power evaluation

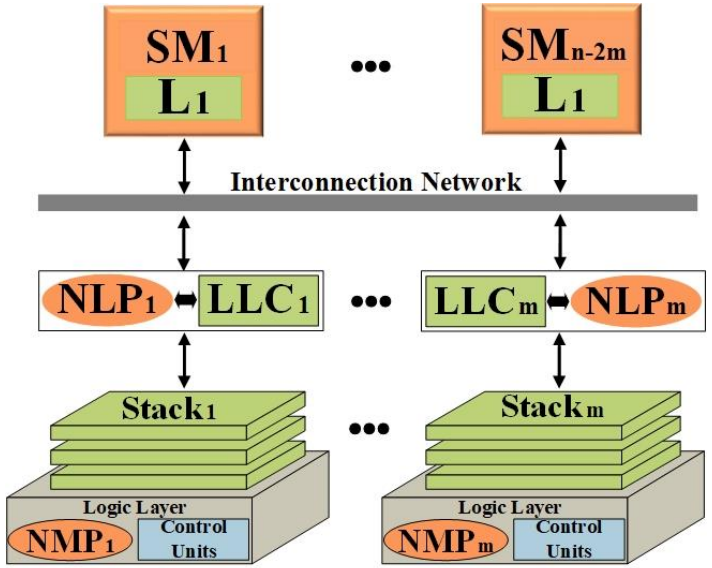
Near data processing architecture



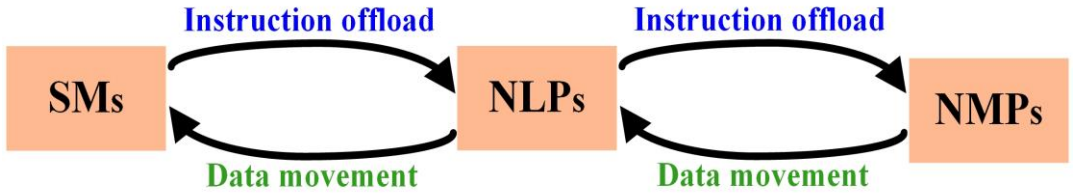
NMP



NLP

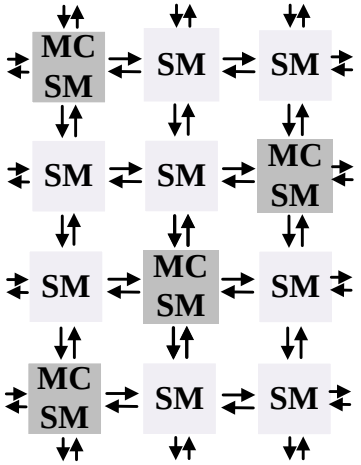


Joint architecture

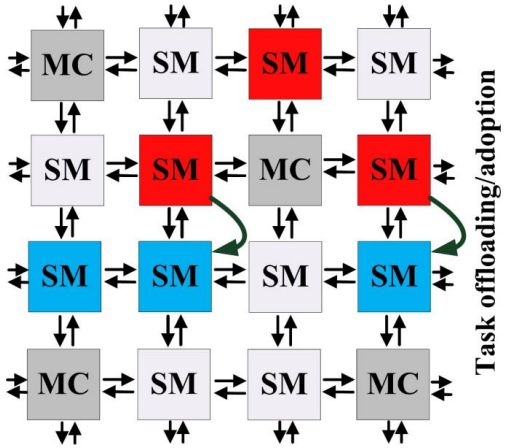


Load balancing among the cores

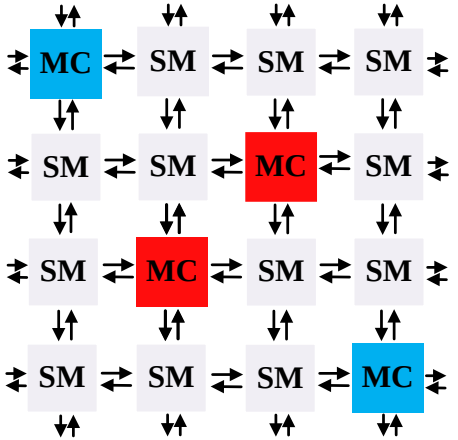
Load balancing



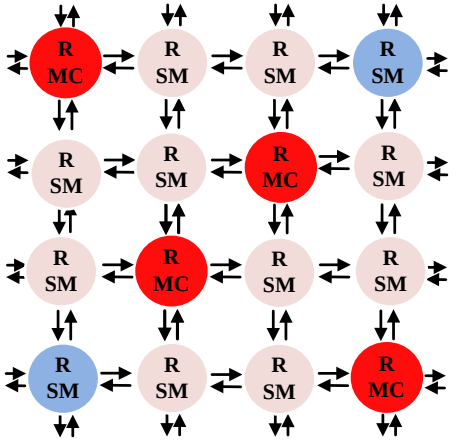
Near memory processing



Load balancing
among SMs



Load balancing
among MCs

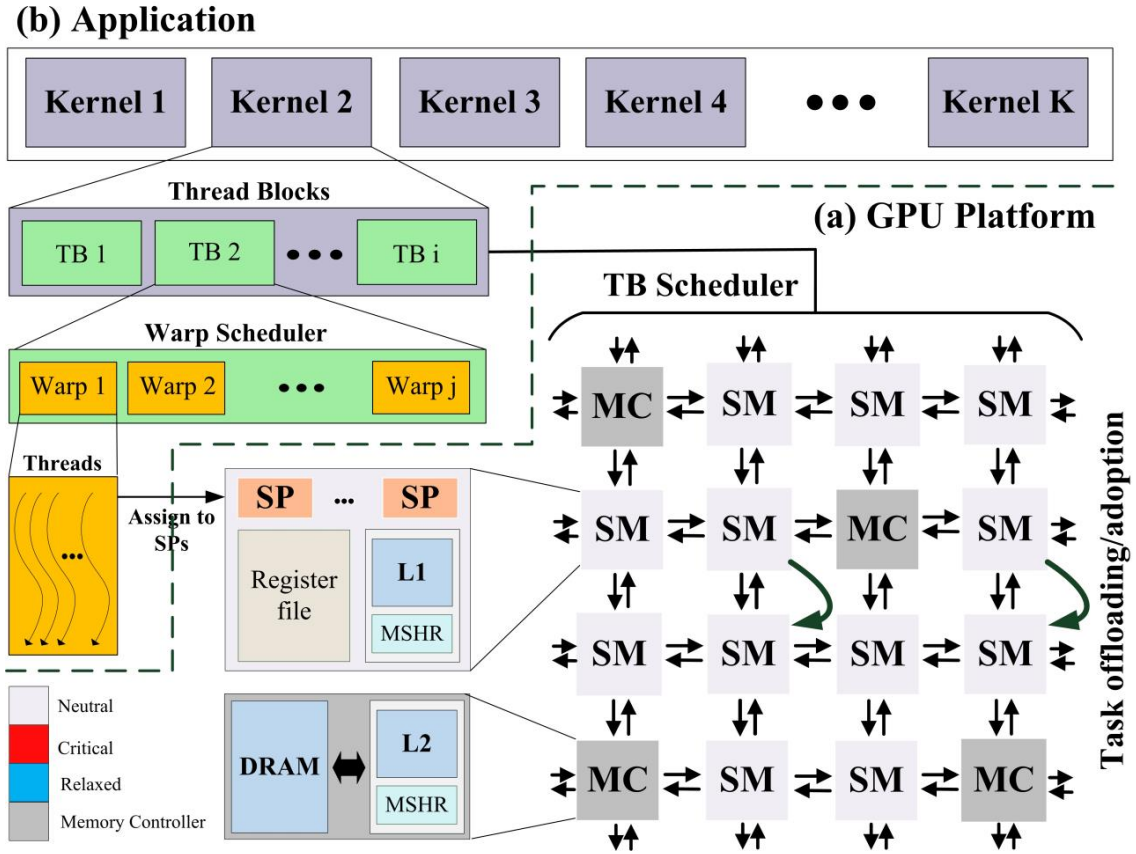


Load balancing in
interconnection network

The impact of core's load unbalance

- If the execution of a warp stalls due to the L1 miss, the processing core proceeds with the execution of the next warp.
- In case all warps stall, waiting for their data from the higher levels of the memory hierarchy, the entire core goes to the stall state.

SM: Streaming Multiprocessor
SP: Streaming Processor
TB: Thread Block
MC: Memory Controller



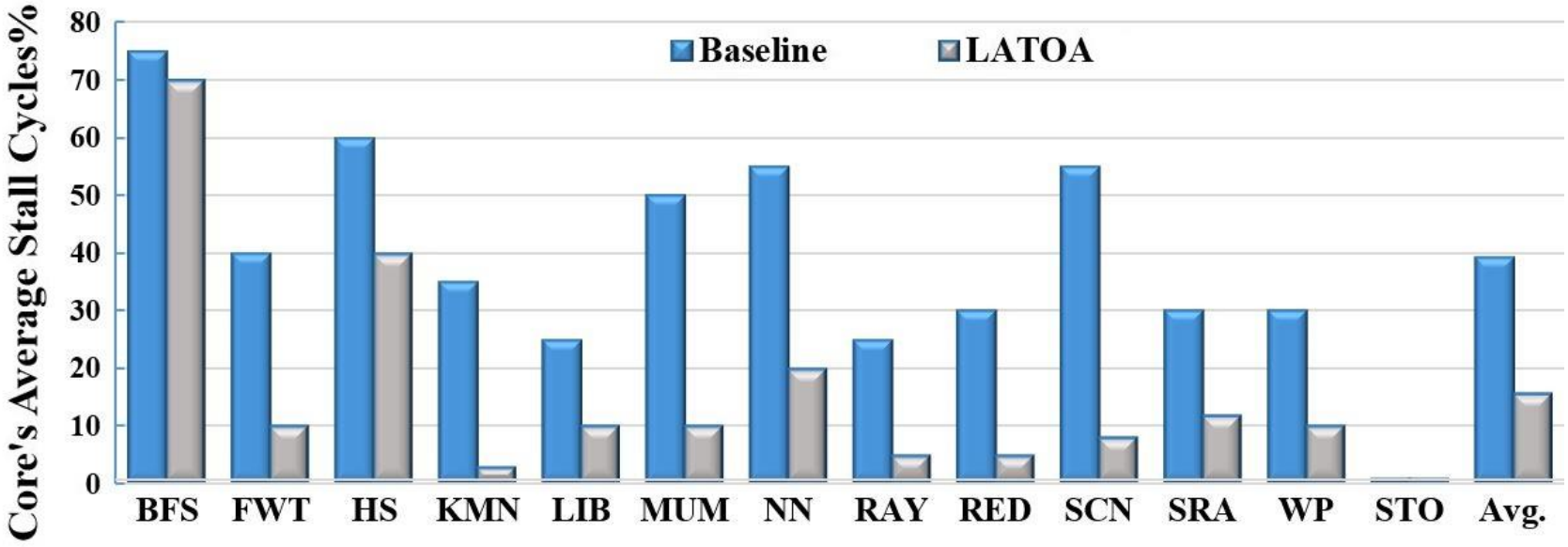
An overall scheme (a) GPU platform; (b) Application running on GPU

H. Bitalebi, V. Geraeinejad, F. Safaei, and M. Ebrahimi, "LATOA: Load-Aware Task Offloading and Adoption in GPU", In Proc. of GPGPU, 2023.

The impact of core's load unbalance

- Different processing cores within a GPU do not follow the same execution time.
  - **Awaiting cores while busy:** usually spend a considerable number of stall cycles waiting for a response from the main memory associated with their irregular memory requests (*busy but stalled*).
  - **Relaxed cores:** perform their duties normally without facing any specific pipeline stalls. These cores have to remain idle until busy cores complete their tasks (*relax and idle*).
- Root of the issue:
  - Conventional TB scheduling methods rely on compile-time information, trying to equally distribute TBs among SMs.

The challange of cores' stall cycles



Average processing cores' stall cycles during the application execution

- The stall rate ranges from 25% in LIB and RAY to around 75% in BFS.
- On average, 40% of cores are stalled, waiting for their memory requests to be responded to.
- LATOA aims to reduce the number of stall cycles through load balancing.

Dynamic load balancing

- LATOA moves from static to dynamic task scheduling based on run-time information obtained from MSHR.

- It offloads warps from critical cores to relaxed ones.
  - **Candidate cores for offloading**
    - The number of not responded memory requests in a processing core is directly linked to the state of the core.
      - The number of filled entries in the **MSHR** table is a reliable indicator that a core is close to go to the stall or idle state.
  - **Candidate warps for offloading**
    - Random warp offloading may hurt the overall performance by imposing extra overhead. It may disturb locality properties.
    - We need to determine the warps that generate irregular memory accesses and offload them into the relaxed cores.
  - **Candidate core for warp adoption**
    - Choose among the neighboring cores to reduce traffic

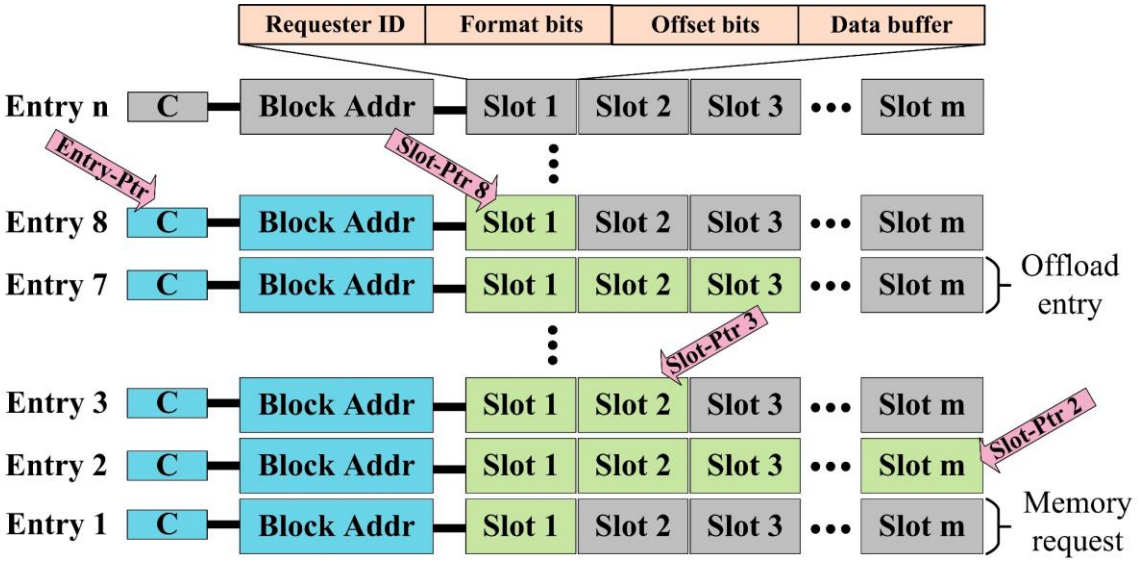
H. Bitalebi, V. Geraeinejad, F. Safaei, and M. Ebrahimi, "LATOA: Load-Aware Task Offloading and Adoption in GPU", In Proc. of GPGPU, 2023.

Candidate cores for offloading by using MSHR

Cache Misses:

1) **Primary miss:** the existing cache lines do not contain the newly requested address, and thus a **new entry** must be assigned to the new request.

2) **Secondary miss:** the newly requested address shares the same cache line as the previous miss(es). Thereby a **new slot** in the matching entry will be allocated to the request.

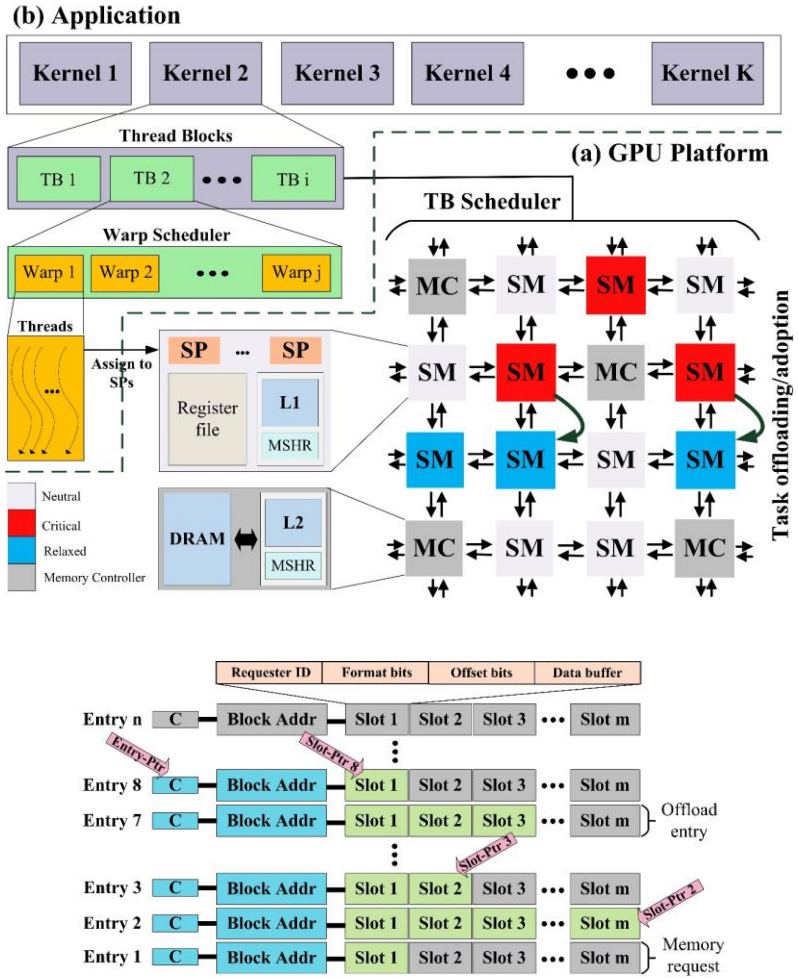


Miss Status Holding Register (MSHR)

H. Bitalebi, V. Geraeinejad, F. Safaei, and M. Ebrahimi, "LATOA: Load-Aware Task Offloading and Adoption in GPU", In Proc. of GPGPU, 2023.

Taging cores based on MSHR information

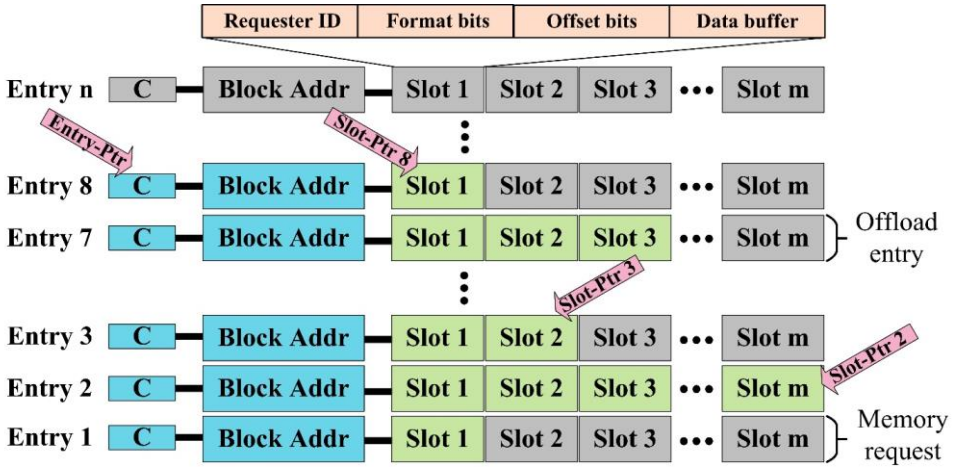
- We tag the cores with three states as **Critical**, **Neutral**, and **Relaxed** by dividing the number of MSHR entries into one of the three states:
  - **Relaxed core (blue)** may enter an idle state. The cores with the filled entries between zero and 1/3 of the total number of MSHR entries.
  - **Neutral core (gray)** is neither critical nor relaxed. The cores with filled entries between 1/3 to 2/3.
  - **Critical core (red)** is likely to switch to the stall state. The cores with filled entries of more than 2/3.



H. Bitalebi, V. Geraeinejad, F. Safaei, and M. Ebrahimi, "LATOA: Load-Aware Task Offloading and Adoption in GPU", In Proc. of GPGPU, 2023.

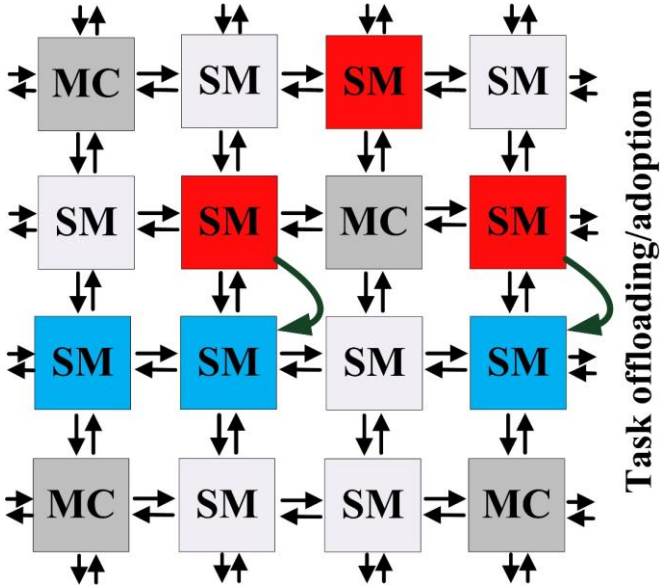
Candidate warps for offloading

- Regular instructions
  - Memory requests are synchronized with most existing requests (i.e., high locality rate among different memory requests), and thus reusing data blocks.
  - Consequently, MSHR slots are filled up rather than entries.
- Irregular instructions
  - The cache fails to work efficiently with irregular parts of the application, and the rate of cache misses dramatically increases.
  - Consequently, MSHR entries are quickly filled up rather than slots.

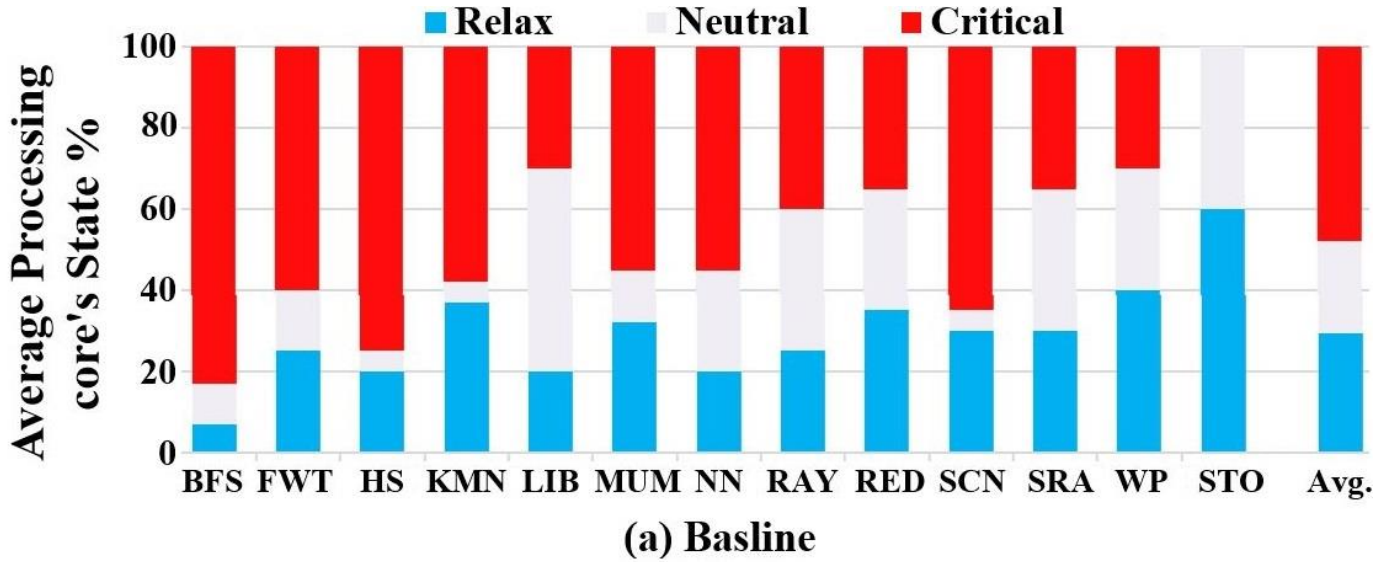


Candidate core for warp adoption

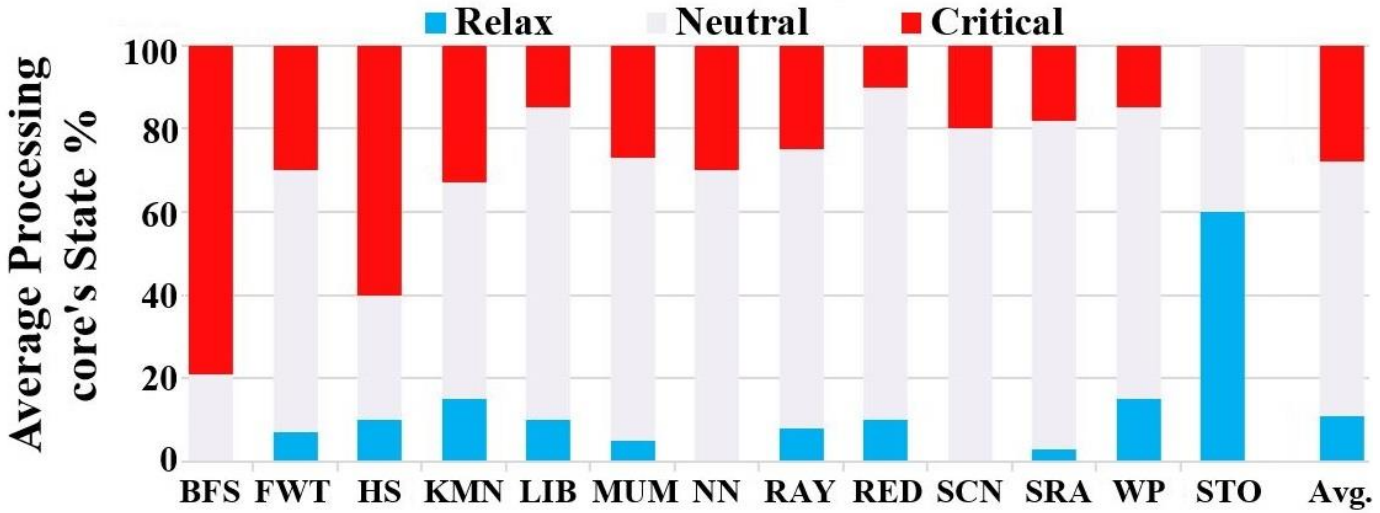
- Offloading a warp without considering the core's position in the network may lead to additional network traffic.
- To minimize the offloading overhead to the interconnection network, relaxed cores will only be selected among the neighboring cores.
- Each core is equipped with a 4-bit register to hold the state of its neighbors. 1-bit is allocated to each neighbor to indicate the core is relax or not.



Cores' states during execution

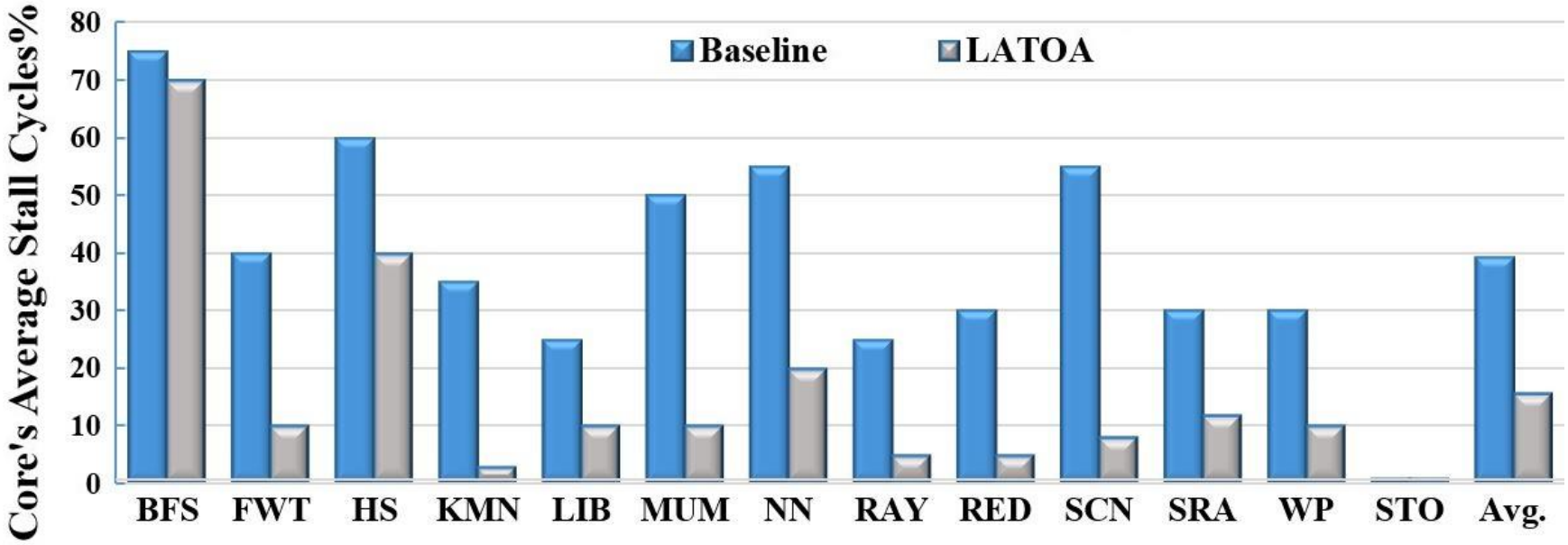


(a) Basline



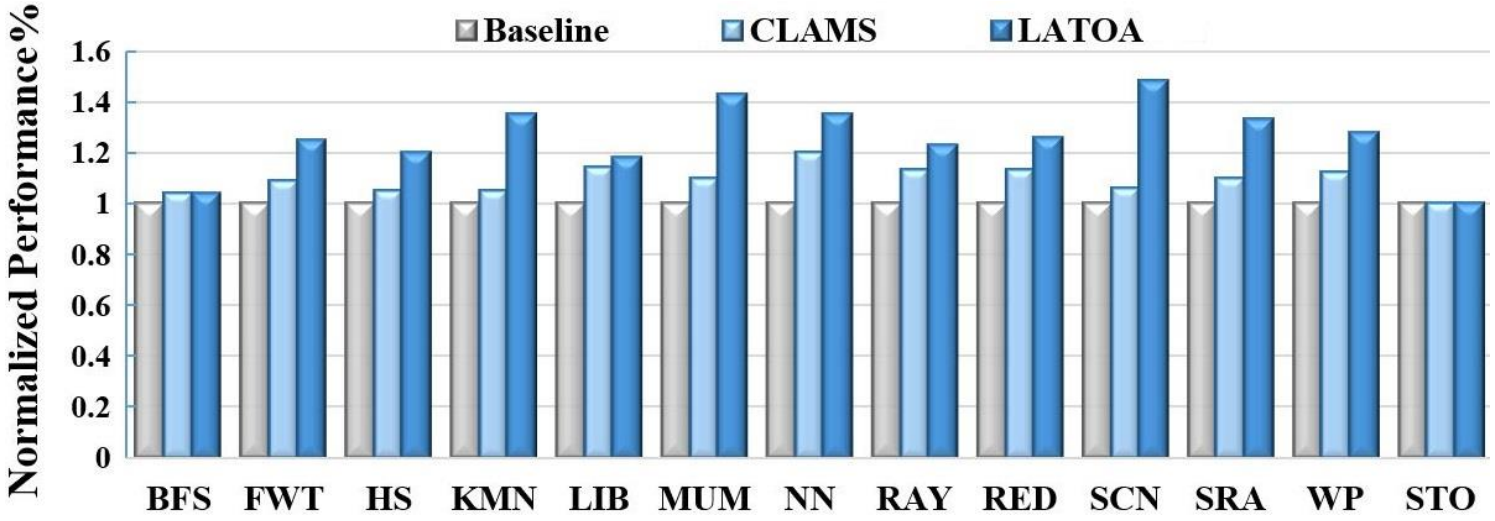
(b) LATOA

Cores’ stall cycles



Average processing cores’ stall cycles during the application execution

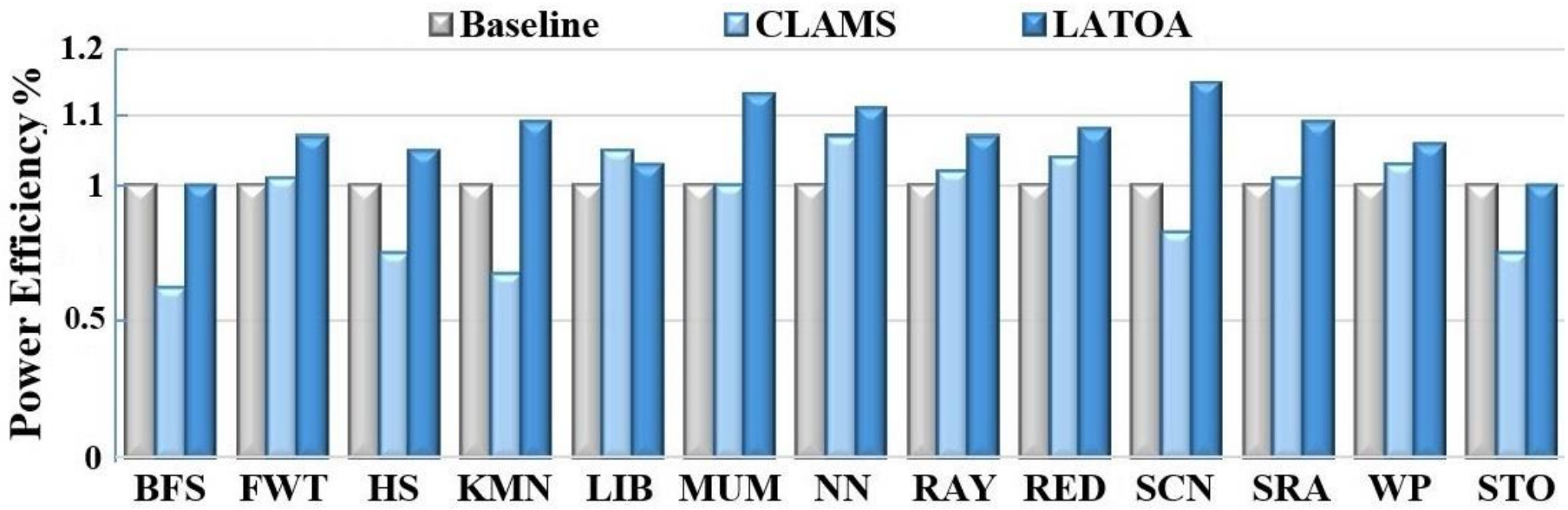
Performance evaluation



Performance evaluation (IPC)

- Applications like FWT, KMN, MUM, NN, and SCN are memory intensive. Also they show unbalanced behavior where some cores are relaxed while others are critical.
- RAY is heavily sensitive to locality property. LATOA improves the locality property of processing cores, thus leaving more space to hold in-use data blocks.
- STO is severely process intensive, and most of its requested data is provided by the shared memory. So, LATOA is unable to improve its performance.

Power evaluation

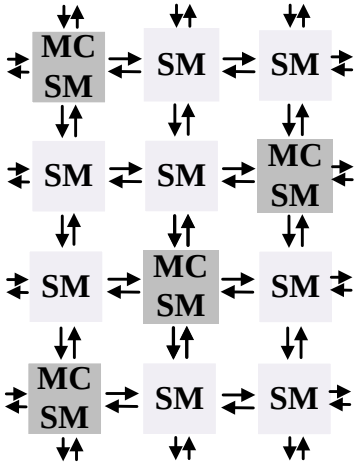


Power evaluation (Ins/J)

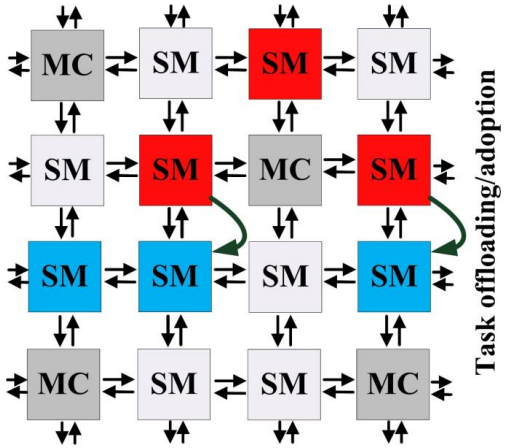
- LATOA succeeds to reduce the power consumption, on average, by 7%.
- This is mainly due to increased data block reusability in critical cores and lower static and leakage power consumption because of the reduced stalled and idle cycles.

Load balancing in Interconnection network

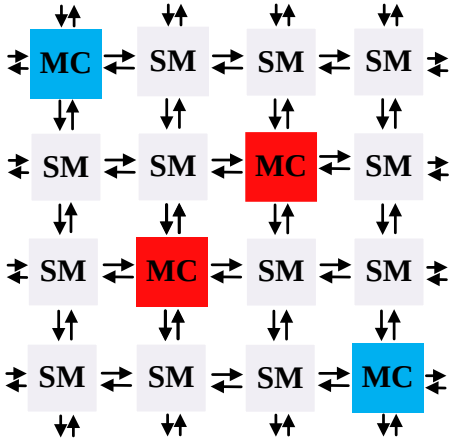
Load balancing



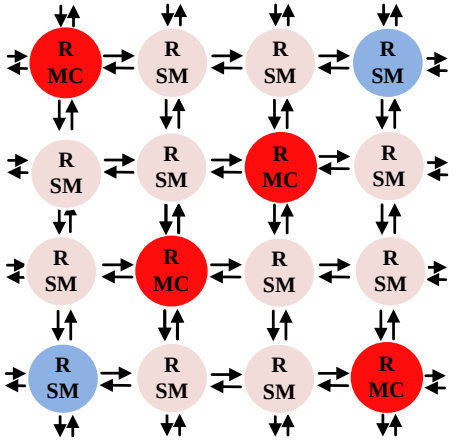
Near memory processing



Load balancing
among SMs



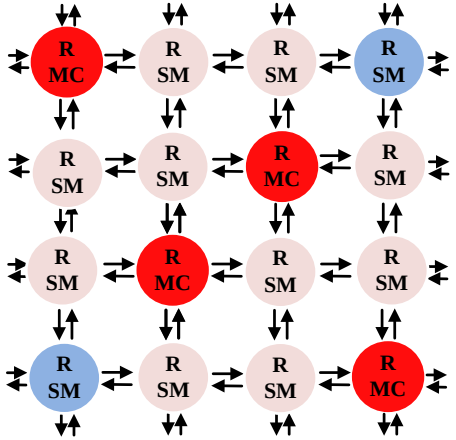
Load balancing
among MCs



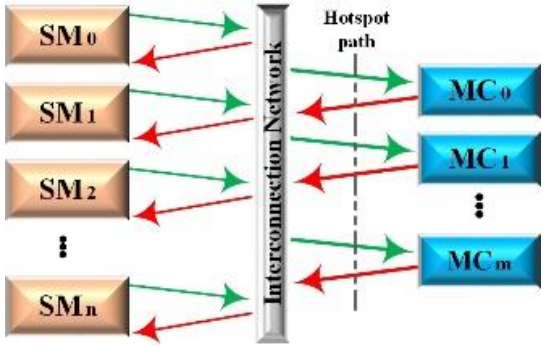
Load balancing in
interconnection network

Load balancing in interconnection network

- Moving from deterministic routing algorithm to adaptive routing algorithms?
- How to map SMs and MCs on the grid?
- How to allocate memory partitions to MCs?



Load balancing in interconnection network



As all SM cores communicate with several shared MC nodes, hotspot is usually formed around the MC nodes.
Many-Few-Many

Thank you!