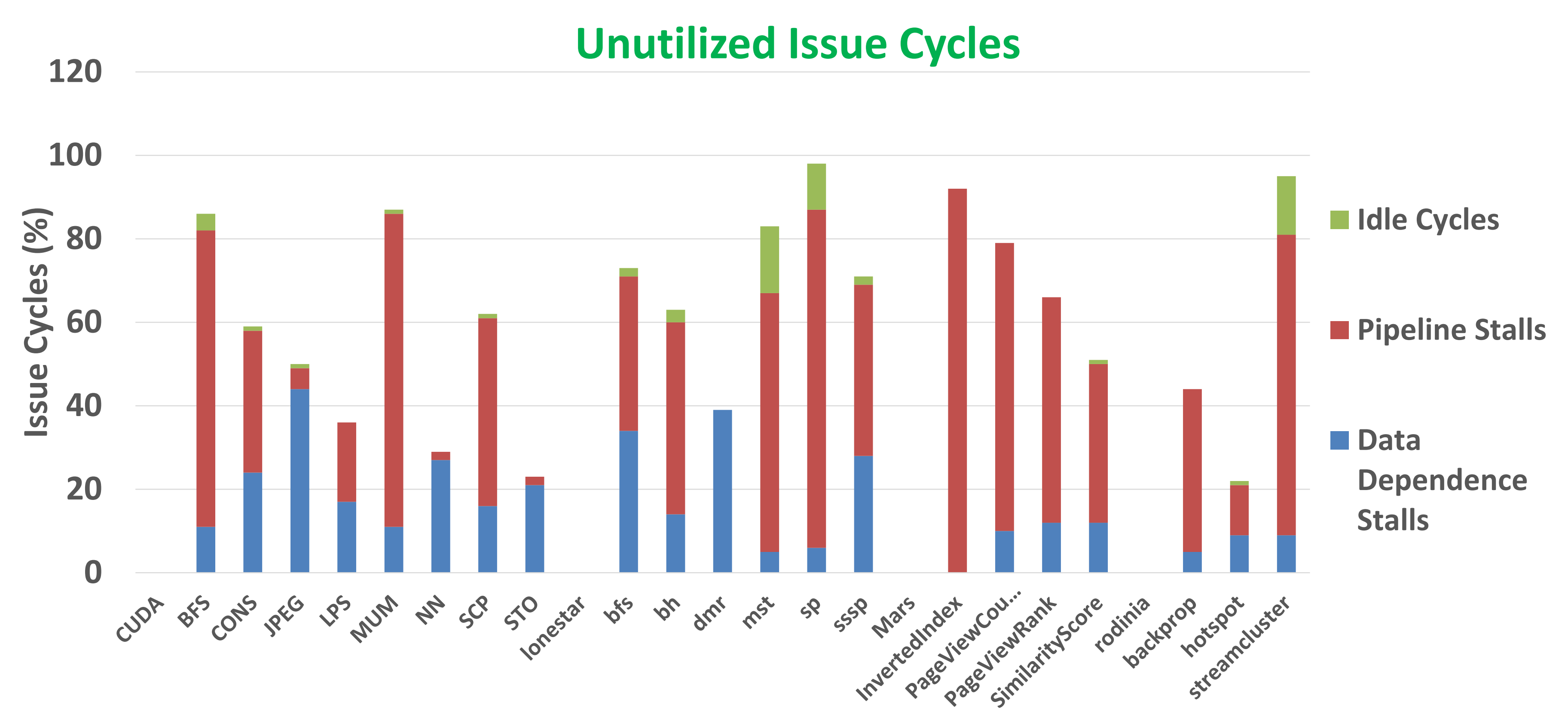
Shader-Assisted Dynamic Bottleneck Acceleration

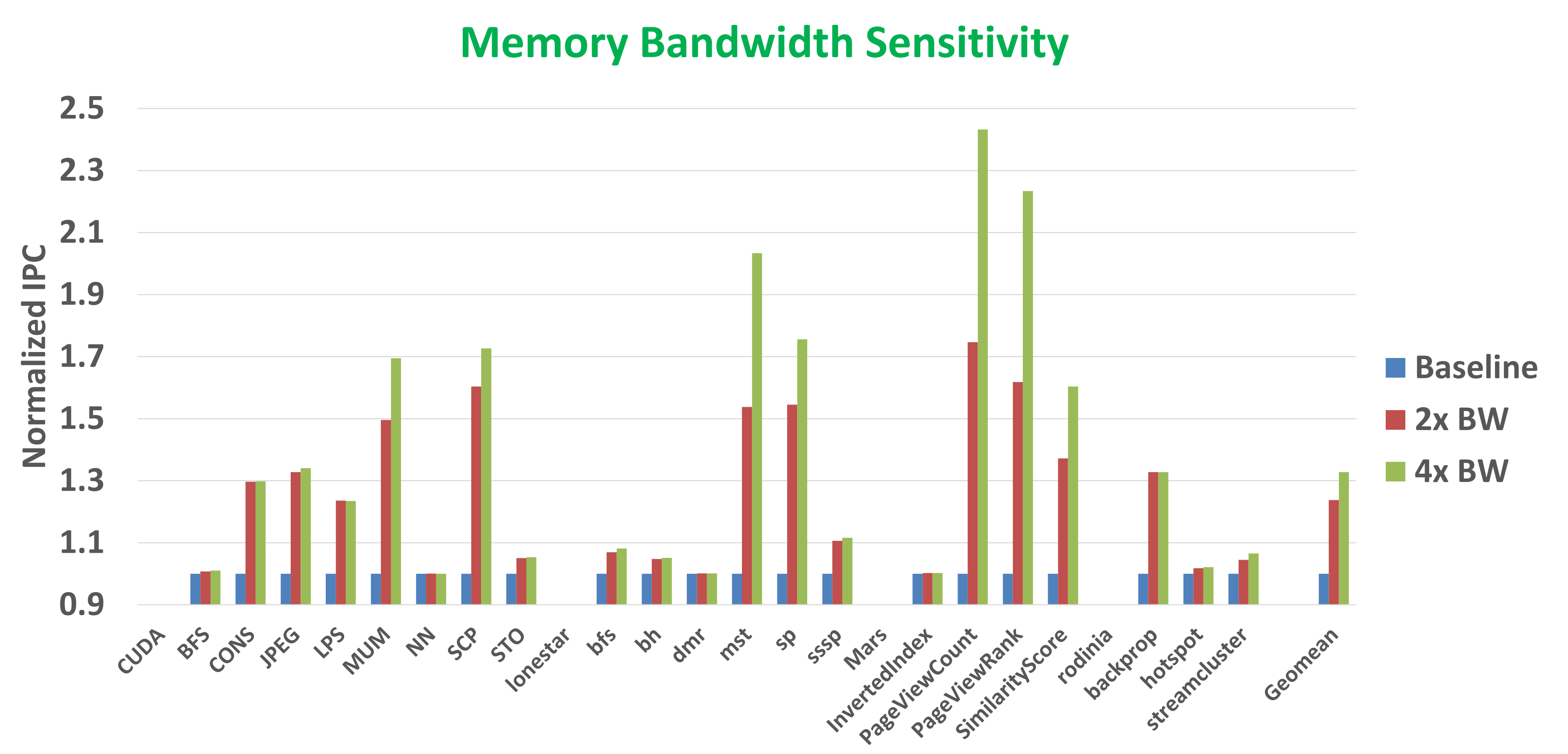
Nandita Vijaykumar, Gennady Pekhimenko, Adwait Jog, Abhishek Bhowmick, Onur Mutlu (CMU)

Idle Resources in GPUs



- A lot of stalls and idle time in the shaders
- Often there are insufficient warps to hide the memory/computational latencies

The Memory Bandwidth Bottleneck



- Performance is often dictated by available memory bandwidth
- Memory accesses are in bursts of both fine and coarse grained phases

Observations

- A significant amount of time is spent waiting for data from memory
- Stalls are also due to long latency ALU operations and resource contention
- Compute units are idle during this time
- *Idle cores can be used to perform useful computation*

Accelerating Bottlenecks

Idle pipelines in the cores can be used to accelerate bottlenecks

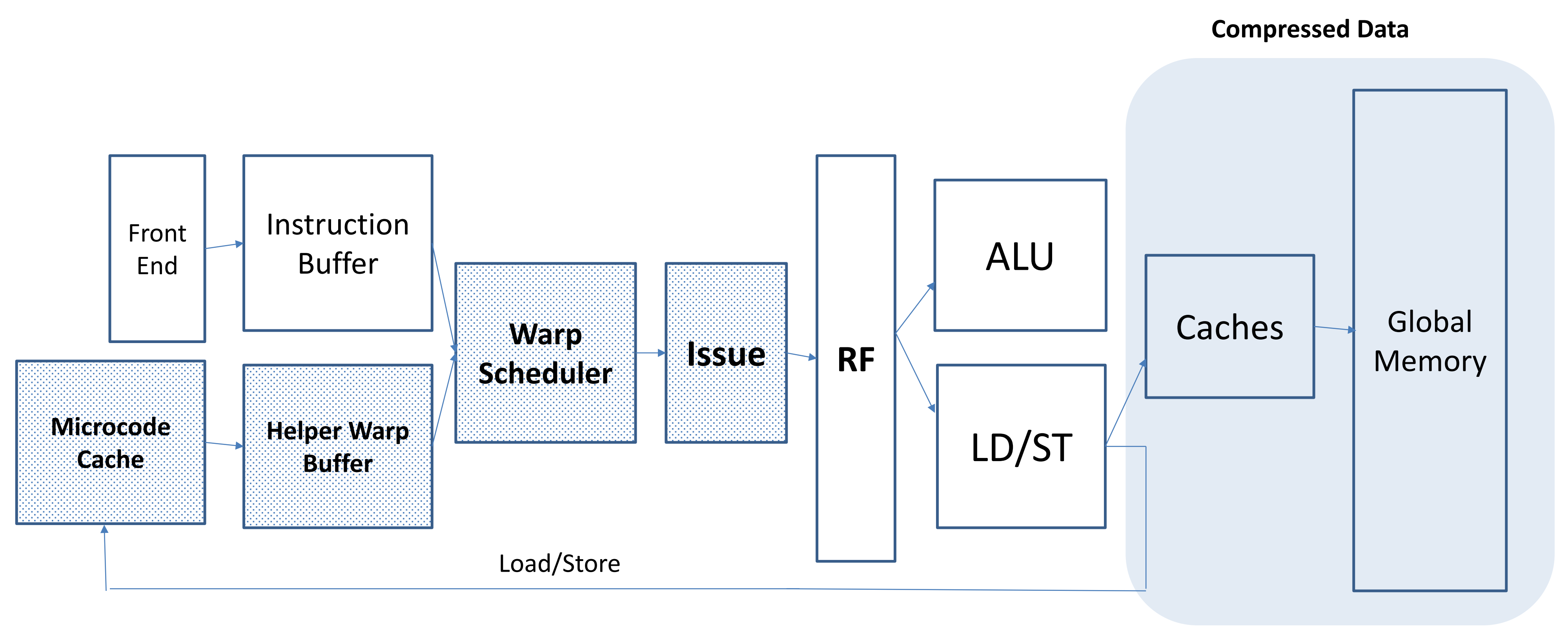
➢ **Memory limited**
  - Data Compression
  - Redundant/Speculative Execution
  - Scheduling

➢ **Computationally limited**
  - Memoization
  - Prefetching

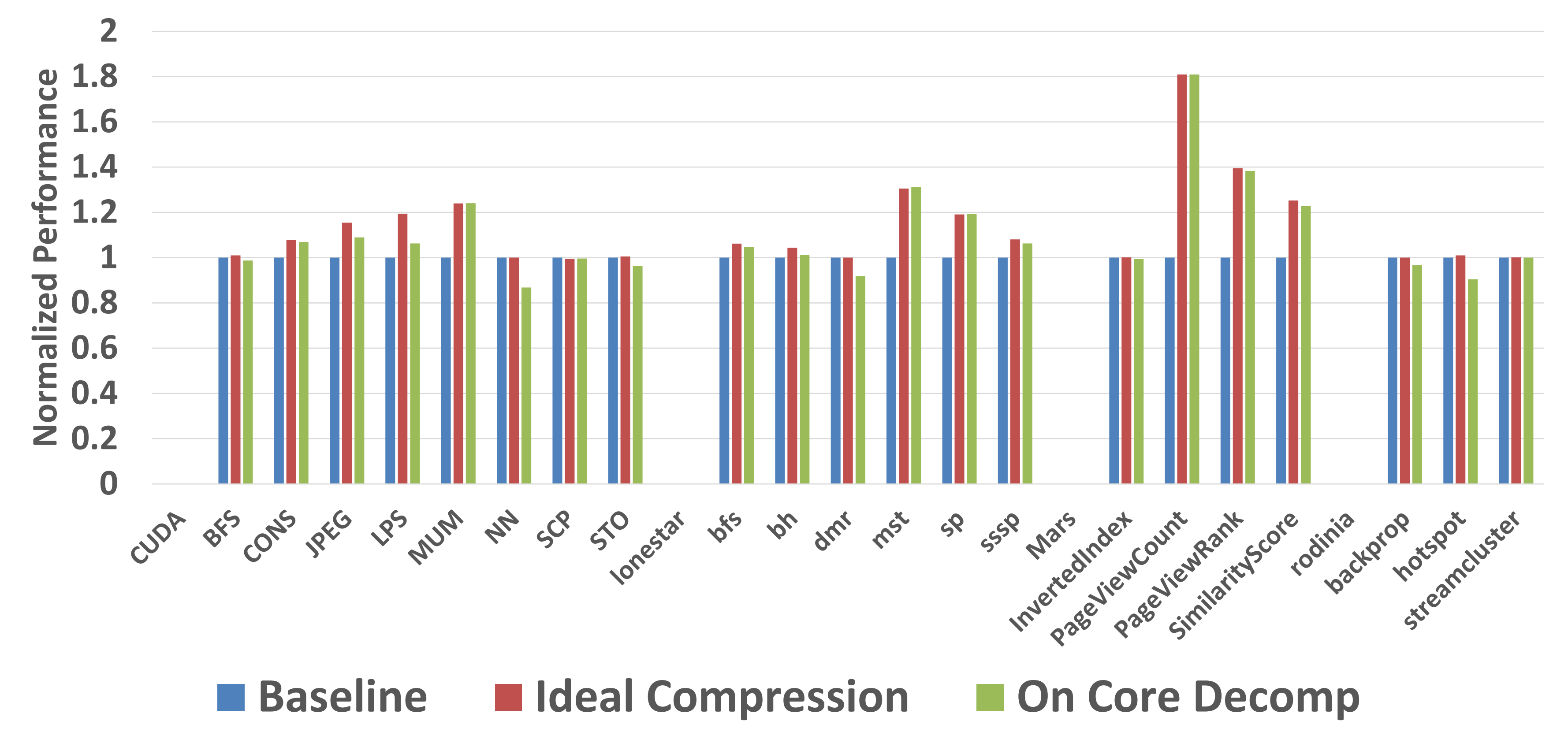
Shader Assisted Data Decompression

Idle shaders can be used to perform computationally intensive decompression algorithms

➢ **Mechanism**
  - Convert decompression algorithm into multiple simple instructions
  - Retrieve compressed data into registers
  - Spontaneously generate a helper warp to perform decompression



Performance Impact



Conclusion

➢ **Conclusions**
  - Helper warps can be inserted without significant performance loss.
  - Data in global memory is highly compressible
  - Data Compression helps alleviate the bandwidth problem

➢ **Advantages**
  - No dedicated logic required
  - Flexibility in algorithm
  - Reduced interconnect bandwidth usage
  - Flexibility to compress at different levels of memory hierarchy

Intel Science & Technology Center for Cloud Computing

Carnegie Mellon University Georgia Tech intel PRINCETON UNIVERSITY UC Berkeley UNIVERSITY of WASHINGTON