Optimizing GPU-Accelerated Similarity Joins: Addressing Data-Dependent Workloads

Benoît Gallet, Michael Gowanlock
benoit.gallet@nau.edu, michael.gowanlock@nau.edu
School of Informatics, Computing and Cyber Systems, Northern Arizona University

Abstract

The distance similarity self-join finds all pairs of objects that are within a distance ε from each other. The data-dependent nature of this application, combined with the Single Instruction Multiple Threads (SIMT) architecture of the GPU, can lead to severe workload imbalance between GPU threads, resulting in a loss of performance due to idle periods. We thus propose to balance the workload by sorting the points based on their workload, and by executing the points in a specific order by using a work queue.

Motivation

- GPU's architecture: 32 threads executed simultaneously (warp)
  - Divergent paths executed sequentially
- Different workloads within a warp
  - Threads may idle for a period of time
- Unused computational power leads to higher execution time
- Reduce workload imbalance to improve performance
  - Both execution time and GPU's resource utilization

Introduction

- For a query point *q*, find all its neighboring points within a Euclidean distance ε (also called a range query)
- For a dataset *D*, there are |*D*| total range queries
- Range queries are independent and memory intensive
  - Suited to the GPU
- Every range query does not have the same number of distance calculations than the others
  - Different workloads, thus execution time (Fig. 1)
- The workload of a single range query is characterized by the number of distance calculations computed

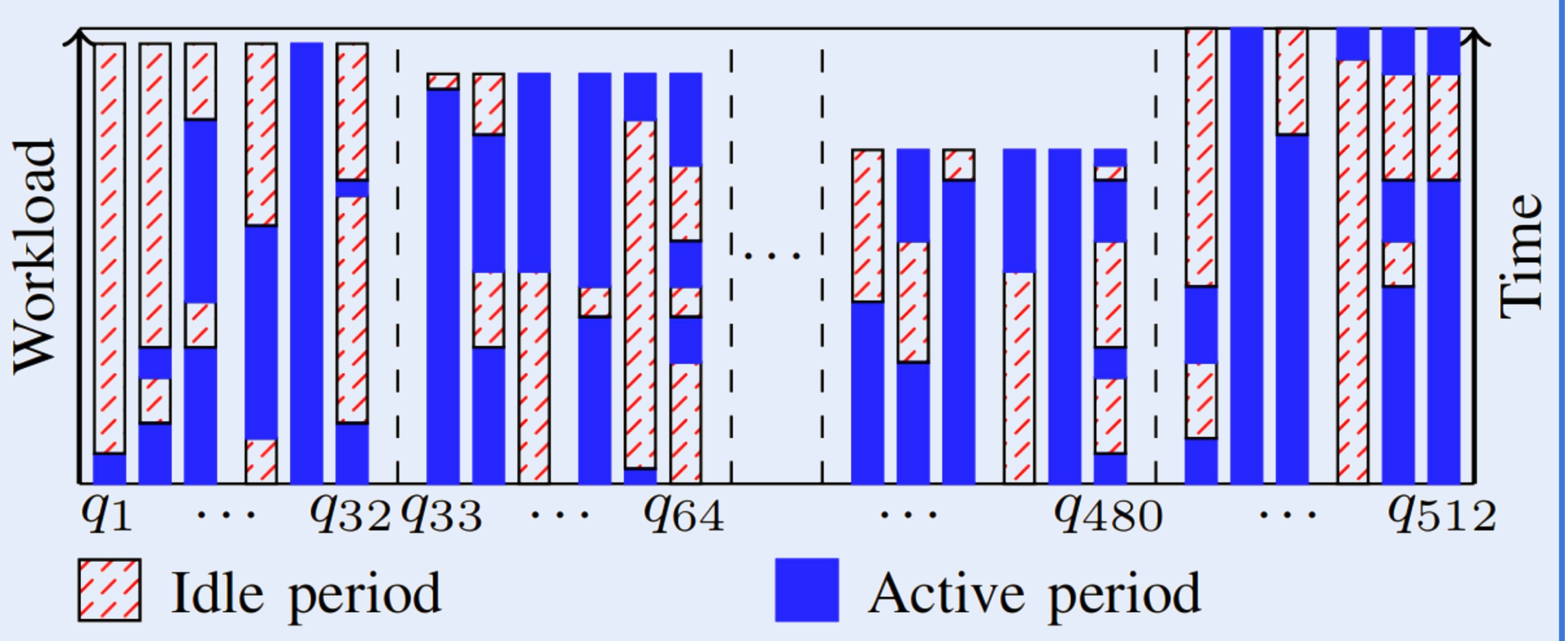

Workload
Time
q_1 ⋯ q_{32} q_{33} ⋯ q_{64} ⋯ q_{480} ⋯ q_{512}
▨ Idle period
■ Active period

Figure 1: Representation of the workload imbalance between the threads

Solution

- Sort points by their workload, execute from most to least workload
  - Reduces workload imbalance within a warp (Fig. 2)
- Because of the GPU's hardware scheduler, cannot ensure this execution order
  - Points may not be executed from most to least workload
- Use a blocking work queue to force the scheduling of threads to points
- Threads retrieve the first point not already computed in non-increasing order of work
  - Within a warp, this yields 32 consecutive points with a very similar workload

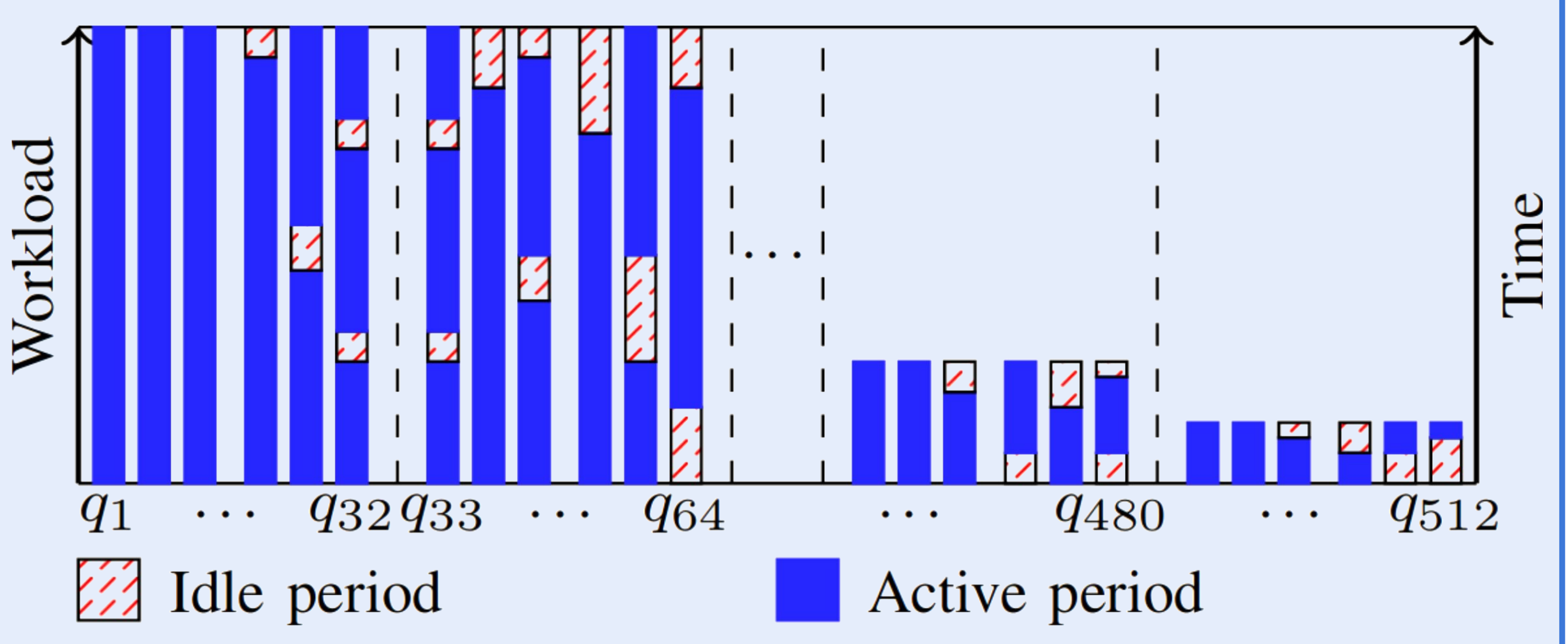

Workload
Time
q_1 ⋯ q_{32} q_{33} ⋯ q_{64} ⋯ q_{480} ⋯ q_{512}
▨ Idle period
■ Active period

Figure 2: Representation of the workload balancing when sorting the points

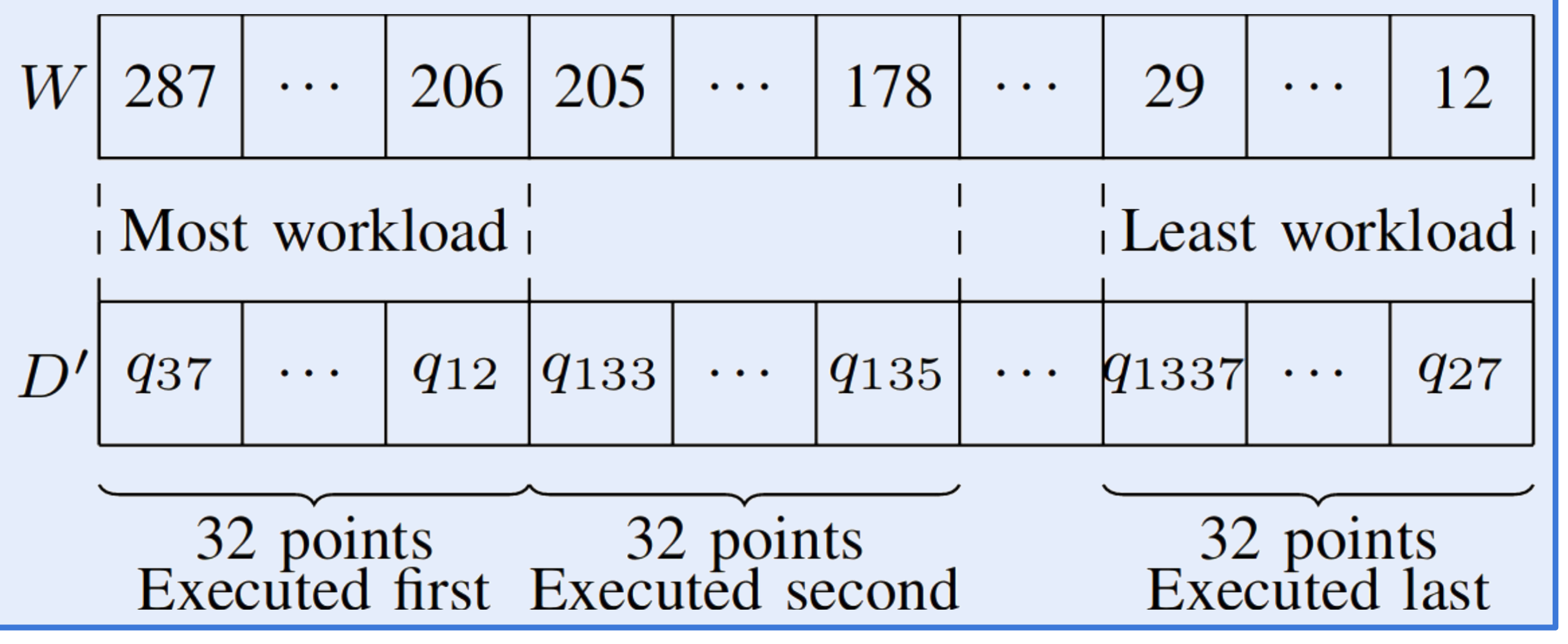
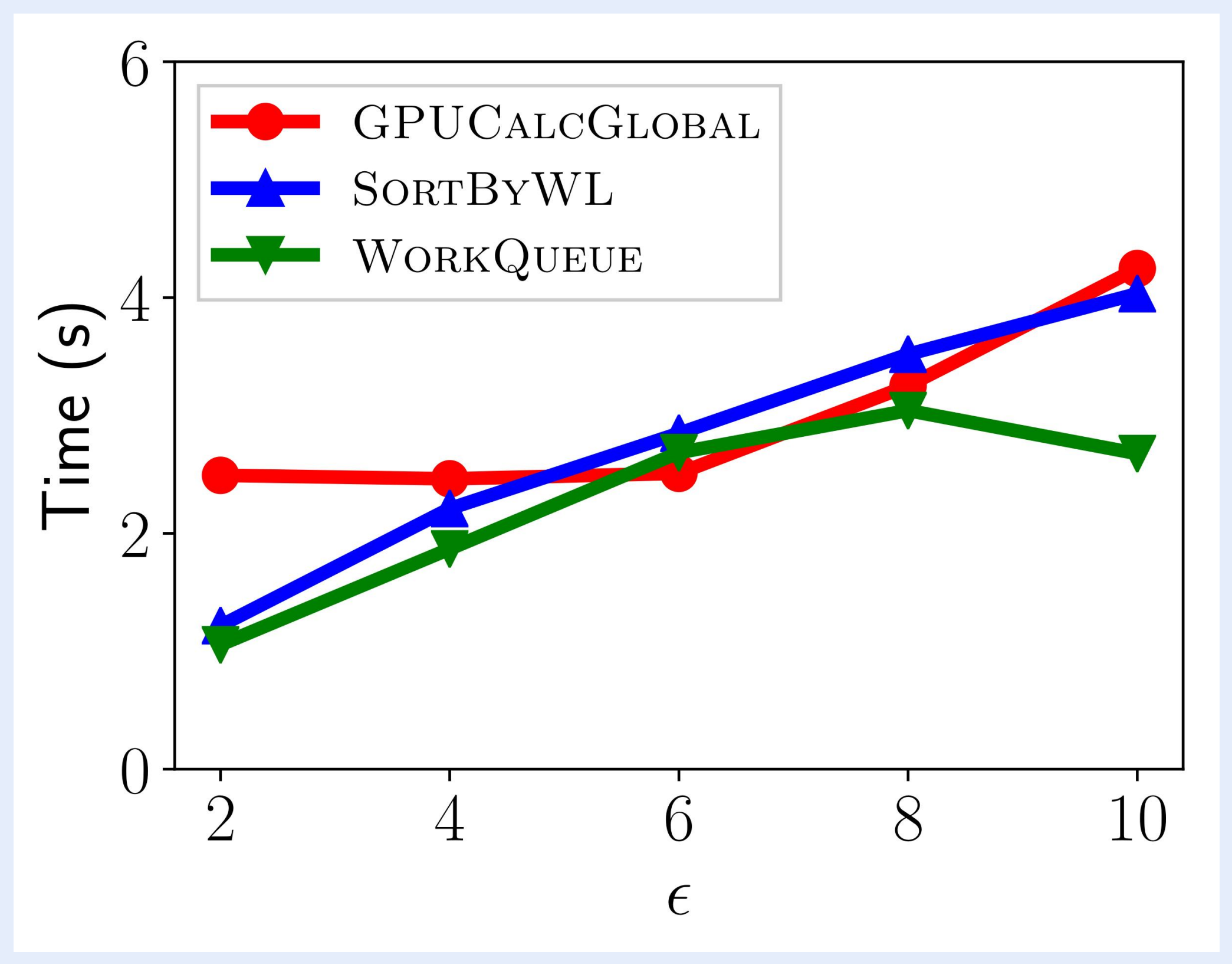

W 287 ⋯ 206 205 ⋯ 178 ⋯ 29 ⋯ 12
Most workload
Least workload
D' q_{37} ⋯ q_{12} q_{133} ⋯ q_{135} ⋯ q_{1337} ⋯ q_{27}
32 points Executed first
32 points Executed second
32 points Executed last

Figure 3: Representation of the functioning of the work queue, the sorted dataset (*D'*) and the workload of the points (*W*)

Results

- Compare GPUCalcGlobal [1], sorting by workload (SortByWL) and our work queue (WorkQueue) [2]
- Focus on the execution time and warp execution efficiency (WEE)
  - Percentage of active threads within a warp: higher is better
- Uniformly distributed datasets have a uniform workload
  - No need to balance the workload between the threads, contrary to exponentially distributed datasets



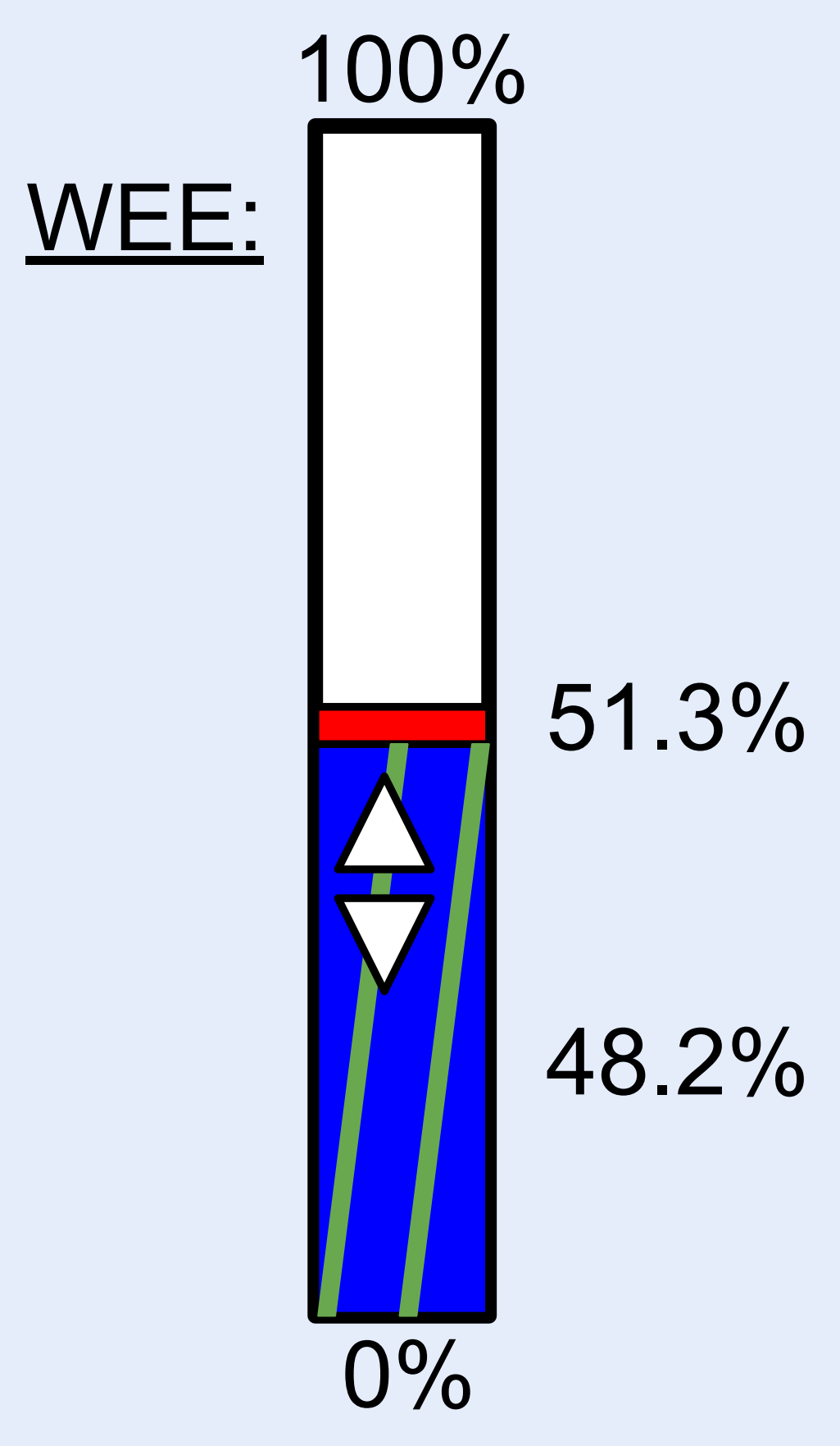
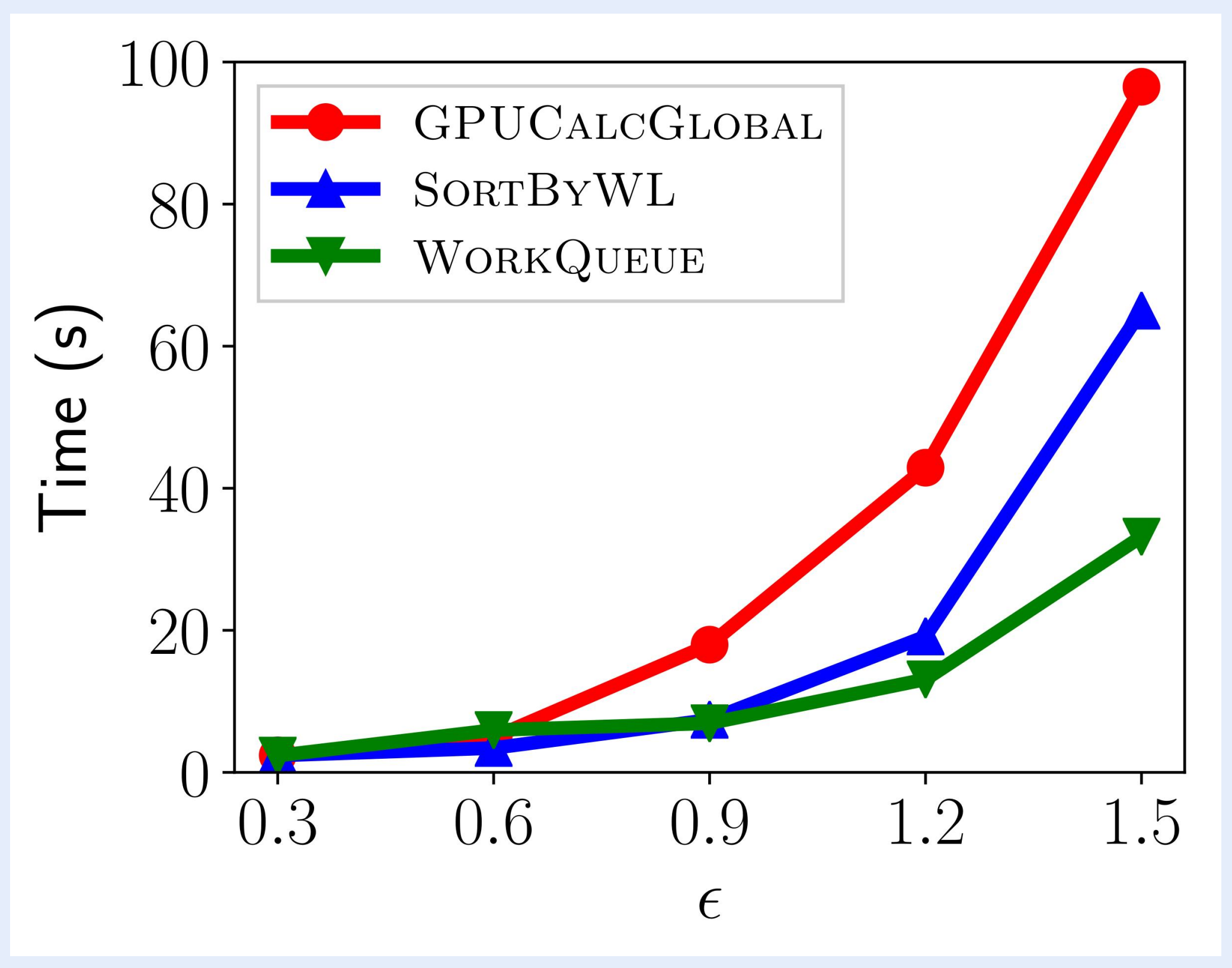


Figure 4: Uniformly distributed dataset, 2M points in 6 dimensions, and WEE for ε = 8



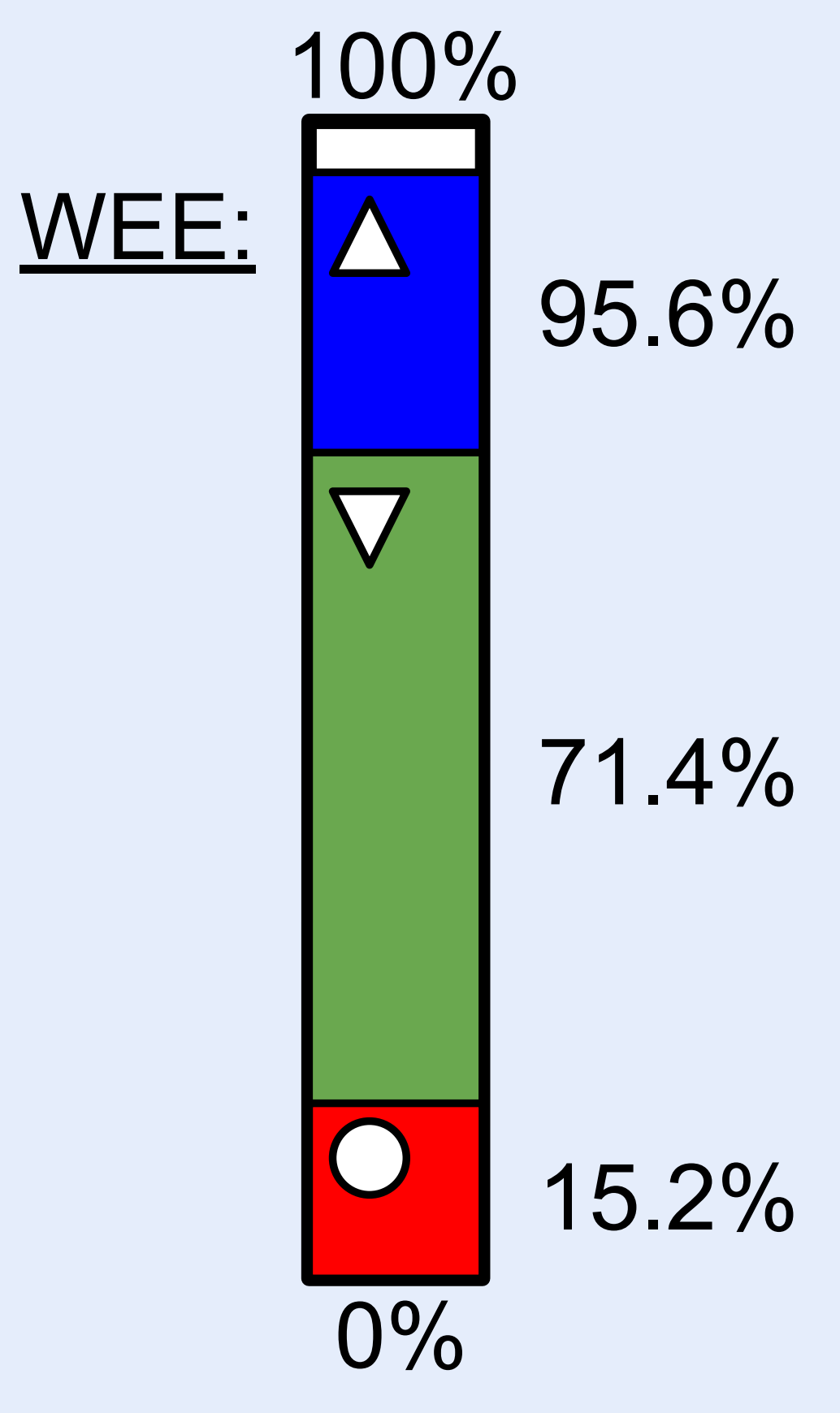


Figure 5: Uniformly distributed dataset, 2M points in 6 dimensions, and WEE for ε = 1.2

Conclusion

- Warp execution efficiency impacts response time
- 100% warp execution efficiency may indicate a computational bound
  - Cannot exceed 100% of active threads per warp
- Use the WorkQueue to improve other data dependent applications

References

[1] M. Gowanlock and B. Karsin, "GPU Accelerated Self-join for the Distance Similarity Metric," Proc. of the 2018 IEEE Intl. Parallel and Distributed Processing Symposium Workshops, pp. 477–486, 2018.
[2] B. Gallet and M. Gowanlock, "Load Imbalance Mitigation Optimizations for GPU-Accelerated Similarity Joins", Proc. of the 2018 IEEE Intl. Parallel and Distributed Processing Symposium Workshops, 2019