# GLOVE: An Interactive Visualization Service Framework with Multi-Dimensional Indexing on the GPU

Jinwoong Kim*, Sehoon Lee†, Joong-Youn Lee†, Beomseok Nam* and Min Ah Kim†
* *School of Electrical and Computer Engineering*
*Ulsan National Institute of Science and Technology (UNIST), Korea*
*E-mail: {jwkim,bsnam}@unist.ac.kr*
† *Korea Institute of Science and Technology Information (KISTI), Korea*
*E-mail: {sehooi,jylee,petimina}@kisti.re.kr*

***Abstract*—In this poster, we present an interactive visualization service framework - *GLOVE* for massively large scale scientific datasets. GLOVE developed at KISTI is designed to support multiple user workloads and to provide interactive user interfaces. GLOVE improves the scalability of the framework and reduces the query latency by managing datasets in large scale distributed memories and by employing multi-dimensional indexing trees that help navigate through massively large datasets. In this poster, we primarily focus on our design and implementation of GPU-based multi-dimensional indexing for GLOVE. Our massively parallel tree traversal algorithm - MPRS for the GPU avoids irregular memory access patterns and recursive backtracking and significantly improves the warp efficiency of the GPU and reduces query response time. Our preliminary experiments show GPU-based indexing accelerates the query performance by an order of magnitude.**

## 1. Introduction

In the past decades, there have been intense research efforts to develop efficient and scalable visualization tools for large scale scientific datasets, such as ParaView [1], VisIt [2], Ensight [3], just to name a few. Our framework GLOVE is distinguishable from the previous visualization tools in that GLOVE is not a tool but an interactive visualization service framework that supports multiple concurrent queries. For that purpose, GLOVE provides user-friendly customizable interfaces, dynamic data management and in-memory caching for better load balancing and workload distribution. Also GLOVE supports dynamic computing resource allocation, and efficient multi-dimensional indexing with the help of GPU acceleration.

## 2. GLOVE

Figure 1 illustrates three components of GLOVE - GLOVE Integrated Visualization Interface (GIVI), GLOVE Rendering Engine (GLORE), and GLOVE Data Manager (GDM). GIVI interface is responsible for receiving concurrent multiple user queries, forwarding the queries to GLORE, receiving polygons or geometric shapes from GLORE, and rendering the results.

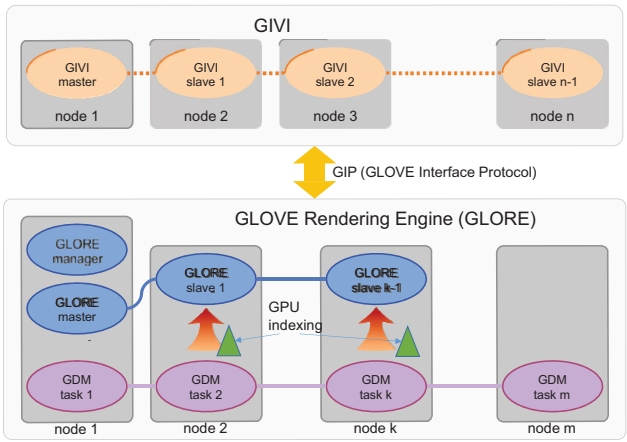


Figure 1. *GLOVE Architecture*

GLORE runs various visualization algorithms in parallel using the Visualization Toolkit (VTK) and MPI, generates polygons, and returns them back to GIVI. GDM provides a parallel IO interface for GLORE and handles multiple IO requests concurrently. GDM employs user-level distributed shared memory using "Global Arrays (GA)" PGAS programming model [4] in order to improve the performance and scalability of the framework by avoiding expensive disk IOs but reading datasets from memories. On top of the Global Arrays, GDM leverages flexible data supply schemes to support multiple user queries and to achieve better load balancing behaviors. GDM provides two kinds of data supply modes - the loop based supply mode and the position based supply mode. In the loop based supply mode, GDM provides data in a round-robin fashion to avoid duplication when each GLORE aggressively requests data as soon as it finishes its job. Because scalar visualization algorithms such as iso-surface or cutting slice require entire data fields, this scheme provides data optimally while preserving load balancing [5]. On the other hand, GDM provides data blocks containing requested particle positions on demand in the position based supply mode. This scheme is designed for the vector visualization algorithms such as streamline or path-

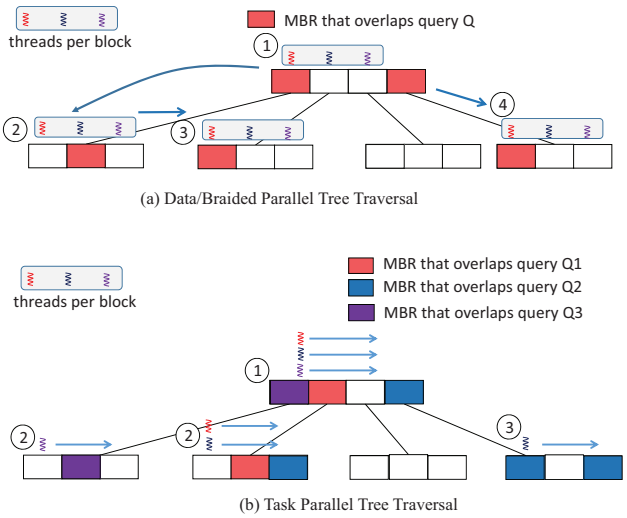


Figure 2. *In data parallel MPRS tree traversal algorithms, a block of CUDA threads cooperate to process a single query while task parallel tree traversal algorithms let each thread access different part of tree nodes and cause warp divergence.*

line computations. We also have adopted data cache based on the path-line optimized replacement algorithm into the GDM to avoid fetching duplicated data while computing streamlines or path-lines [6].

For scientific datasets, GPU is now widely being used for high performance parallel computations. In GLOVE, we implemented multi-dimensional indexing trees and range query functions that run on the GPU so that GLOVE can navigate through large scale scientific datasets efficiently. Multi-dimensional range query is one of the most common data access patterns in scientific applications. However, hierarchical tree structures make it challenging to traverse the trees in a regular and contiguous manner, which is necessary for GPU kernel functions.

## 3. GPU Indexing

Inherently, traversing hierarchical multi-dimensional indexes such as R-trees requires recursive back-tracking and irregular memory accesses. Thus tree structured indexes are not well suited for the GPU, and a large number of previous works employ brute-force scanning for the GPU indexing [7], [8]. More importantly the recursive tree traversal algorithms often crash due to tiny runtime stack size of modern GPUs.

In order to address the challenges of GPU indexing, we designed and implemented a stackless tree traversal algorithm - *Massively Parallel Restart Scanning (MPRS)* that traverses hierarchical tree structures without recursion and scans contiguous leaf nodes [9]. In order to maximize the core utilization of the GPU, we let the degree of indexing tree nodes to be a multiple of the number of cores in a GPU streaming multiprocessor, so that all the bounding boxes of

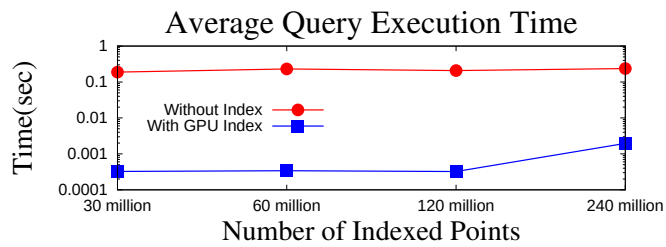


Figure 3. *Range Query Respone Time (DeltaWing Datasets)*

a single tree node can be compared against a given query in parallel while avoiding warp-divergence.

Figure 2 illustrates how data parallel MPRS algorithm and task parallel traversal algorithms differ in accessing tree nodes. In data parallel MPRS tree traversal algorithms, a set of threads in a block concurrently access the same tree node but threads in task parallel tree traversal algorithms independently access different parts of tree structure.

In MPRS algorithm, a block of GPU threads fetches a single tree node at a time, chooses the leftmost overlapping child node until it reaches a leaf node, and scans right sibling leaf nodes until we visit a leaf node that does not contain any overlapping data point. Once it finds a non-overlapping leaf node, MPRS algorithm restarts the tree traversal from root node, but this time, it avoids visiting the already visited nodes or their ancestor nodes by keeping track of the largest index of visited leaf nodes. For more details of MPRS tree traversal algorithm, please refer to [9].

## 4. Evaluations

We integrated R-tree and MPRS search algorithm for the GPU into GLOVE data manager so that GLOVE can find and read the user-requested data blocks with low latency. In the experiments shown in Figure 3, we evaluate the performance improvement in terms of range query response time with and without the multi-dimensional indexing on the GPU. Without the index, GDM compares the spatio temporal coordinates of data points in a brute force way, thus it takes longer than 200 msec. With the GPU indexing, the range query processing does not take longer than 2 msec with 240 millions of point datasets, which is 100 times faster than GDM.

We also measured the query processing throughput of our MPRS algorithm on the GPU since GLOVE is designed for multiple concurrent clients. Needless to say, brute-force search is not even worth to mention, and the MPRS algorithm on the GPU shows 364 times higher query processing throughput against multi-threaded R-trees on CPUs.

## References

[1] "ParaView," *http://www.paraview.org/*.

[2] "VisIt," *http://visit.llnl.gov/*.

[3] "Ensight," *http://www.ensight.com/*.

[4] "Global arrays programming models," *http://hpc.pnl.gov/globalarrays/*.

[5] M. A. Kim, Y. J. Hur, and J.-Y. Lee, "Invis : An interactive visualization framework for massive data supporting multiple users," *Journal of KIISE : Computing Practices and Letters*, vol. 18, no. 1, pp. 1–11, 2011.

[6] J.-Y. Lee and J. Park, "Cache enhancement methods for out-of-core pathline computation," in *Proceedings of the IEEE VIS Conference*, 2013.

[7] J. Chou, K. Wu, O. Rubel, M. Howison, J. Q. Prabhat, B. Austin, E. W. Bethel, R. D. Ryne, and A. Shoshani, "Parallel index and query for large scale data analysis," in *Proceedings of the ACM/IEEE SC2011 Conference*, 2011.

[8] Y. Su, G. Agrawal, and J. Woodring, "Indexing and parallel query processing support for visualizing climate datasets," in *Proceedings of 41st International Conference on Parallel Processing*, 2012, pp. 249–258.

[9] J. Kim, W.-K. Jeong, and B. Nam, "Exploiting massive parallelism for indexing multi-dimensional datasets on the gpu," *Transactions on Parallel and Distributed Systems*, vol. 26, no. 8, pp. 2258–2271, 2015.