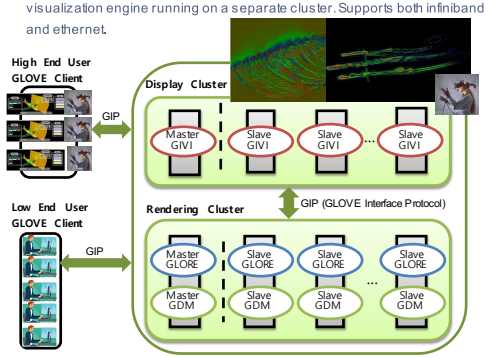
GLOVE: An Interactive Visualization Service Framework with Multi-Dimensional Indexing on the GPU

Jinwoong Kim[1], Sehoon Lee[2], Joong-Youn Lee[2], Beomseok Nam[1], Min Ah Kim[2]

[1] Ulsan National Institute of Science and Technology, [2] Korea Institute of Science and Technology Information

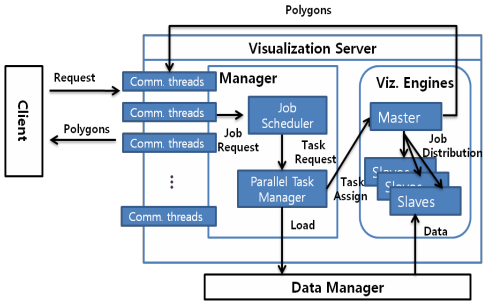
GLOVE

- GLOVE developed at KISTI, is a novel interactive visualization service framework.
- GLOVE consists of several components, most of which are implemented as parallel programs to run on large-scale clusters.
  - Data manager (GDM) : reads simulation dataset into memory and feed the dataset to the visualization engine.
  - Visualization engine (GLORE) : receives command from clients and extracts geometries from dataset. Also it is responsible for sending the geometry back to the visualization client.
  - Communication library (GIP) : lets visualization clients communicate with visualization engine running on a separate cluster. Supports both infiniband and ethernet.



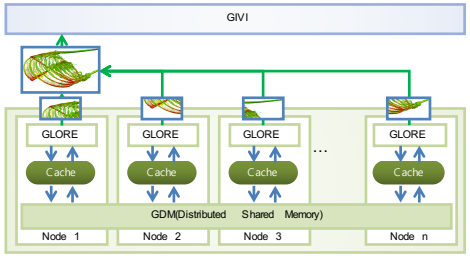
GLORE - GLOVE Rendering Engine

- GLORE is responsible for running various visualization algorithms in parallel using Visualization Toolkit (VTK) and MPI.
  - GLORE consists of GLORE Manager and GLORE Workers (Visualization Engines).
  - GLORE Manager schedules incoming queries and assign them to a set of workers, which run various visualization algorithms such as iso-surface, streamline, cutting plane, etc.
  - In order to generate polygons for clients, GLORE reads raw input datasets through GLOVE Data Manager (GDM).



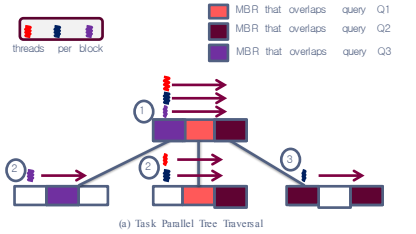
GDM - GLOVE Data Manger

- GLOVE Data Manager (GDM) serves GLORE's IO requests, i.e., reads input data points and caches them in networked shared memories.
- GDM implements the distributed shared memory using "Global Arrays", PGAS programming model.
- 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 employs two data supply modes:
  - In 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.
  - In position based supply mode, GDM provides data blocks containing requested particle positions on demand.
- Data cache based on the path-line optimized replacement algorithm is adopted between GDM and GLORE to avoid fetching duplicated data while computing streamlines or path-lines.
- In order to find and fetch requested data points, multi-dimensional indexing is needed in GLOVE.

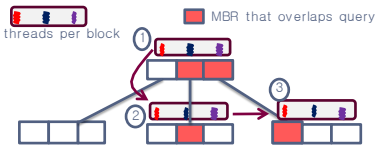
GPU Indexing - Task Parallel Tree Traversal



(a) Task Parallel Tree Traversal

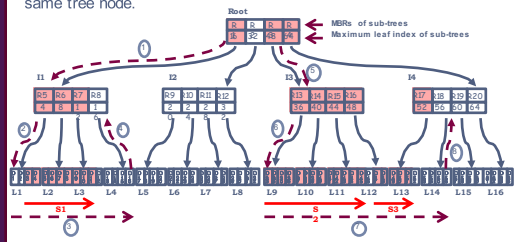
- Irregular tree structures and the hierarchical but irregular tree traversal patterns make it difficult to achieve good load balance and to maximize the utilization of parallel computing resources.
- With the task parallelism, there can be varying number of tree node accesses per thread depending on the query range, thus the query execution times of queries are not homogeneous and the query turn-around times are determined by the slowest query.

GPU Indexing - Data Parallel Tree Traversal



(b) Data/Braided Parallel Tree Traversal

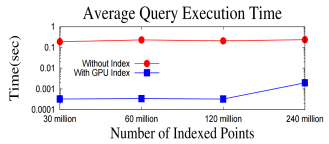
- In our data parallel tree traversal algorithm – MPRS (Massively Parallel Restart Scanning), a set of threads in a block concurrently access the same tree node.



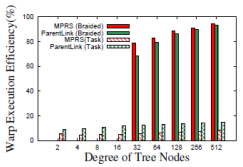
- In MPRS algorithm, a block of GPU threads fetches a single tree node at a time, chooses the leftmost overlapping child node until it reaches a leaf node.
- Once it reaches a leaf node, it scans right sibling leaf nodes until it visits a leaf node that does not contain any overlapping data point. Then, 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.

GPU Indexing on GLOVE & Performance Improvement

Experimental Environments
DeltaWing dataset
CentOS 6.3
64GB Memory
Xeon X5450 3.0GHz
NVIDIA Quadro 6000



- GPU Indexing helps GDM find requested data points **100x** faster.
- Due to high SIMD efficiency, MPRS yields fast query response time as well as high query processing throughput, which is necessary for interactive service frameworks.



Performance Evaluations

Experimental Environments	
Number of Nodes	64
CPU core per node	4 (Xeon X5450 3.0GHz)
Total CPU cores	256
Memory size per node	32GB
Total memory size	8TB

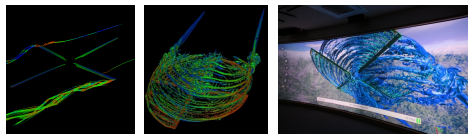
Rotor Dynamics Dataset	
Number of Variables	10
Time-steps	90
Precision	Double
Grid Type	Multi-Block Structured Grid
Size	1.11TB

- Performance comparison

Tool	GLOVE(secs)	ParaView(secs)	VisIt(secs)
Streamline	7.89	241.92	13.74
Path-line	261.36	N/A	N/A
※Iso-surface animation	3.37	31.43	7.21

※ GLOVE loads the entire dataset in advance while others load data in run-time.
※ For Iso-surface animation, we measure an average time to create a single time-step iso-surface during generating a series of iso-surfaces on entire time-steps (90 for these experiments).

- Visualization results



(a) streamline (b) iso-surface (c) iso-surface on VR environment

Discussion & Conclusion

- Although GLOVE makes use of application-level caches to reduce number of fetches from external nodes, creating streamlines is still too slow due to the computing time to find requested data points.
- GPU Indexing is a promising technique to accelerate a process to find data cells where requested particles are located in.
- In this poster, we focus on design and implementation of GPU-based multi-dimensional indexing for GLOVE.
- Experiment results show our GPU-based indexing accelerates the query performance by an order of magnitude.

References

- 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.
- J.-Y. Lee and J. Park, "Cache Enhancement Methods for Out-Of-Core Pathline Computation", in Proceedings of the IEEE VIS 2013 Conference, 2013.
- J. Kim, W.-K. Jeong, and B. Nam, "Exploiting massive parallelism for indexing multi-dimensional datasets on the gpu," IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 8, pp. 2258–2271, 2015.