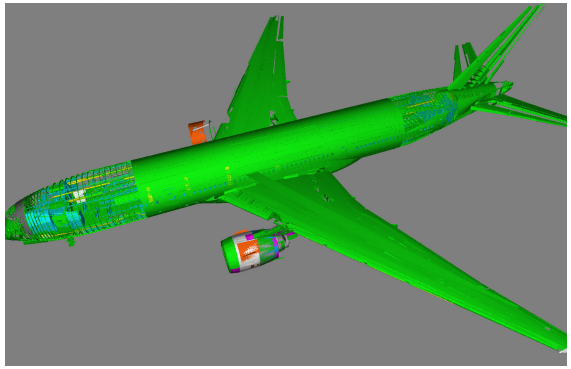
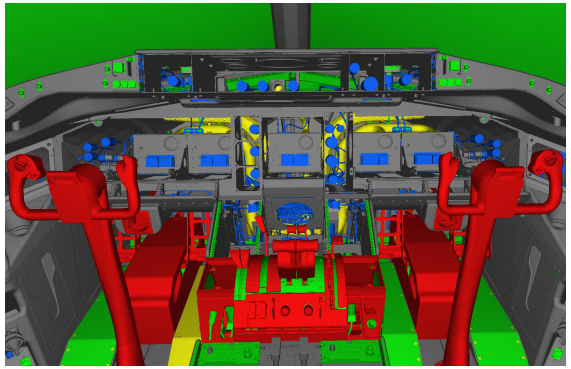
VirginiaTech
Invent the Future

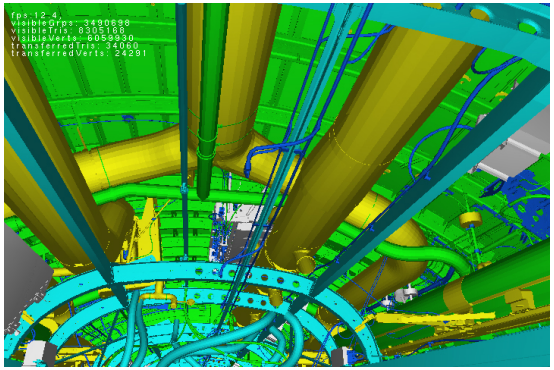
Load Balanced Parallel GPU Out-of-Core for Continuous LOD Model Visualization

Chao Peng, Peng Mi and Yong Cao

Department of Computer Science,
Virginia Tech, Blacksburg, Virginia, USA

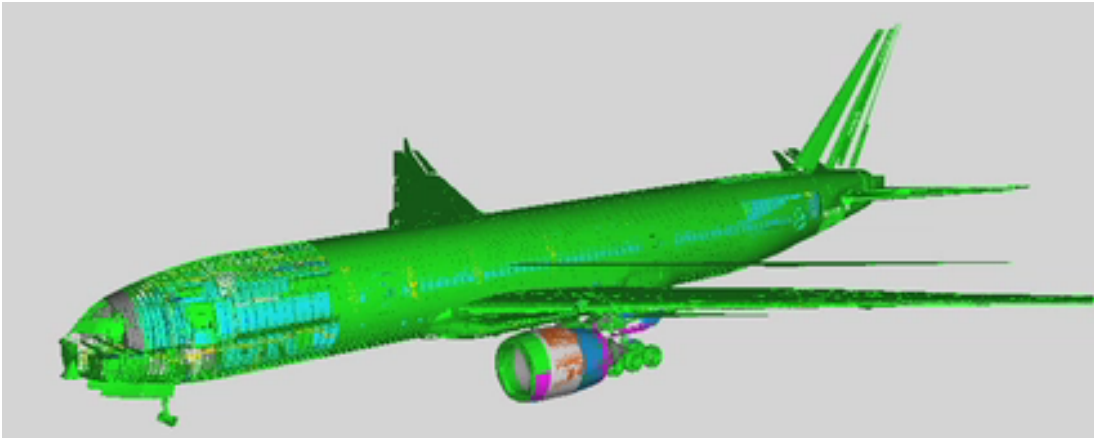






Motivation

- How to efficiently render a large 3D model that contains a lot of objects and triangles?



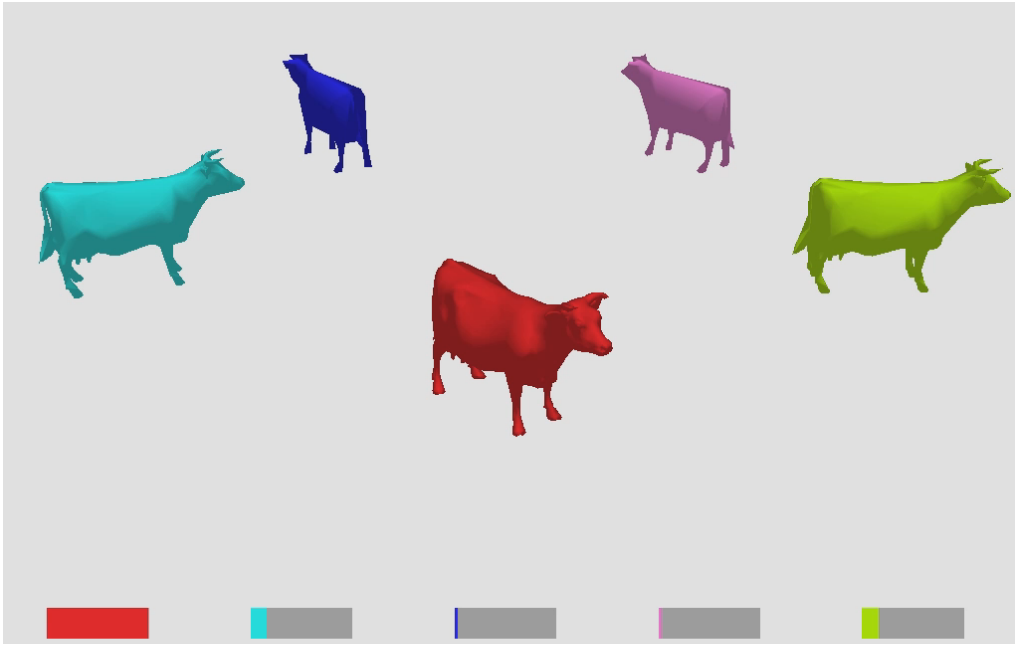
The Boeing 777 model:

Triangles: 332 million
Vertices: 223 million
Objects: 719 thousand

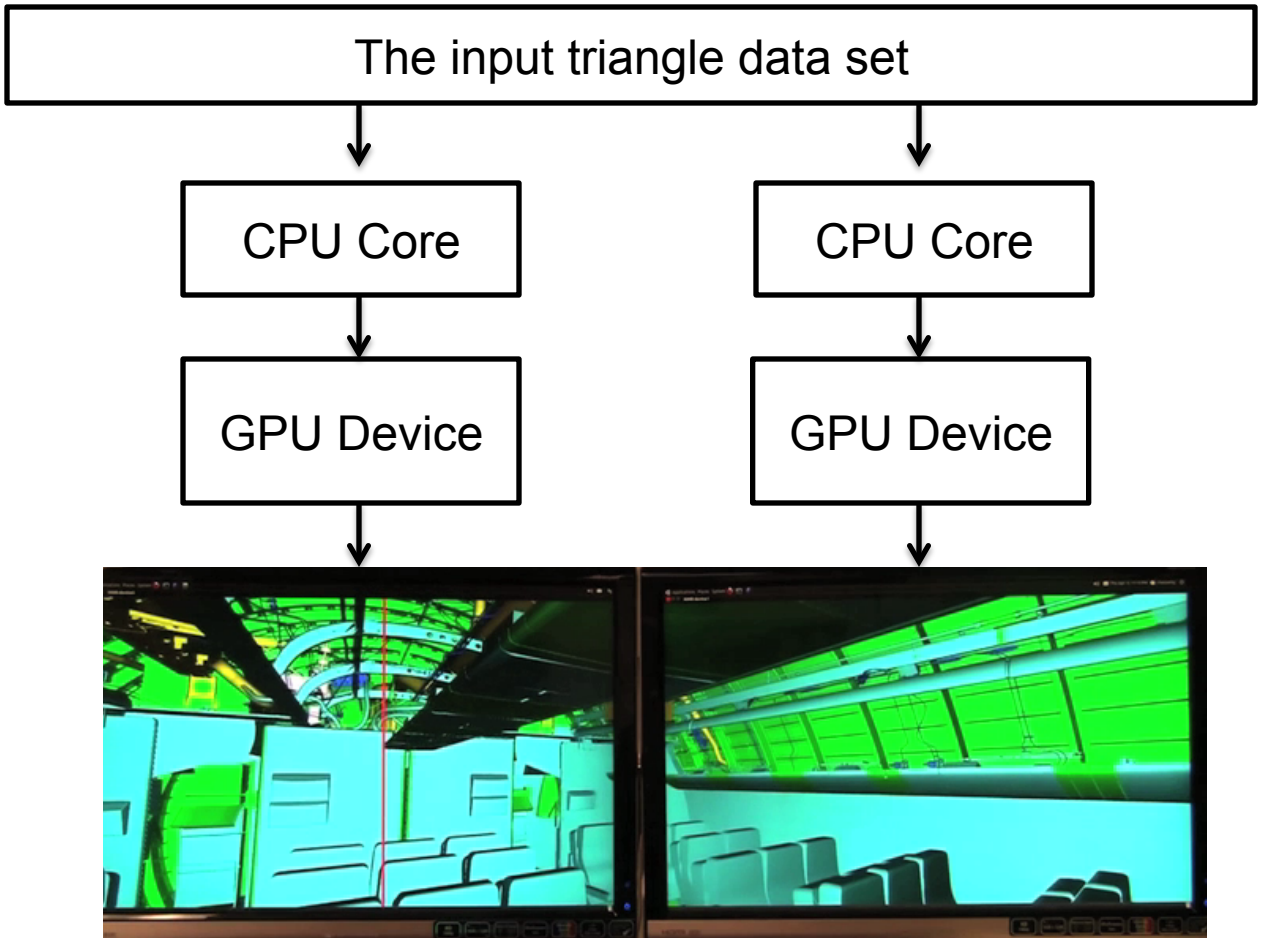
Rendering difficulties:
The objects have dramatically different shapes and are topologically disconnected.
The data size is far beyond the GPU rendering capabilities.

The Previous Approach

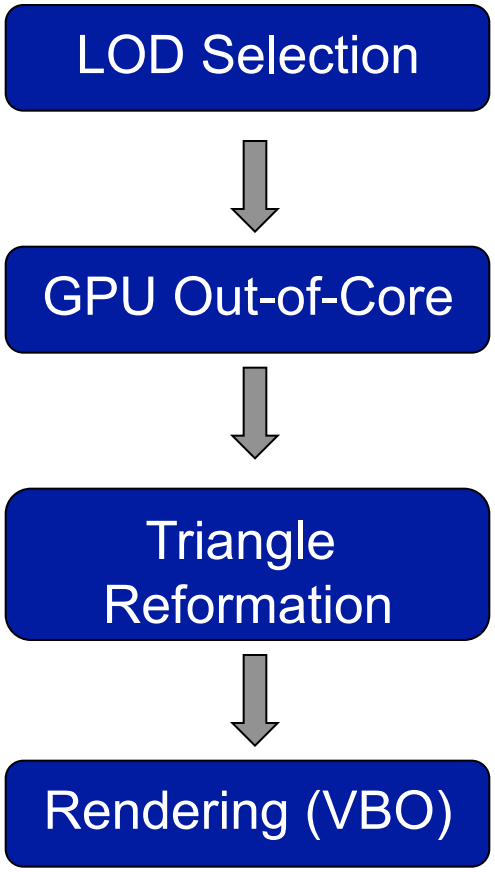
- Our GPU-based approach in EuroGraphics'12.
  - Parallel Continuous LOD: triangle-level mesh simplification.
  - GPU Out-of Core: CPU-GPU data streaming.

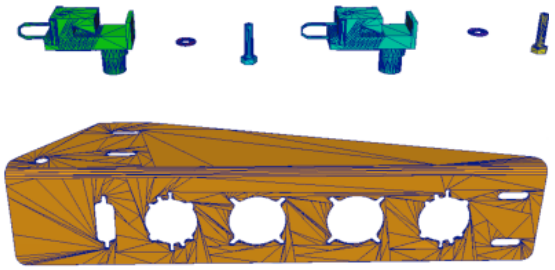


A Multi-GPU and Multi-Display System



The approach on a single GPU



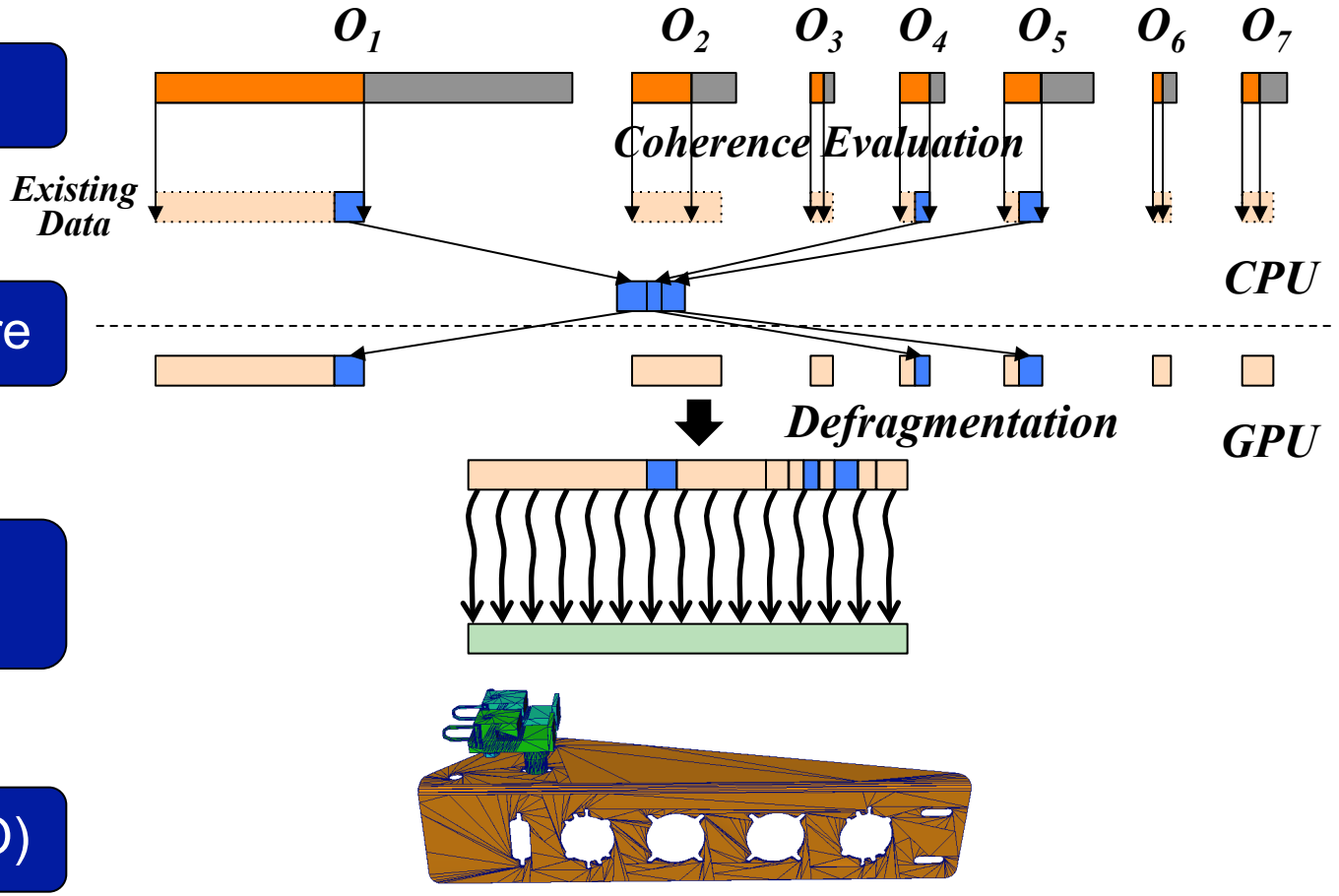


LOD Selection

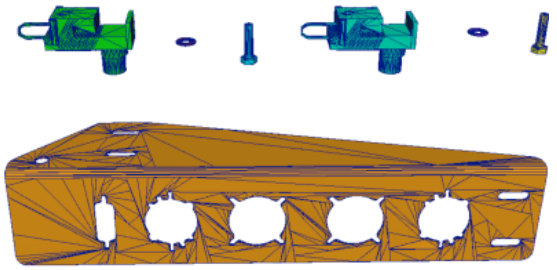
GPU Out-of-Core

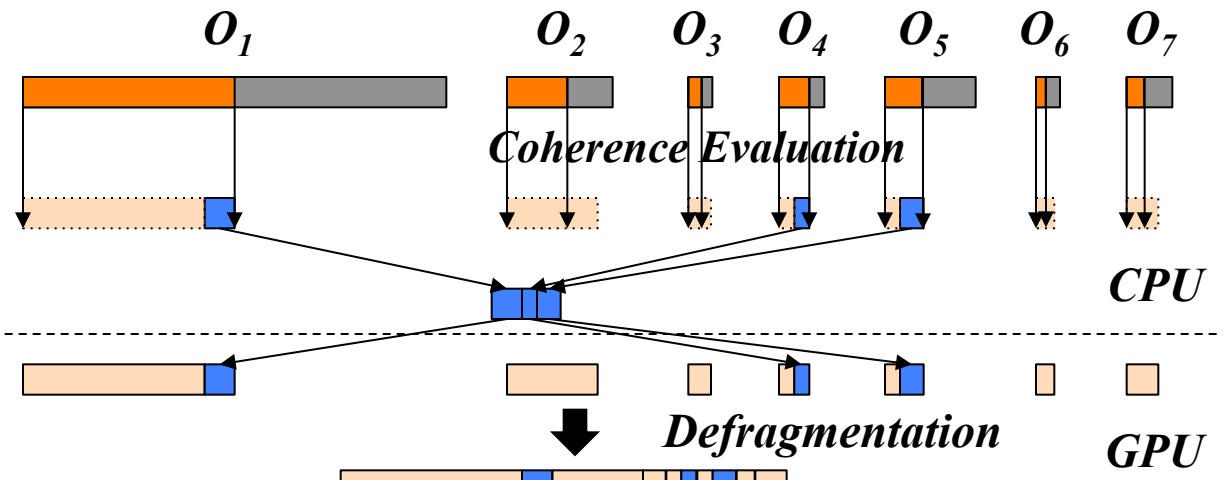
Triangle Reformation

Rendering (VBO)



Performance Bottleneck

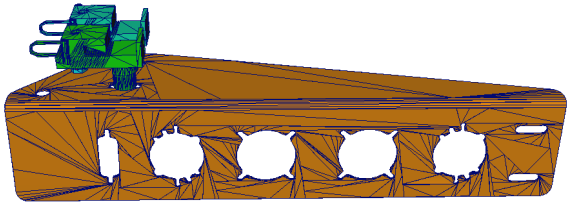




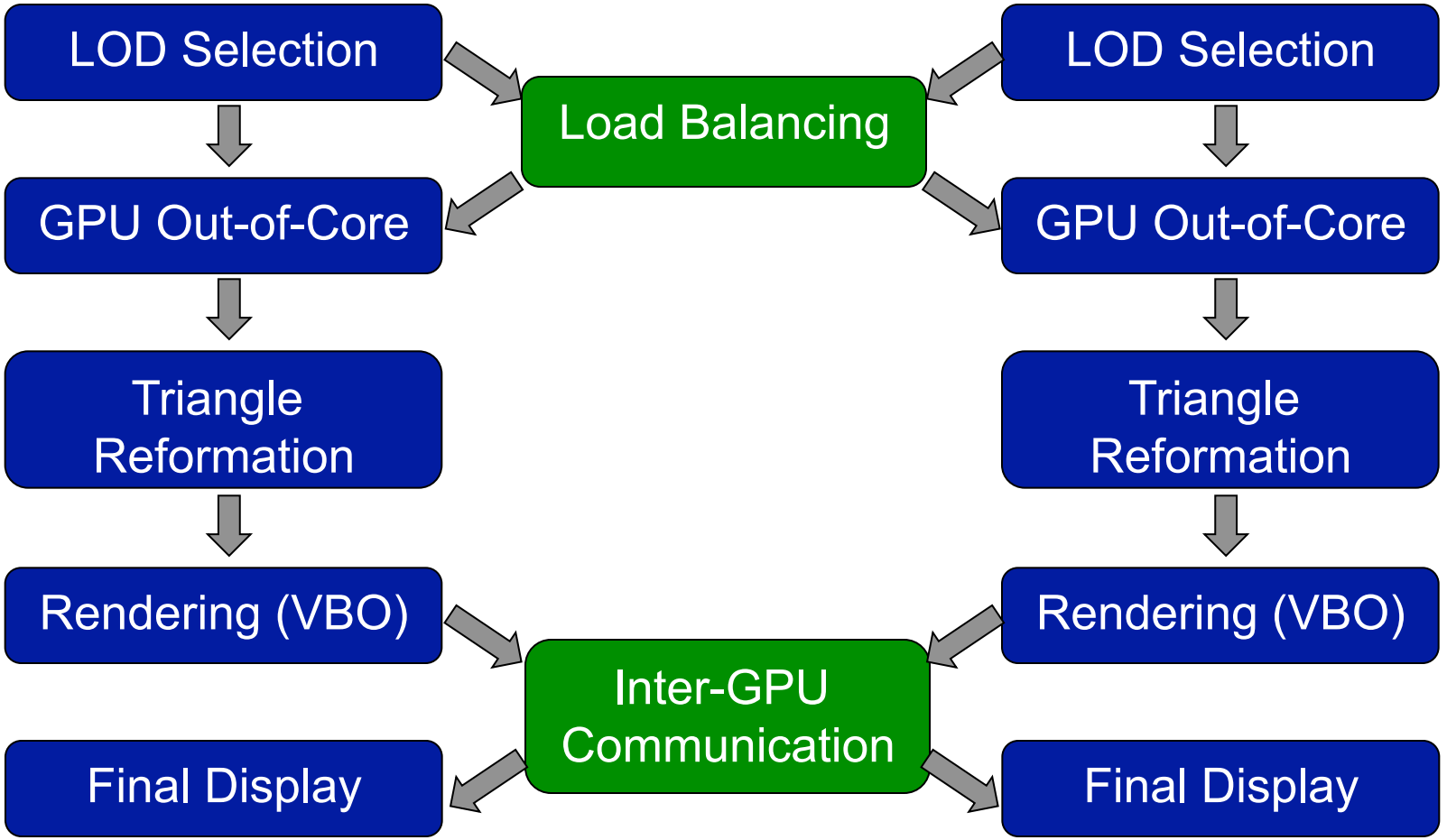
GPU Out-Of-Core
45%

Triangle Reformation
20%

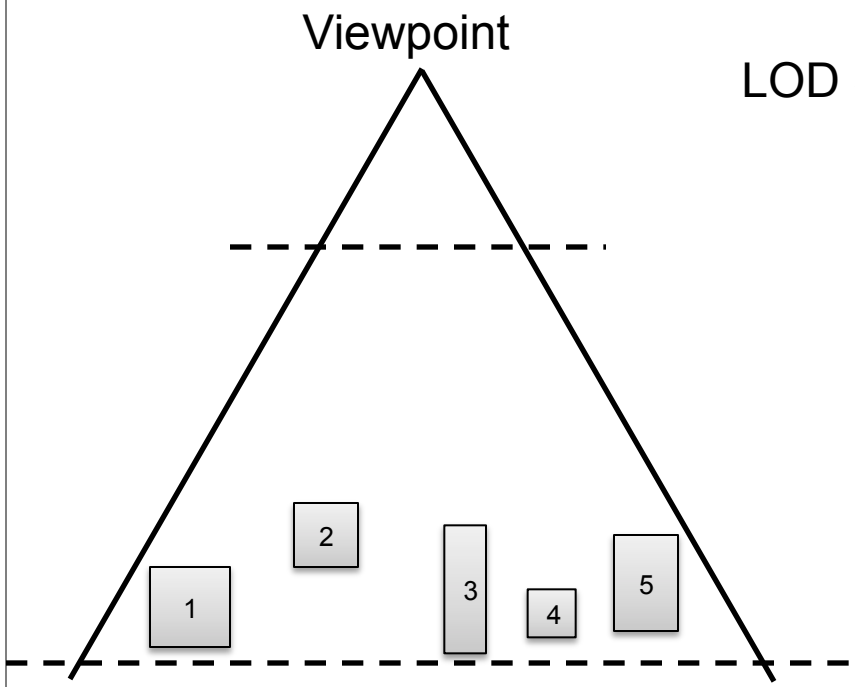
OpenGL VBO Rendering
28%



Contributions



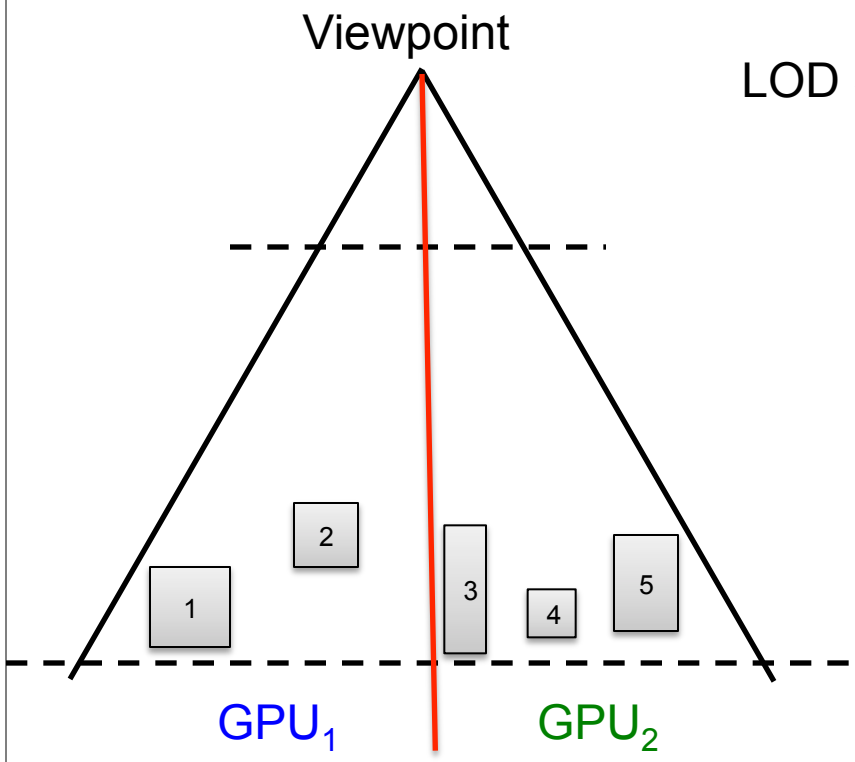
Load Balancing



LOD Selection Result:

1	2	3	4	5
n_1	n_2	n_3	n_4	n_5

Load Balancing



LOD Selection Result:

1	2	3	4	5
n_1	n_2	n_3	n_4	n_5

GPU1:

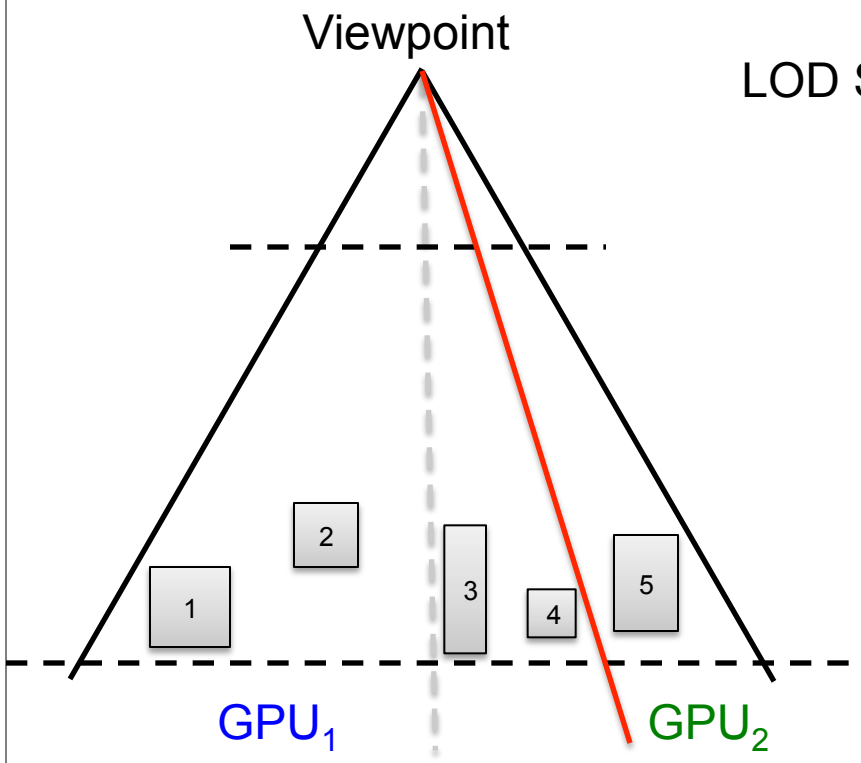
n_1	n_2	0	0	0

GPU2:

0	0	n_3	n_4	n_5

$$\frac{n_1+n_2}{n_3+n_4+n_5} \notin [1-t, 1+t]$$

Load Balancing



LOD Selection Result:

1	2	3	4	5
n_1	n_2	n_3	n_4	n_5

GPU1:

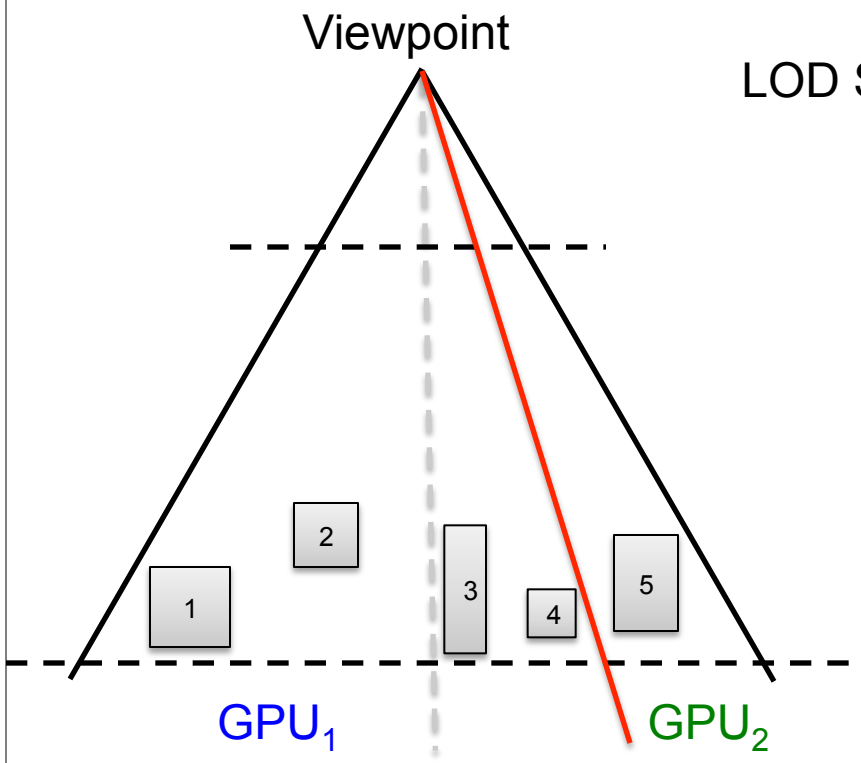
n_1	n_2	0	0	0

GPU2:

0	0	n_3	n_4	n_5

$$\frac{n_1+n_2}{n_3+n_4+n_5} \notin [1-t, 1+t]$$

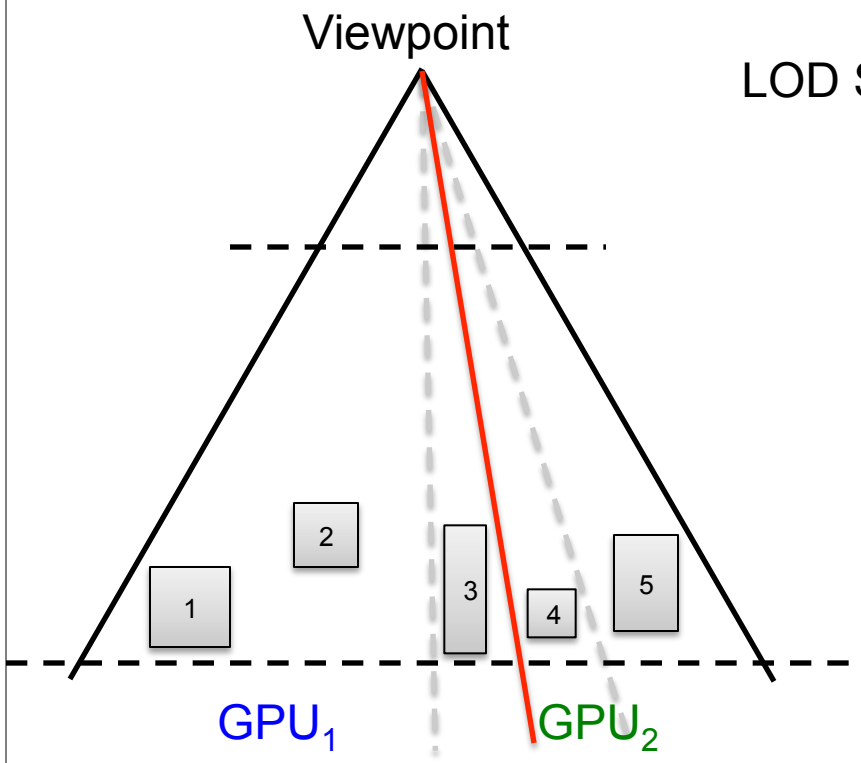
Load Balancing



	1	2	3	4	5
LOD Selection Result:	n_1	n_2	n_3	n_4	n_5
GPU1:	n_1	n_2	n_3	n_4	0
GPU2:	0	0	0	0	n_5

$$\frac{n_1+n_2+n_3+n_4}{n_5} \notin [1-t, 1+t]$$

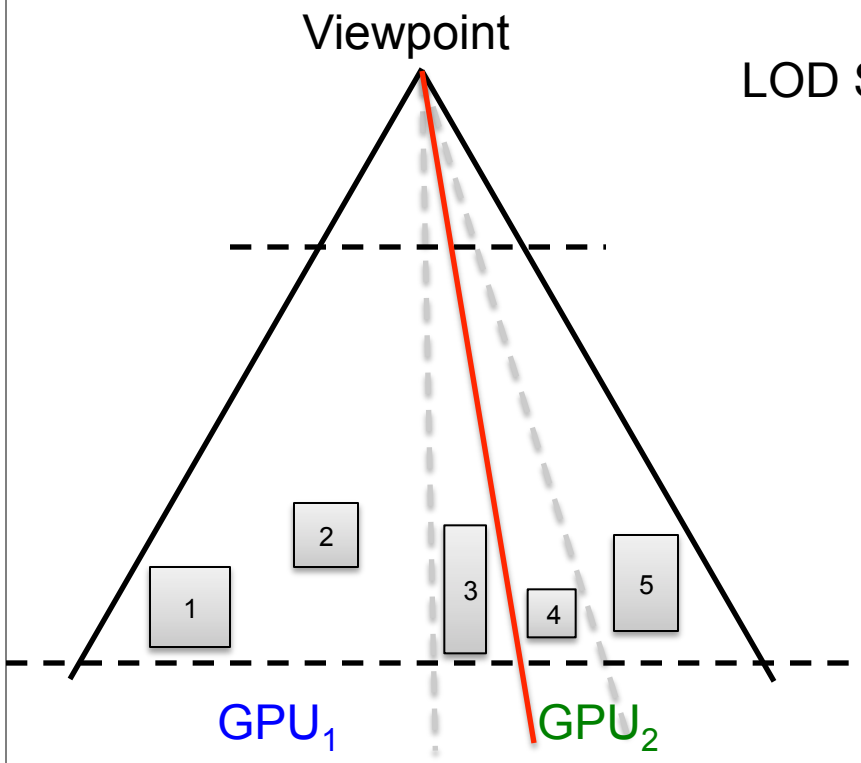
Load Balancing



	1	2	3	4	5
LOD Selection Result:	n_1	n_2	n_3	n_4	n_5
GPU1:	n_1	n_2	n_3	n_4	0
GPU2:	0	0	0	0	n_5

$$\frac{n_1+n_2+n_3+n_4}{n_5} \notin [1-t, 1+t]$$

Load Balancing



LOD Selection Result:

1	2	3	4	5
n_1	n_2	n_3	n_4	n_5

GPU1:

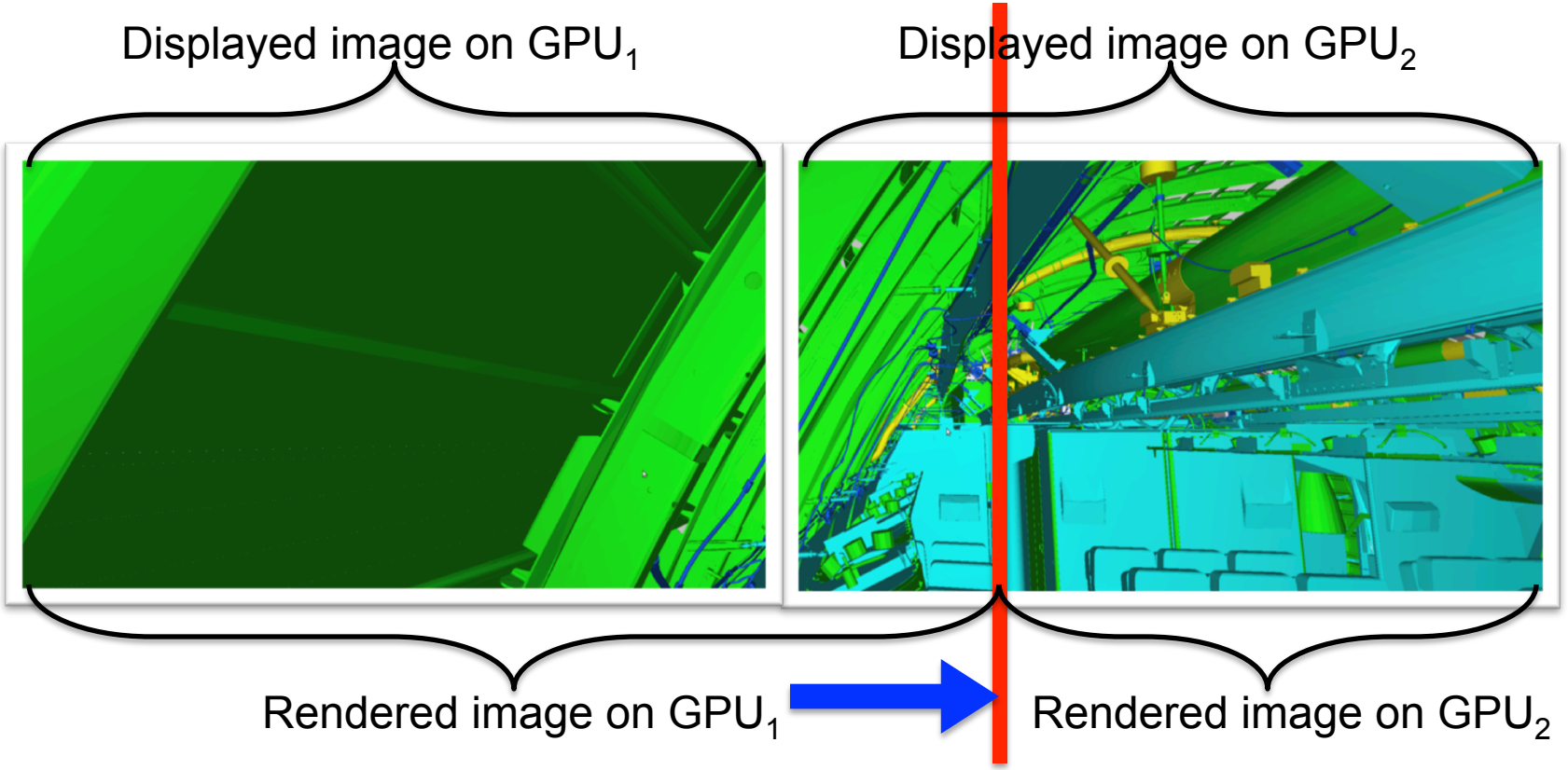
n_1	n_2	n_3	0	0

GPU2:

0	0	0	n_4	n_5

$$\frac{n_1+n_2+n_3}{n_4+n_5} \in [1-t, 1+t]$$

Inter-GPU Communication



CUDA Inter-Process Communication (CUDA 4.1 IPC) for transferring image buffer.

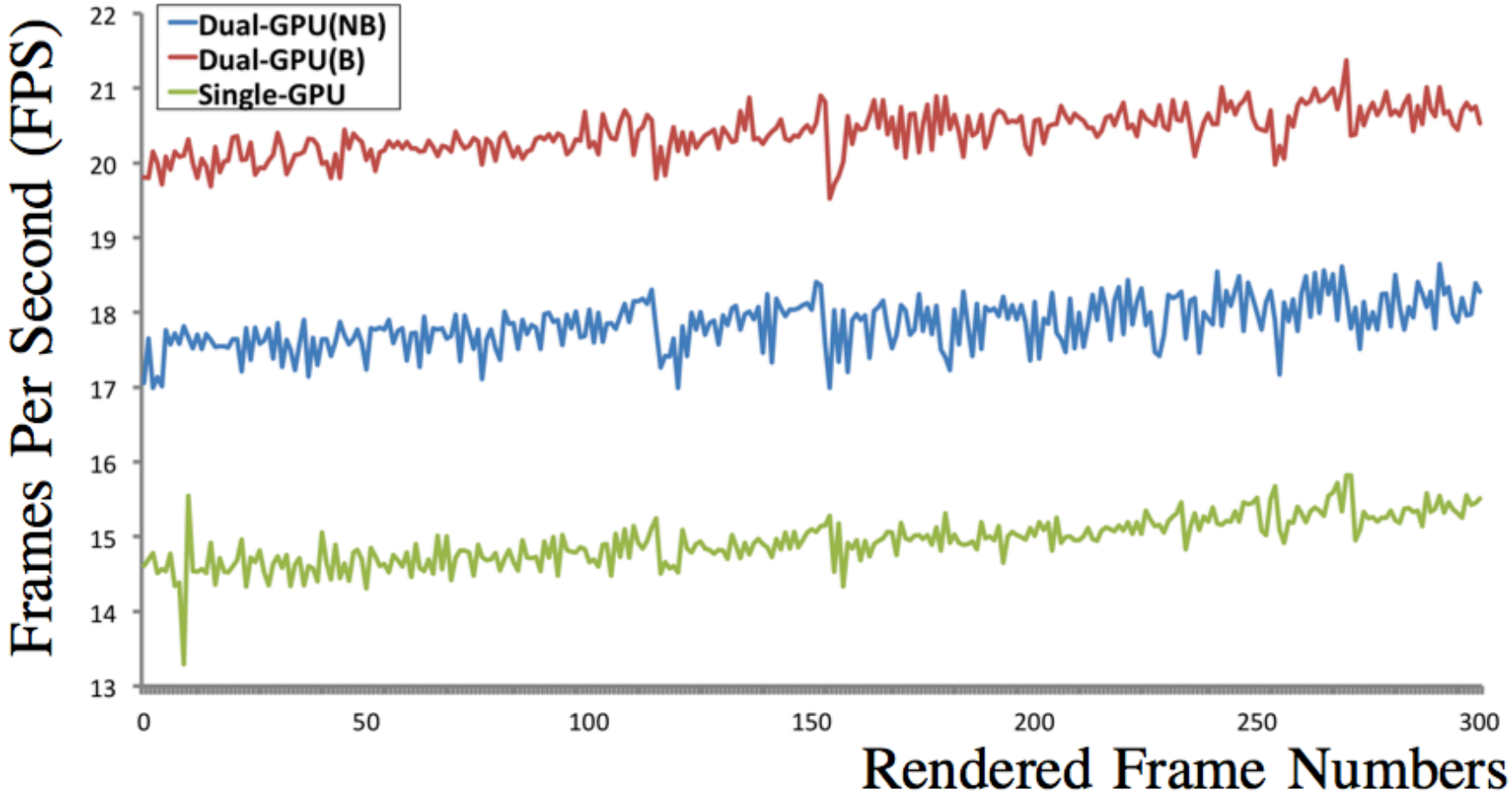
Implementation

- Two GPUs:
  - NVIDIA GTX 580.
  - 512 cores, 3 GB DDR5.
  - 192.4 GB/s memory bandwidth.
- CPU Main Memory:
  - 16 GB RAMs.
- Rendering performance:
  - An average of 20 fps on the Linux system with MPI and CUDA 4.2.

Performance Evaluation

- Comparison
  - Dual-GPU with load balancing (our approach).
  - Dural-GPU without load balancing.
  - Single GPU.

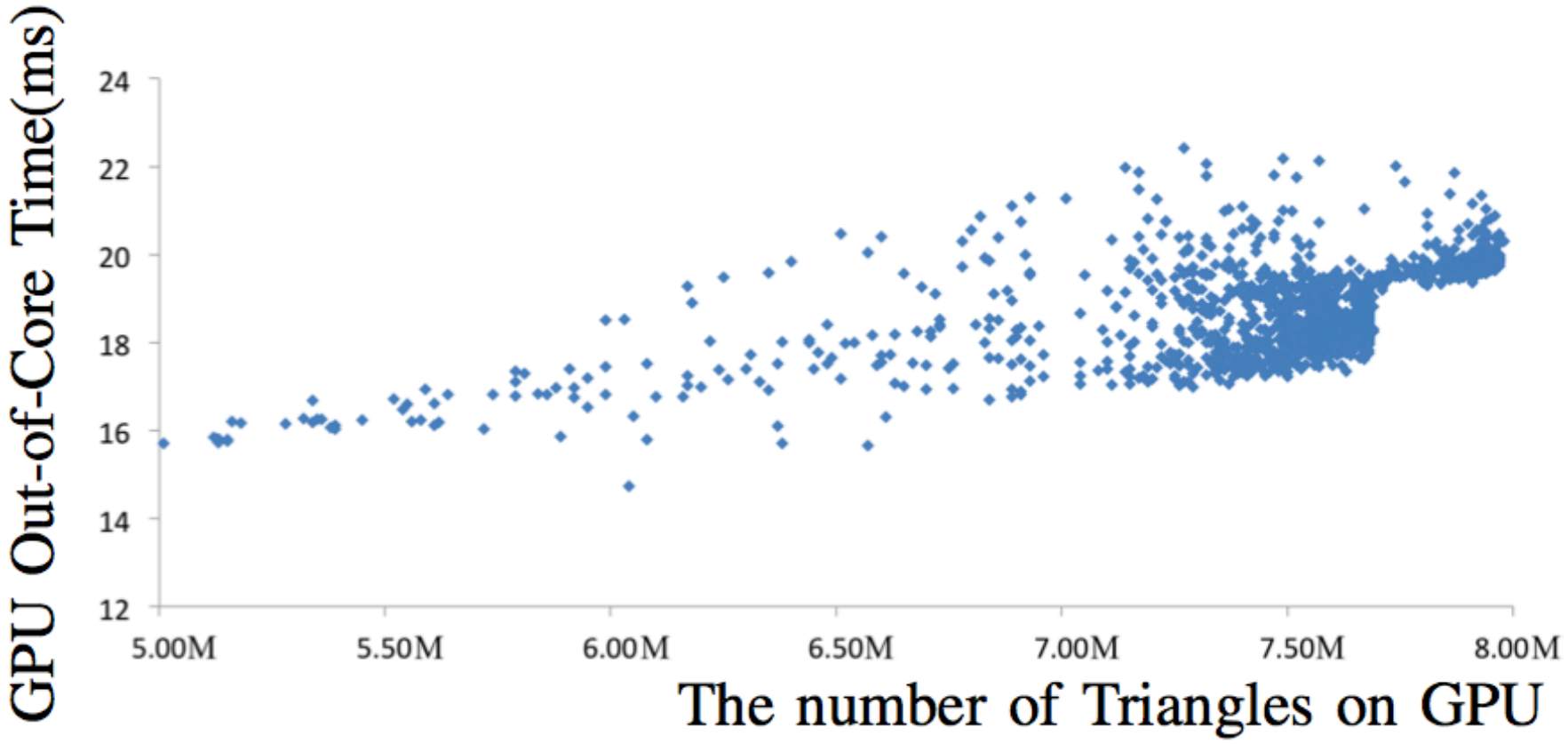
Performance Evaluation



Performance Evaluation

Approach	FPS	Diff. Triangle Num.	Visible Triangle Num.	Load Balancing	GPU Out-Of-Core	Triangle Reformation	GL Rendering
Single-GPU	14.94	---	12.29 M	---	29.62 ms	3.62 ms	30.24 ms
Dual-GPU (NB)	17.84	7.94 M	12.29 M	---	24.54 ms	2.85 ms	25.31 ms
Dual-GPU (B)	20.40	0.37 M	12.29 M	5.38 ms	18.56 ms	1.97 ms	19.13 ms

Performance Evaluation



Conclusion

- A rendering system with two GPUs:
  - The workload balancer based on view-frustum partitioning method.
- Inter-GPU communication for image re-arrangement.
- Future work:
  - Scalability beyond two GPUs.

Acknowledgment

Thank you.