Parallel Rendering on Hybrid Multi-GPU Clusters

Stefan Eilemann
Blue Brain Project
• Based on Equalizer and Collage
• Standard framework for parallel rendering
• Per process threads: main, receive, command, image transmit
• Per GPU threads: render, transfer/async readback
Hybrid Multi-GPU Clusters

- Mixed shared/distributed memory
- NUMA topology within node
- Cost-effective
- We’re back in ’95, and got a cluster
• 13 nodes,  
  2x Xeon X5690,  
  6 cores, 3.47GHz  
• 2x 12GB RAM  
• 3x GTX580, 
  3GB RAM  
• 10 GBit ethernet  
• Used 11 nodes
• **Synthetical:** eqPly
  – PLY renderer using kd-tree
  – 4x David 1mm, >200MTris
  – Realistic camera path

• **Real-world:** RTNeuron
  – Visualizes neocortical column simulations
  – Almost worst-case data structure
  – Transparency, LOD, CUDA-based culling
• Round-robin decomposition
  – Better load balance
  – RGB+Depth compositing
  – No transparency

• Spatial decomposition
  – kd-tree with #GPU leaves, clip planes
  – Compact regions
  – RGBA compositing
DB, Full Cortical Column

Frames per Second

Number of GPUs

speedup due to optimization

- spatial
- round-robin
- speedup

May 2012

Blue Brain Project - Stefan Eilemann
• Readback penalty
  – GPU 1 -> Processor 1: ~250MPx/s
  – GPU 1 -> Processor 2: ~120MPx/s
• **Automatic thread affinity**
  – Render and readback threads to GPU ‘processor’
  – IO threads to primary network interface ‘processor’
• **Based on hwloc library and X11 extension**

**NV_Control**

May 2012  
Blue Brain Project - Stefan Eilemann
Thread Placement

2D, Thread Affinity, 4xDavid

- incorrect
- correct
- speedup

Frames per Second

Number of GPUs

Speedup due to optimization

May 2012
Asynchronous Readback

- Pipeline GPU->CPU transfer with next frame
- One additional, lazy transfer thread per GPU
- Extension of compression plugin API
2D, Asynchronous Readback, 4xDavid

Asynchronous Readback

Frames per Second

Speedup due to optimization

Number of GPUs

May 2012

Blue Brain Project - Stefan Eilemann
• Reduce pixel data during compositing
• Optimize 2D load-balancer
  – refined load grid
  – less oscillation
Round-Robin DB, Region of Interest, Full Cortical Column

Frames per Second vs. Number of GPUs

speedup due to optimization

-2.0% -1.3% -0.7% 0% 0.7% 1.3% 2.0% 2.7% 3.3% 4.0%

3 9 15 21 27 33

ROI off
ROI on
speedup

May 2012
Blue Brain Project - Stefan Eilemann
Spatial DB, Region of Interest, Full Cortical Column

- ROI off
- ROI on
- speedup

Frames per Second:
- 3: 1
- 9: 2
- 15: 1
- 21: 3
- 27: 5
- 33: 6

Speedup due to optimization:
- 30%
- 25%
- 20%
- 15%
- 10%
- 5%
- 0%
- -5%

Number of GPUs:
- 3
- 9
- 15
- 21
- 27
- 33
2D, Region of Interest, 4xDavid

Frames per Second

Number of GPUs

Speedup due to optimization

ROI off
ROI on
Speedup

May 2012
Blue Brain Project - Stefan Eilemann
Multi-Thread vs Multi-Process

- Multi-process ‘MPI mode’
  - Increased memory usage, especially for sort-first
  - Increased inter-node communication cost

- Multi-threaded
  - Driver overhead
  - Memory bandwidth contention for sort-first
Multi-Thread vs Multi-Process

2D, Multi-Process, 4xDavid

Frames per Second

Number of GPUs

Speedup due to optimization

Multithreaded
Multiprocess
speedup

May 2012
Blue Brain Project - Stefan Eilemann
Conclusions

• Order of importance:
  – glFinish
  – Async readback (2D) or ROI (DB)
  – Thread placement

• User shouldn’t need to care

• Time-consuming to implement all of them
Future Work

- RDMA support and benchmarking
- RTNeuron view frustum culling improvements
- Subpixel FSAA compounds for RTNeuron
  - Improve visual quality, not performance
- Asynchronous uploads
Acknowledgements

- Blue Brain Project, EPFL; CeSViMa, UPM; Visualization and MultiMedia Lab, University of Zürich
- Digital Michelangelo Project, Stanford 3D Scanning Repository
- Swiss National Science Foundation grant 200020-129525
- Spanish Ministry of Science and Innovation grant TIN2010-21289-C02- 01/02
- http://www.open-mpi.org/projects/hwloc/
- http://github.com/Eyescale/Equalizer/