Blue
Brain
Project

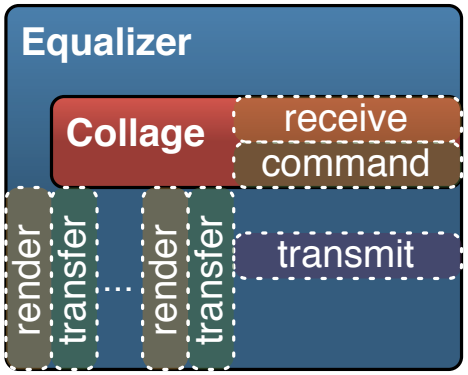
Parallel Rendering on Hybrid Multi-GPU Clusters

Stefan Eilemann

Blue Brain Project

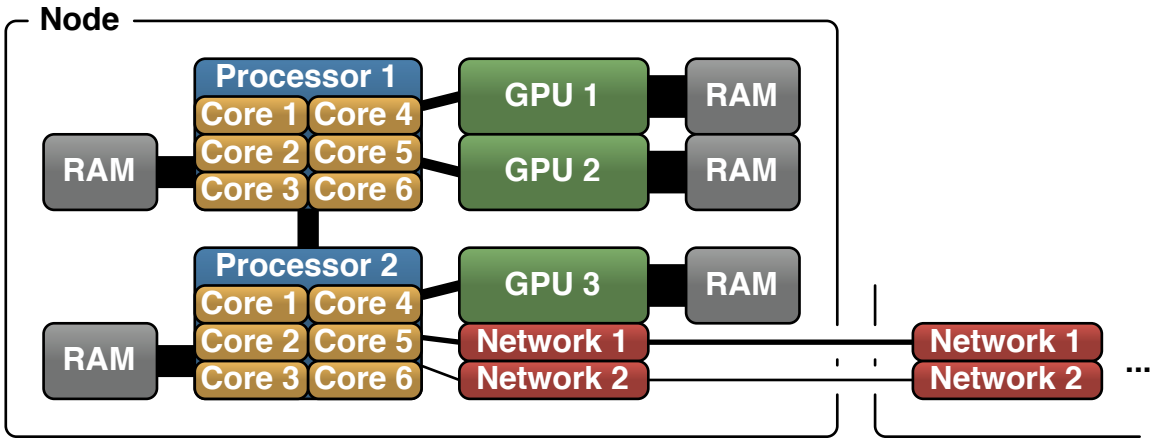
Parallel Rendering

- 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

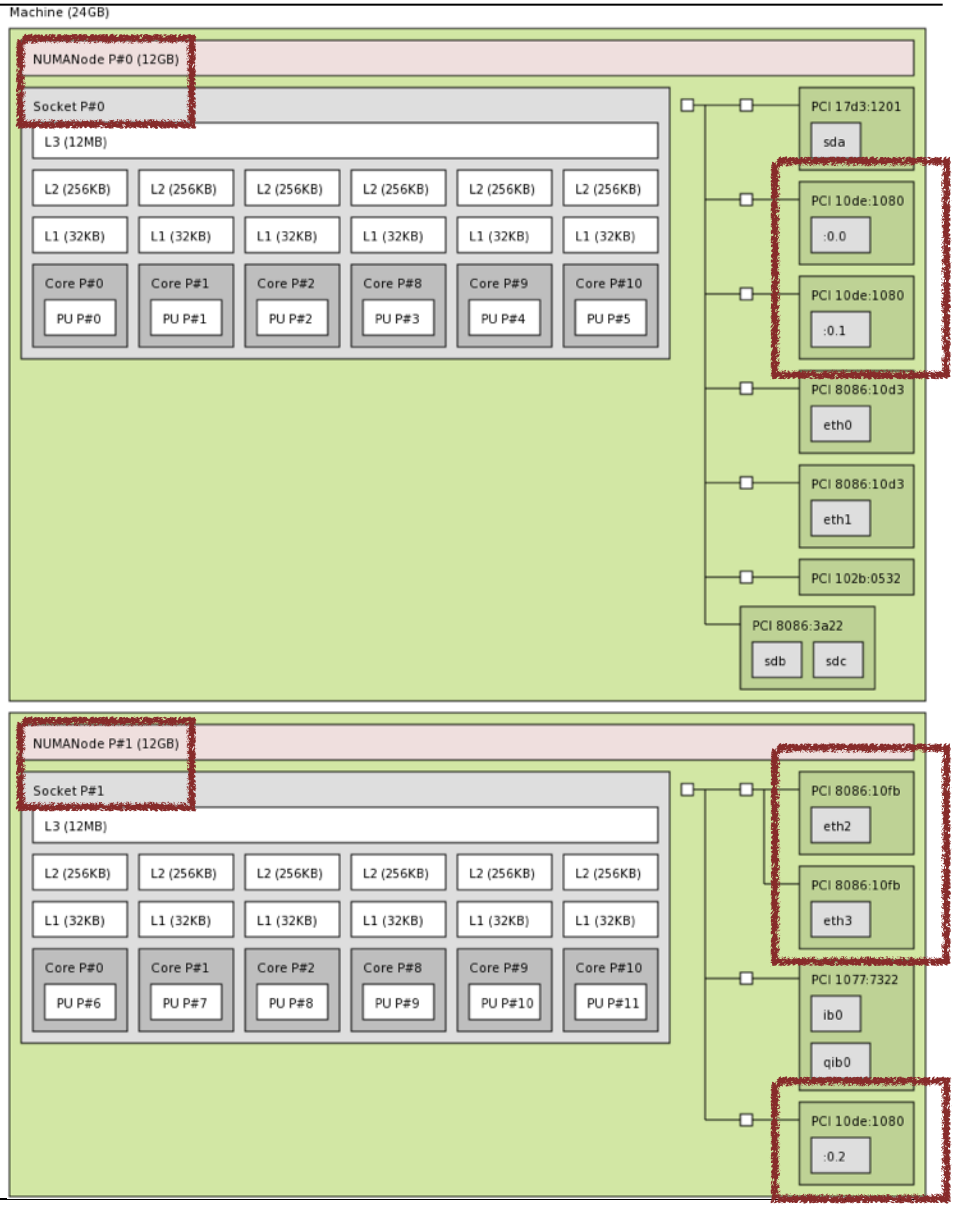
- Mixed shared/distributed memory
- NUMA topology within node
- Cost-effective
- We're back in '95, and got a cluster



Blue Brain Project

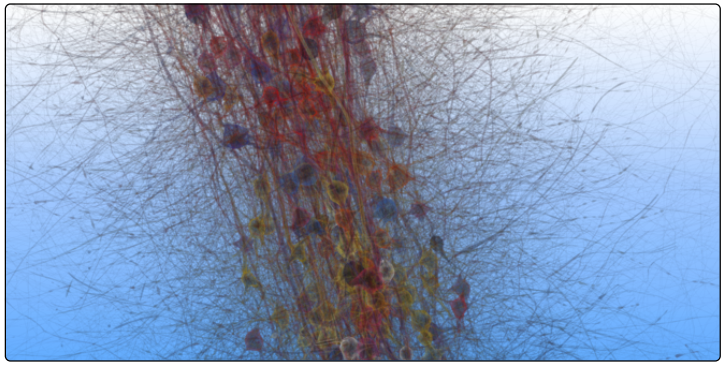
Benchmark Hardware

- 13 nodes,
  2x Xeon X5690,
  6 cores, 3.47GHz
- 2x 12GB RAM
- 3x GTX580,
  3GB RAM
- 10 GBit ethernet
- Used 11 nodes



Benchmark Software

- Synthetical: eqPly
  - PLY renderer using kd-tree
  - 4x David 1mm, >200MTris
  - Realistic camera path

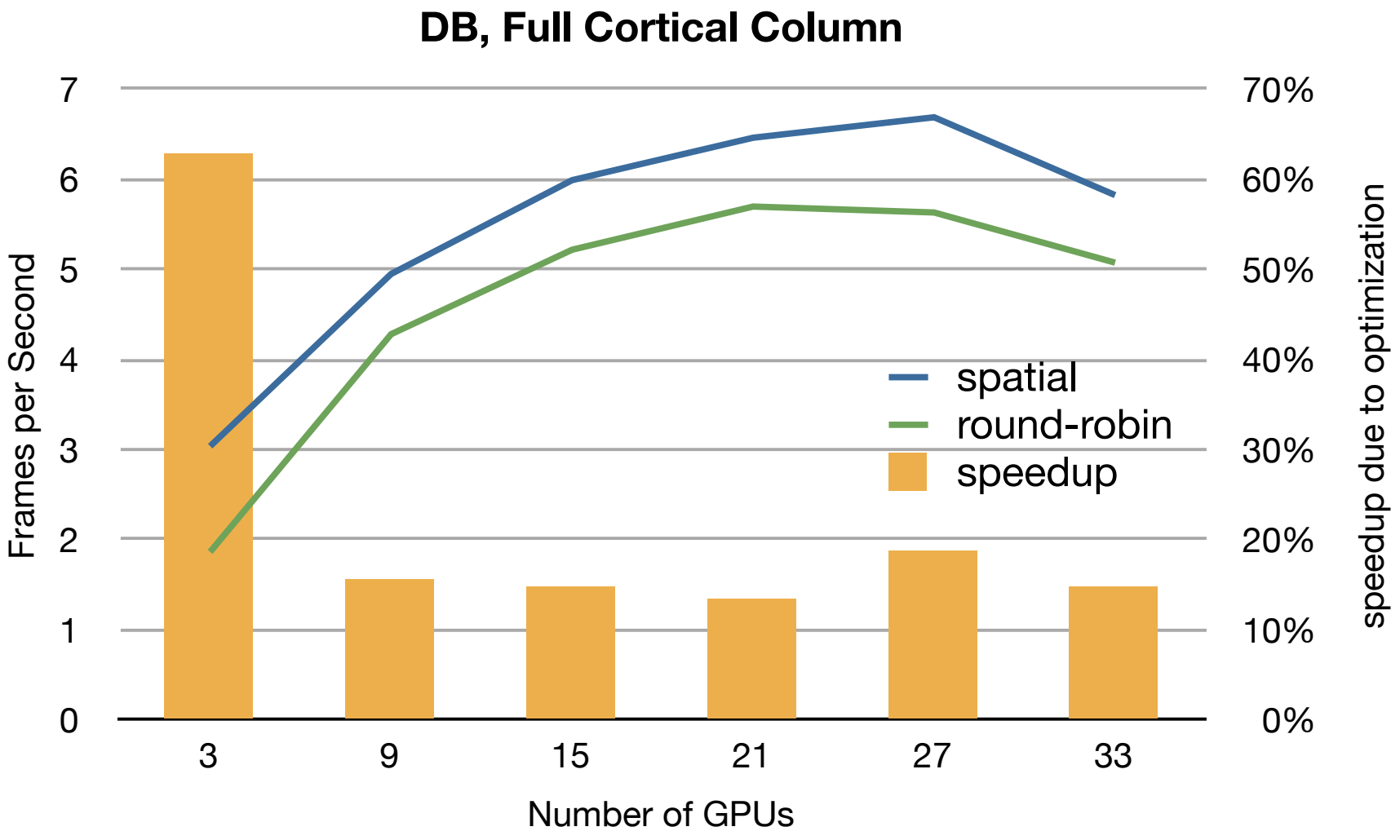


- Real-world: RTNeuron
  - Visualizes neocortical column simulations
  - Almost worst-case data structure
  - Transparency, LOD, CUDA-based culling

RTNeuron Sort-Last

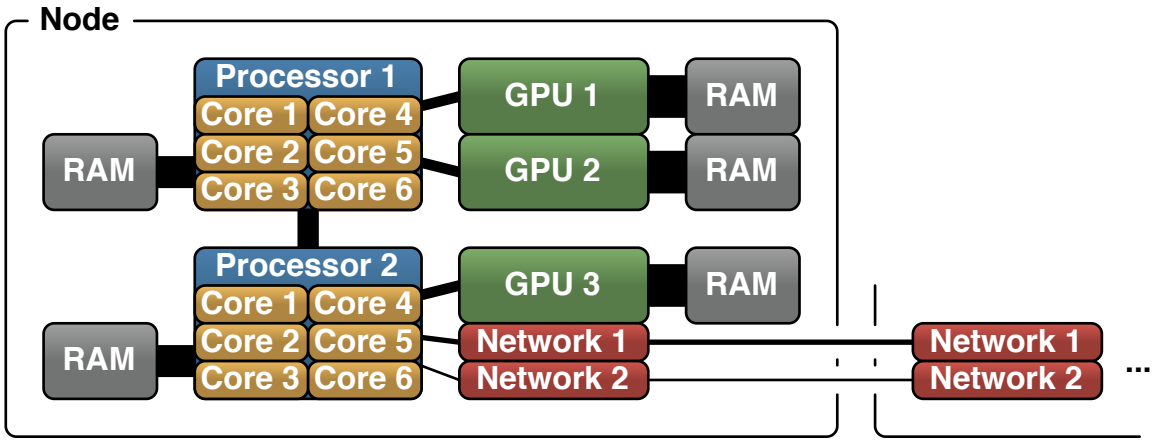
- Round-robin decomposition
  - Better load balance
  - RGB+Depth compositing
  - No transparency
- Spatial decomposition
  - kd-tree with #GPU leaves, clip planes
  - Compact regions
  - RGBA compositing

RTNeuron Sort-Last



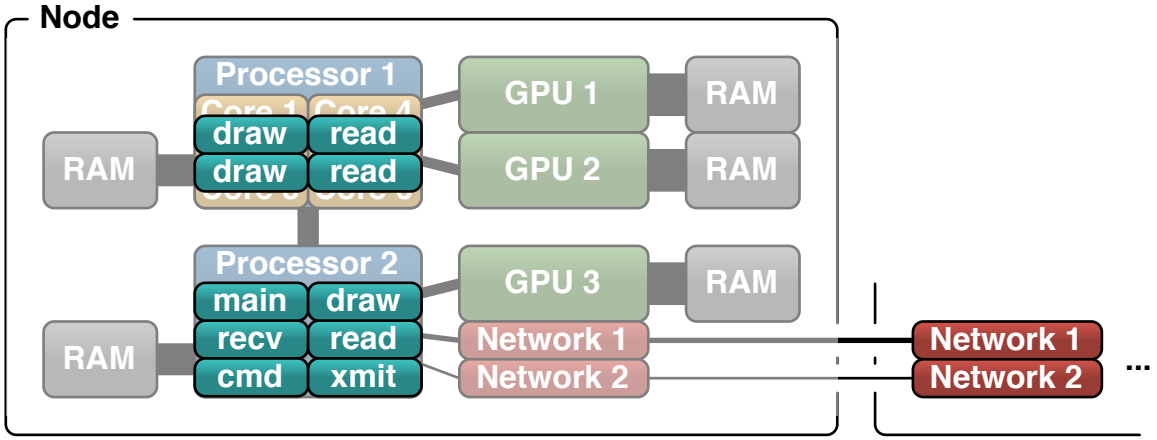
Thread Placement

- Readback penalty
  - GPU 1 -> Processor 1: ~250MPx/s
  - GPU 1 -> Processor 2: ~120MPx/s

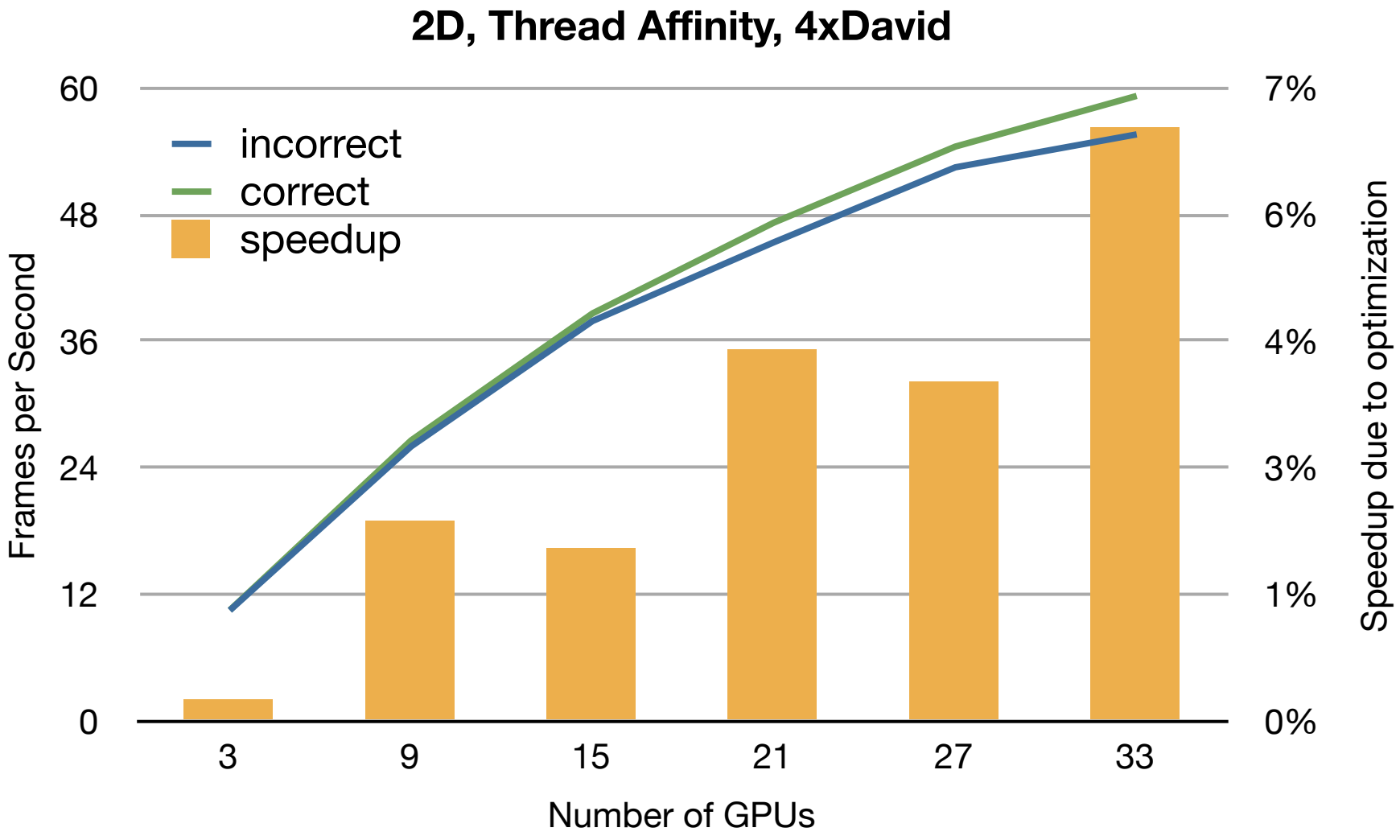


Thread Placement

- 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

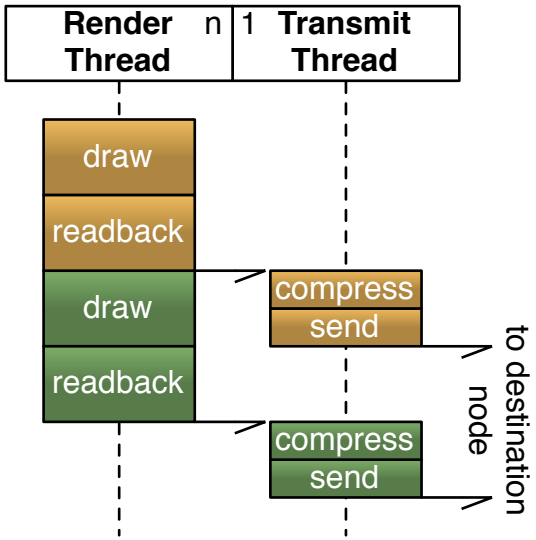


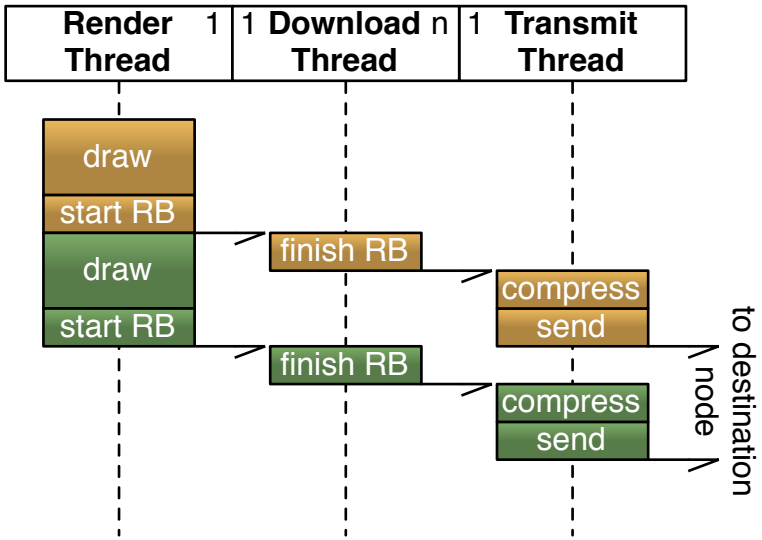
Thread Placement



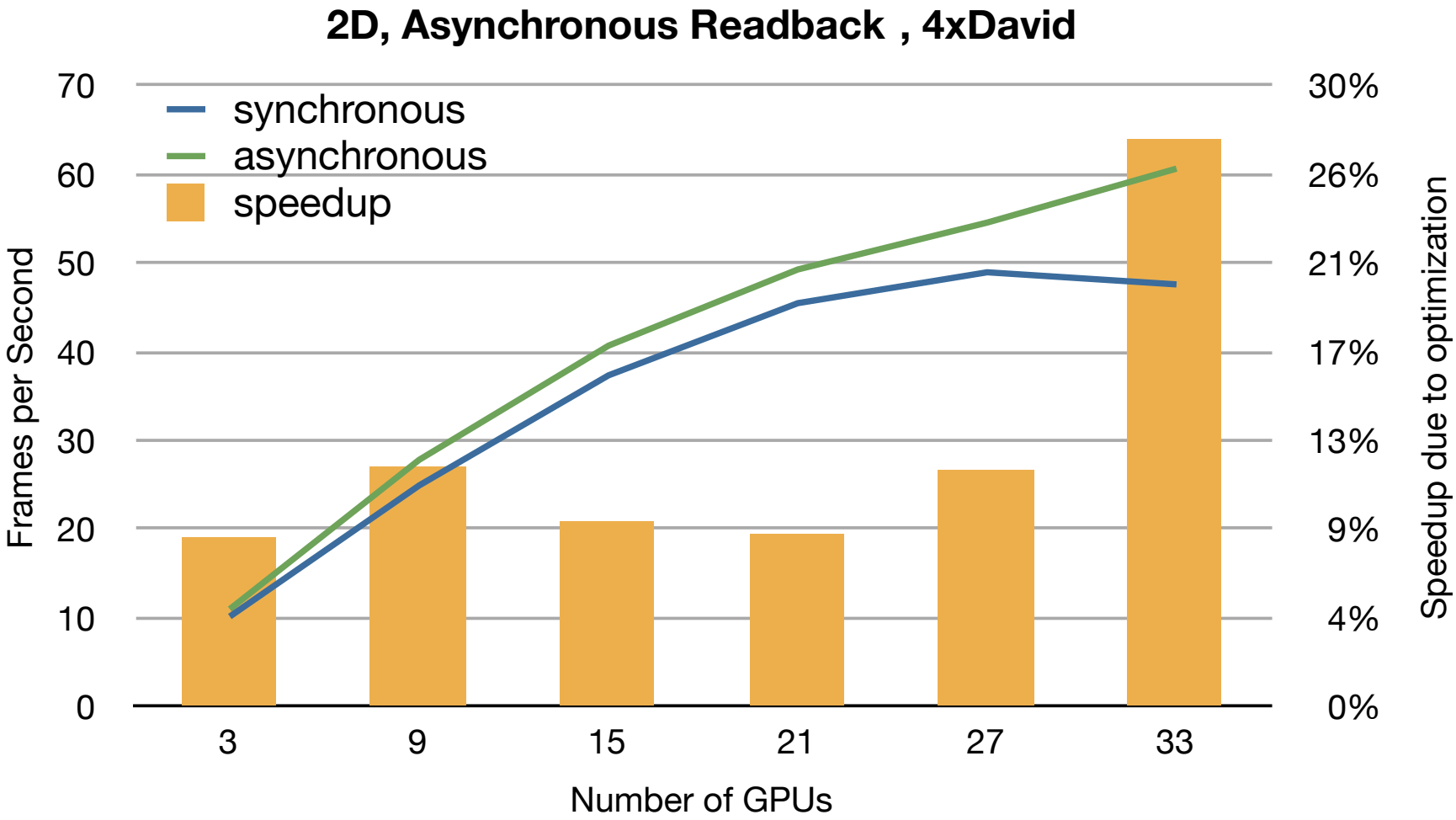
Asynchronous Readback

- Pipeline GPU->CPU transfer with next frame
- One additional, lazy transfer thread per GPU
- Extension of compression plugin API



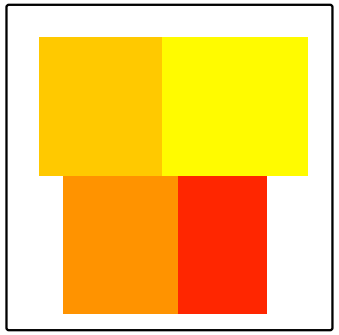


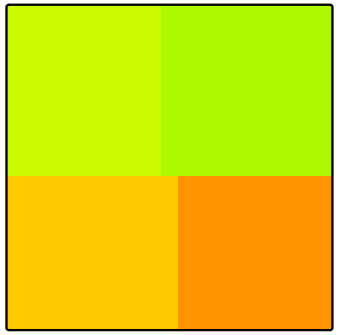
Asynchronous Readback

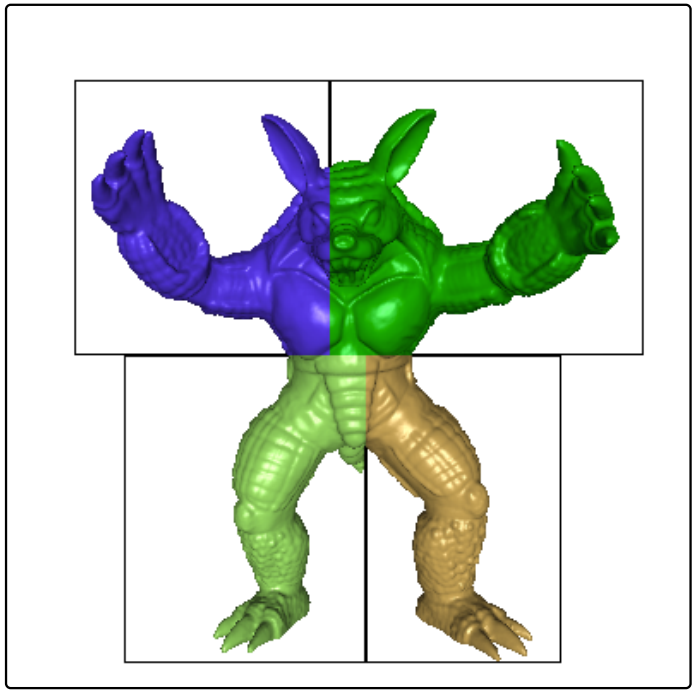


Region of Interest

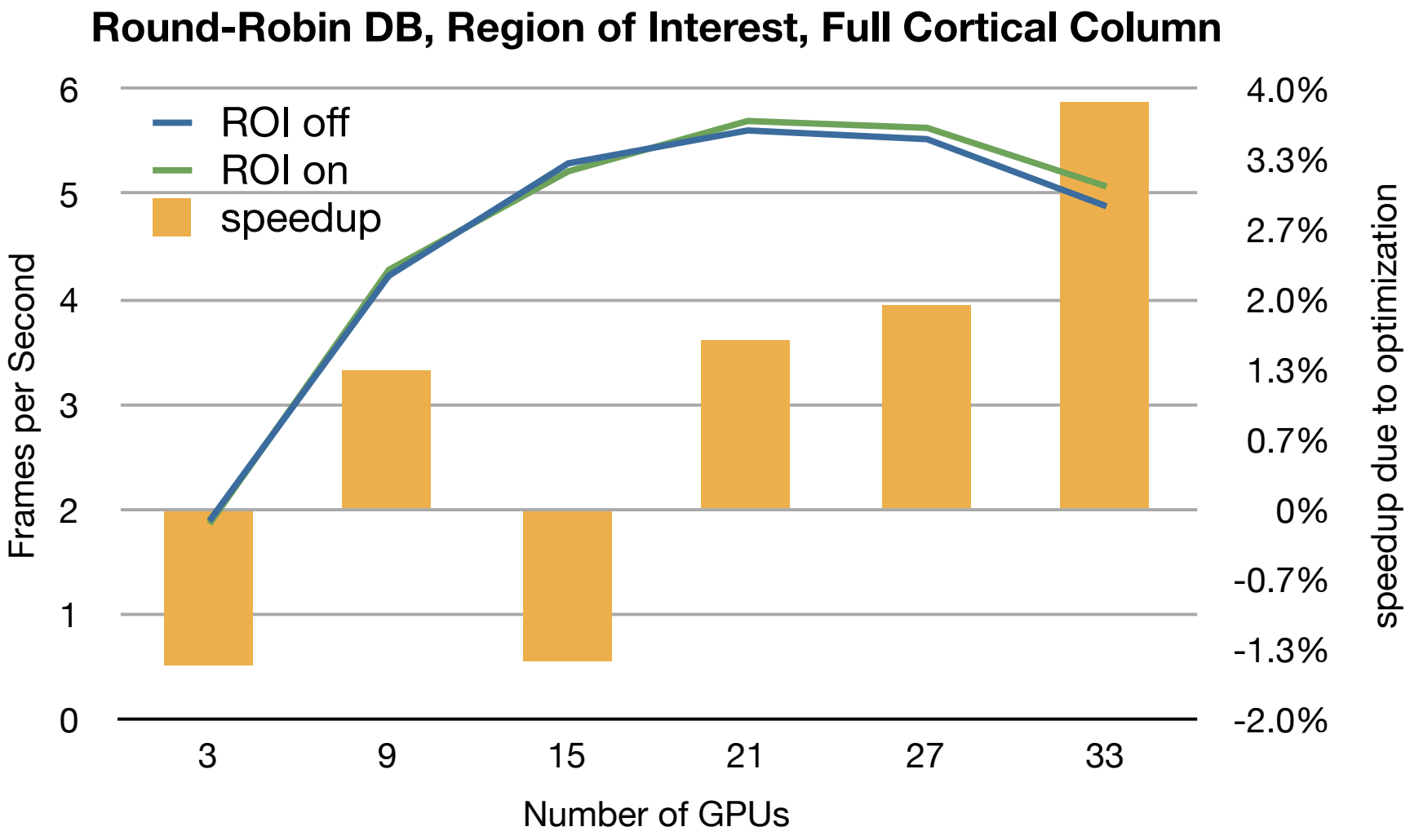
- Reduce pixel data during compositing
- Optimize 2D load-balancer
  - refined load grid
  - less oscillation



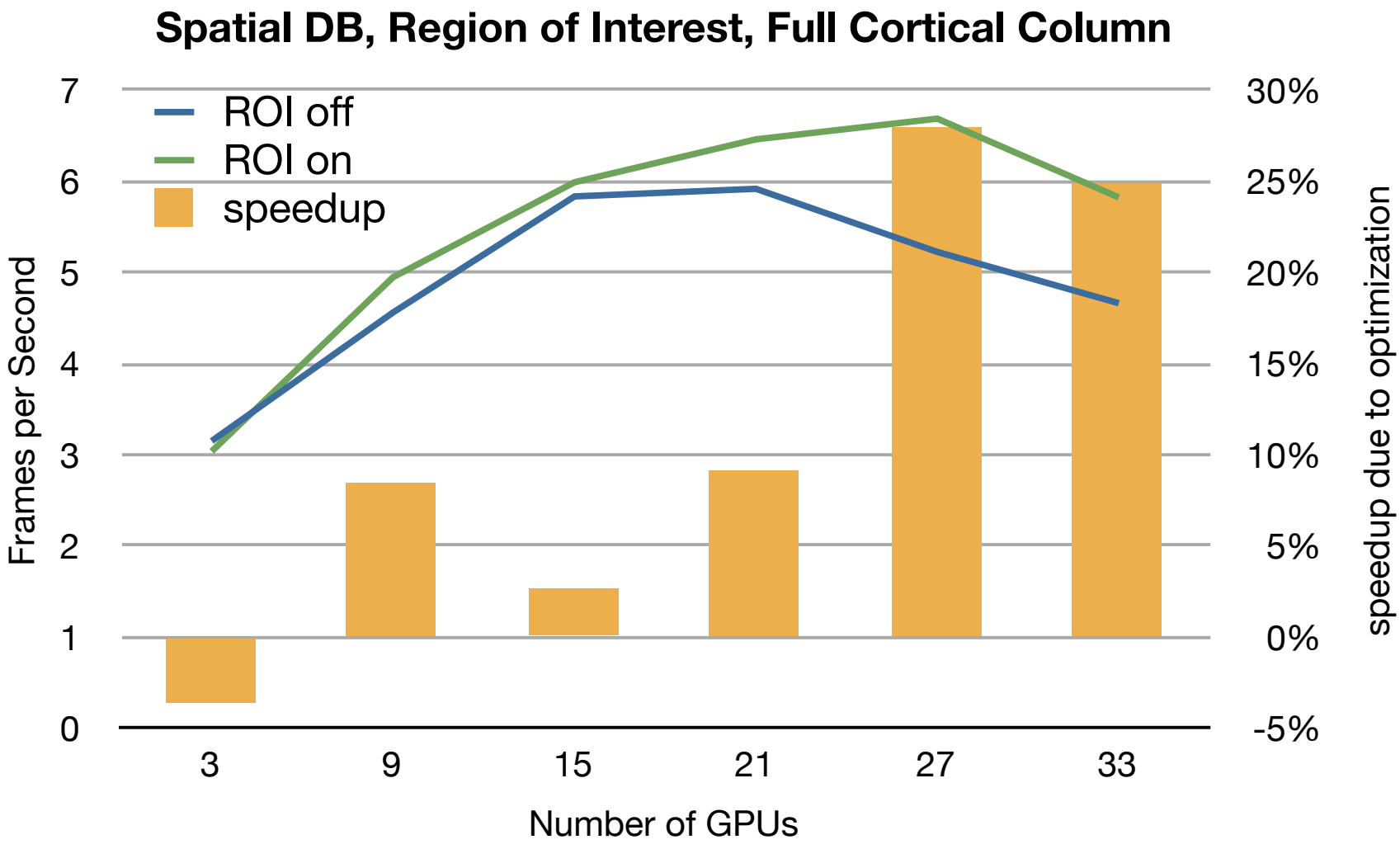




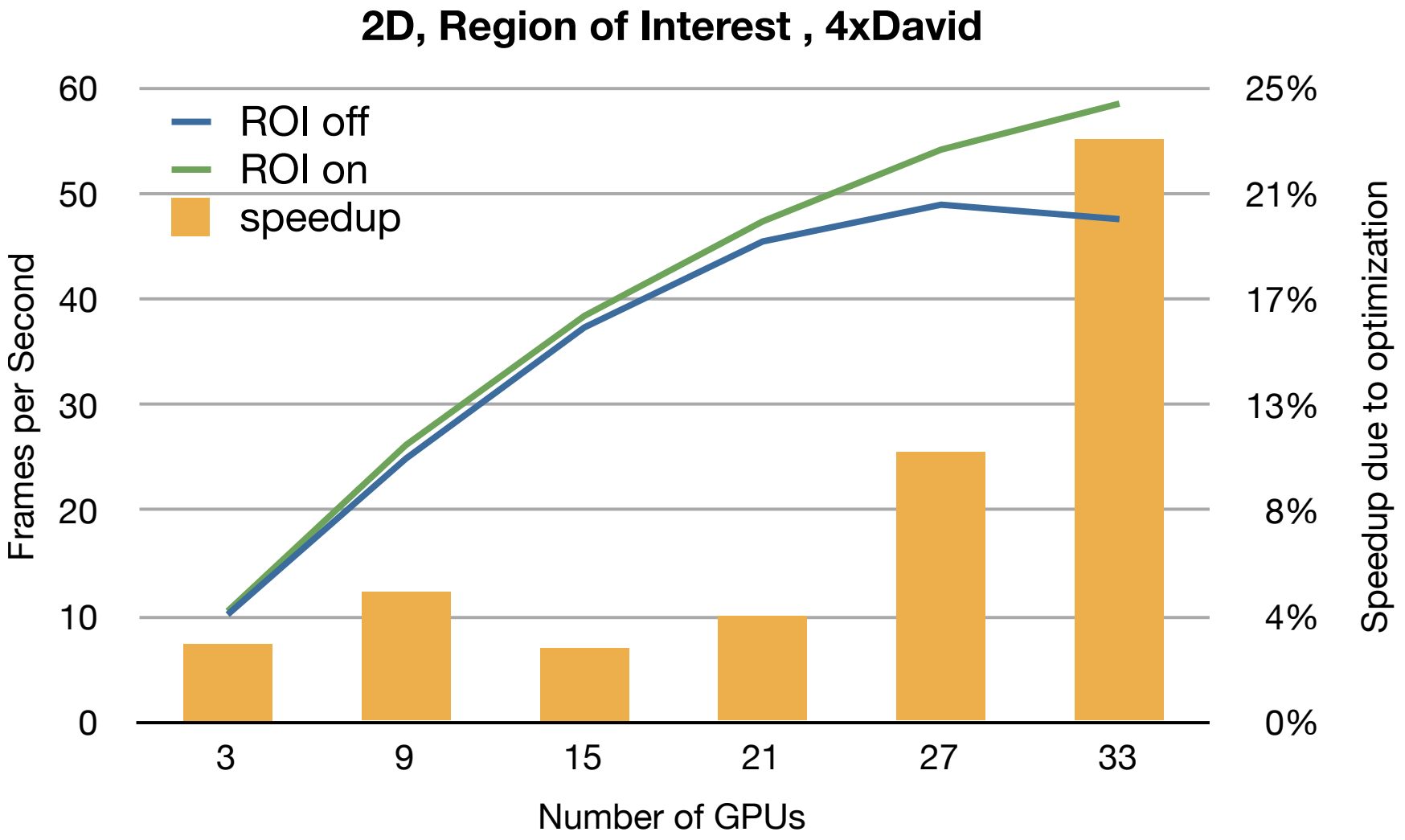
Region of Interest



Region of Interest



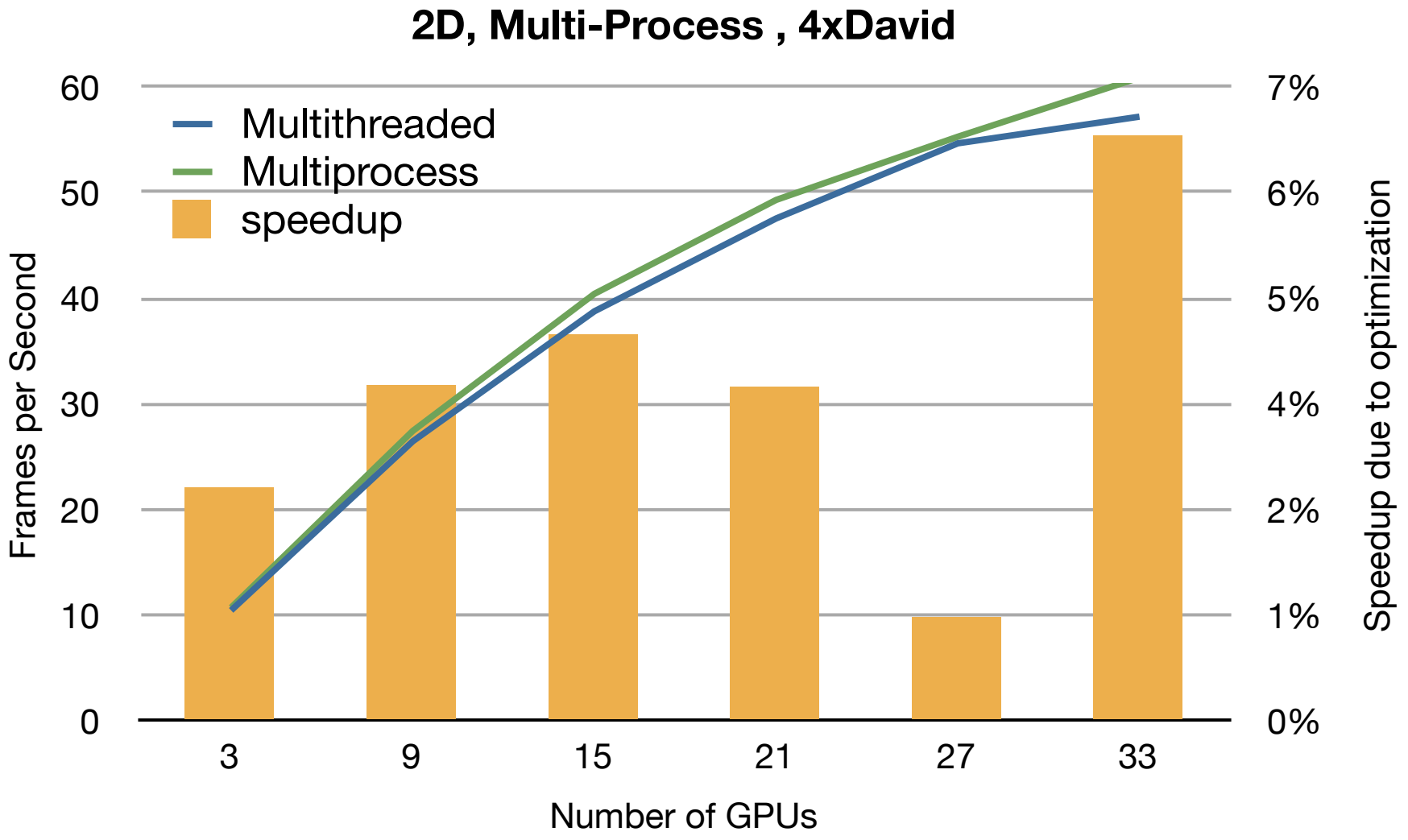
Region of Interest



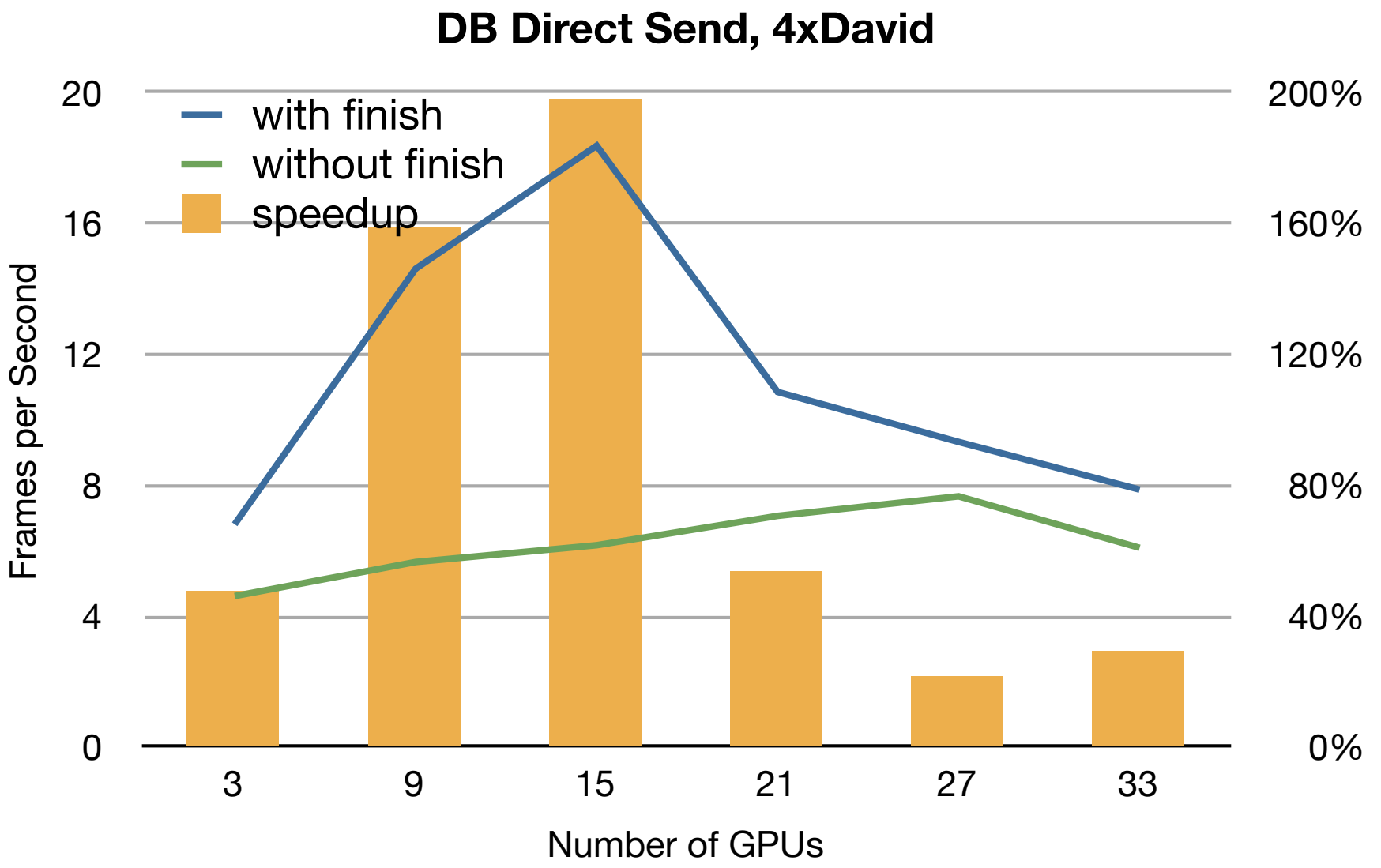
Multi-Thread vs Multi-Process

- 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



glFinish


DB Direct Send, 4xDavid
with finish
without finish
speedup
Frames per Second
Number of GPUs
0
4
8
12
16
20
0%
40%
80%
120%
160%
200%
3
9
15
21
27
33

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/