Fast Compositing for Cluster-Parallel Rendering

M. Makhinya, S. Eilemann, R. Pajarola

University of Zurich
Department of Informatics

Content

- Motivation
- Methods
- Experimental analysis
- Conclusion

Motivation - Focus of Study

Existing Approaches

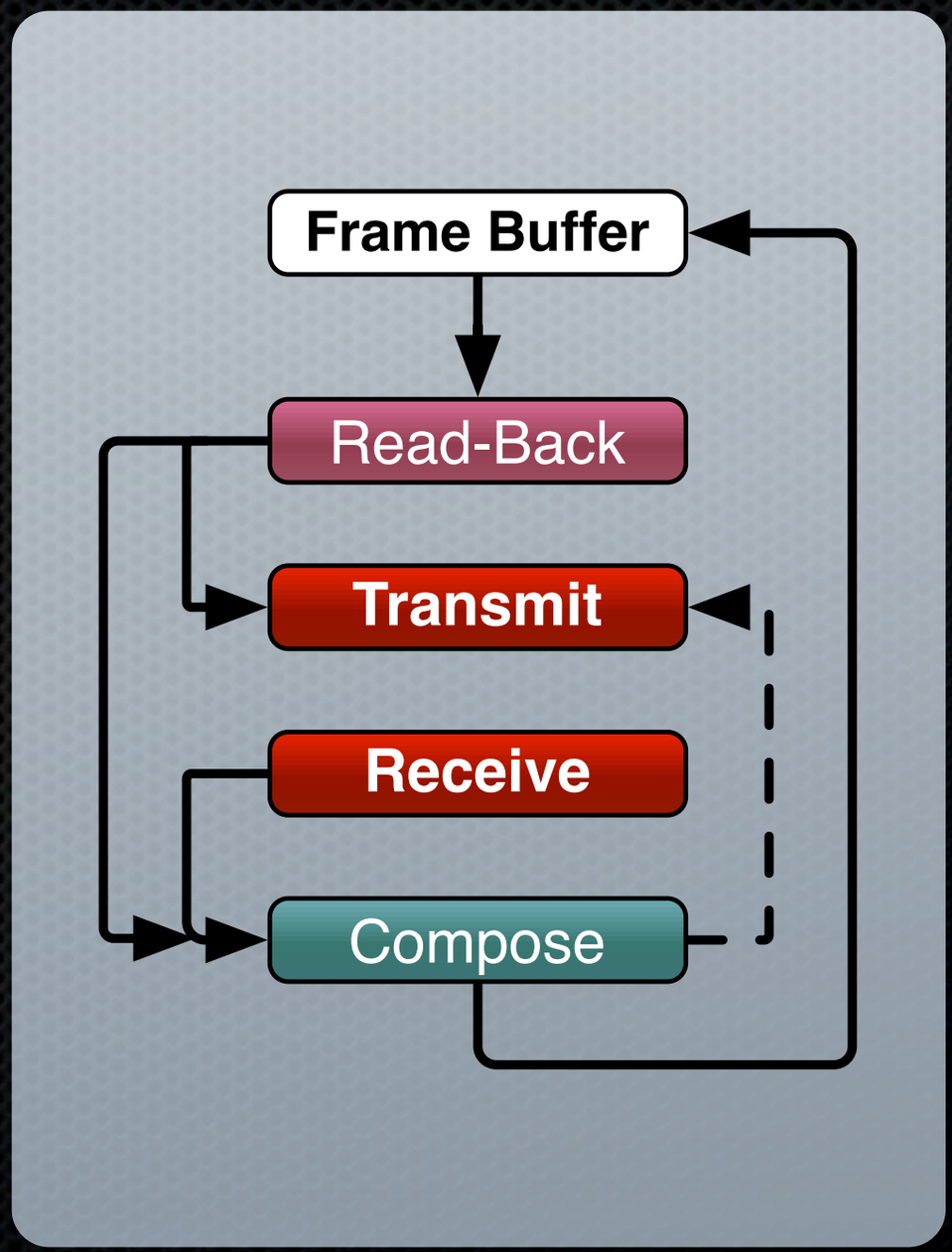
- Read-back / compositing improvement
  - Less data over network
- Data transmission improvement
  - Color / depth buffers compression

Motivation - Other Solutions

- Difficult to integrate in to existent frameworks
- Hardware dependent
- Doesn’t necessarily improve performance on modern hardware

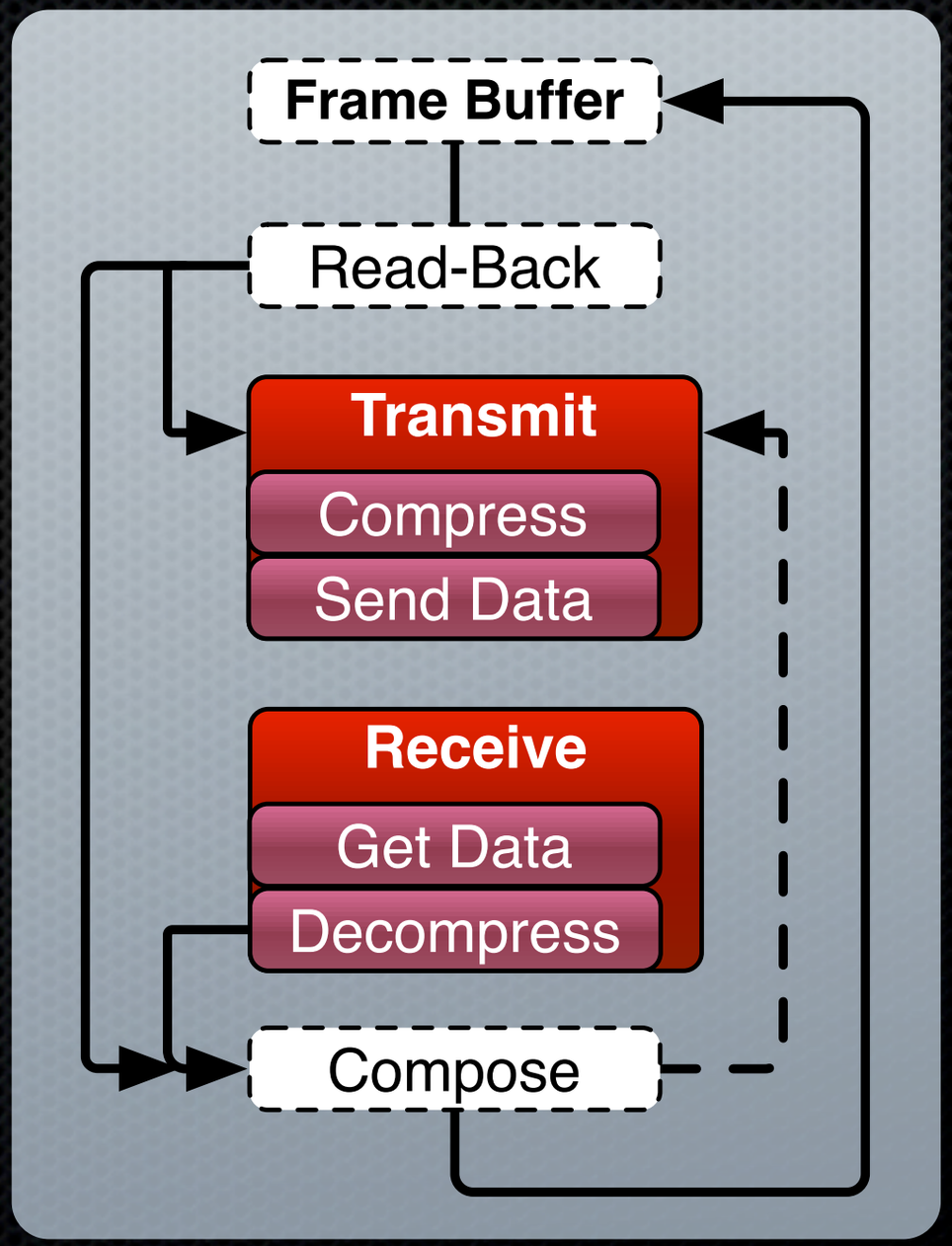
Methods - Equalizer (EQ)

Typical
compositing loop
in parallel rendering



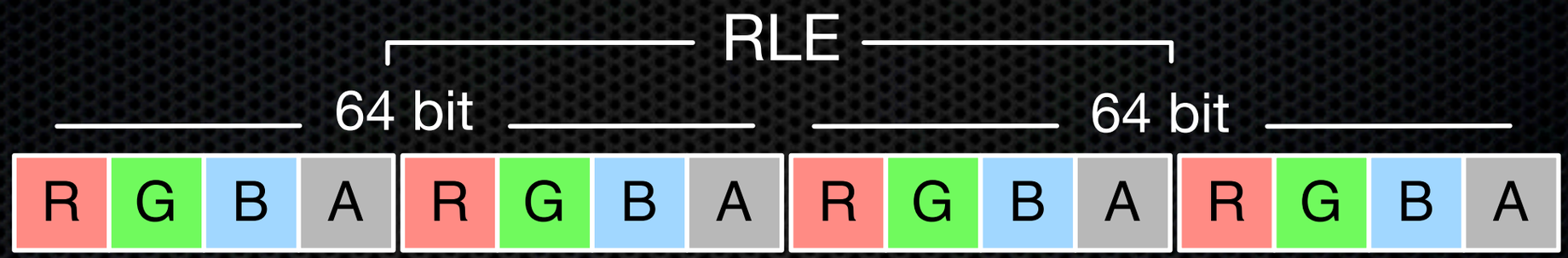
Methods - Equalizer (EQ)

Typical
compositing loop
in parallel rendering



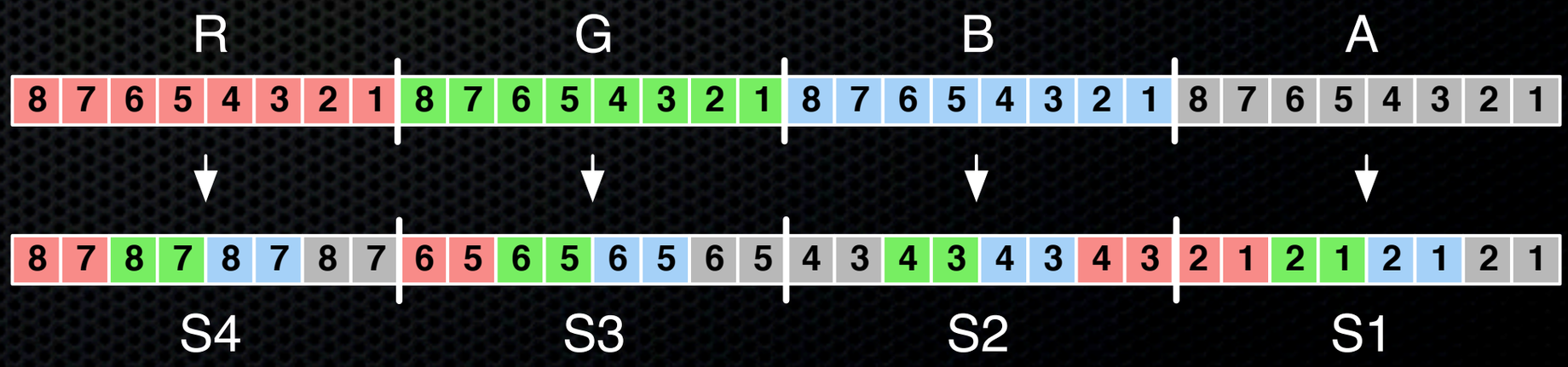
Methods - RLEs

64 bit

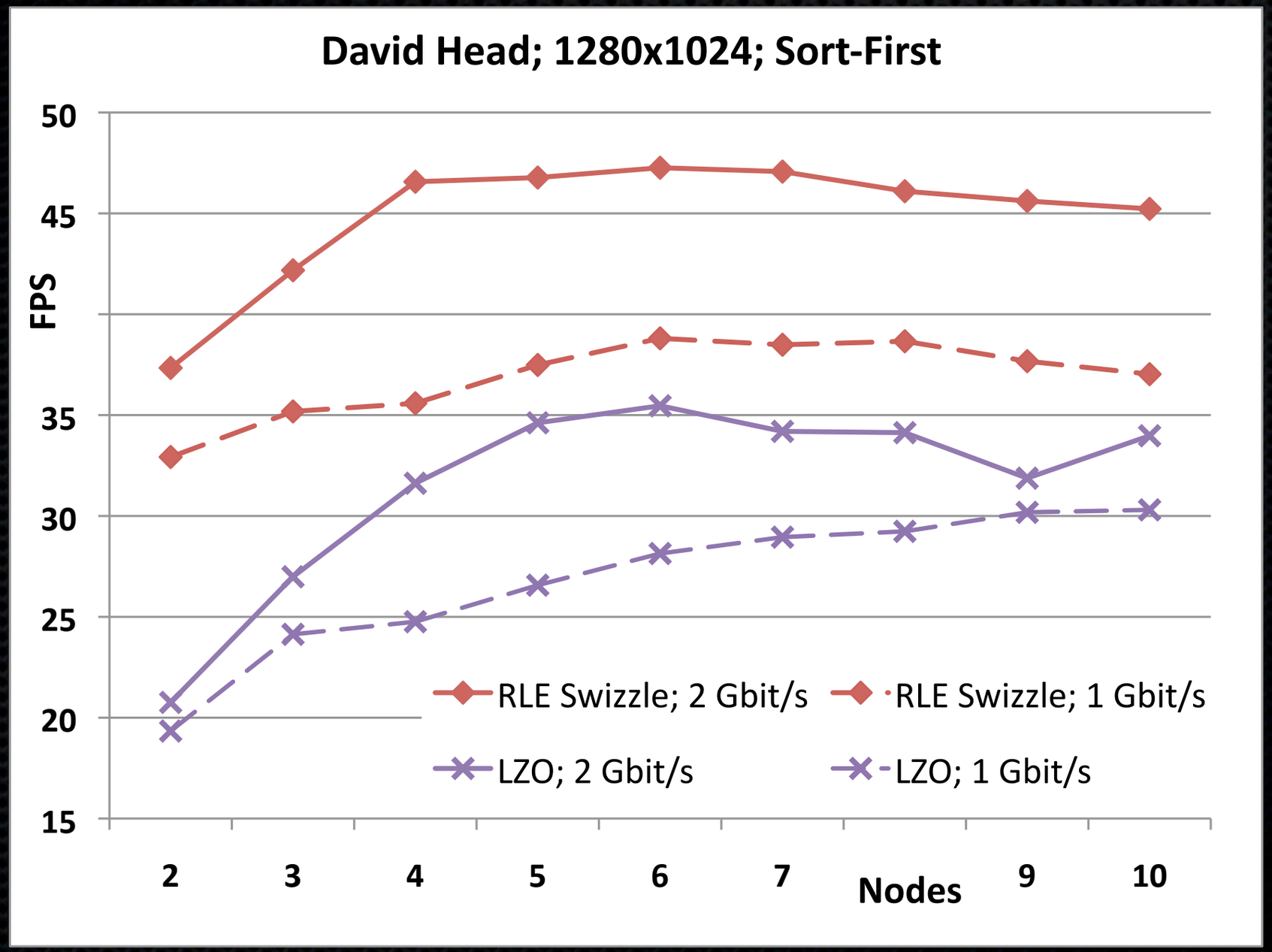


Component

Swizzle



Methods - RLE vs LZO



Methods - YUV12

R_1	G_1	B_1	A_1	R_2	G_2	B_2	A_2
R_3	G_3	B_3	A_3	R_4	G_4	B_4	A_4

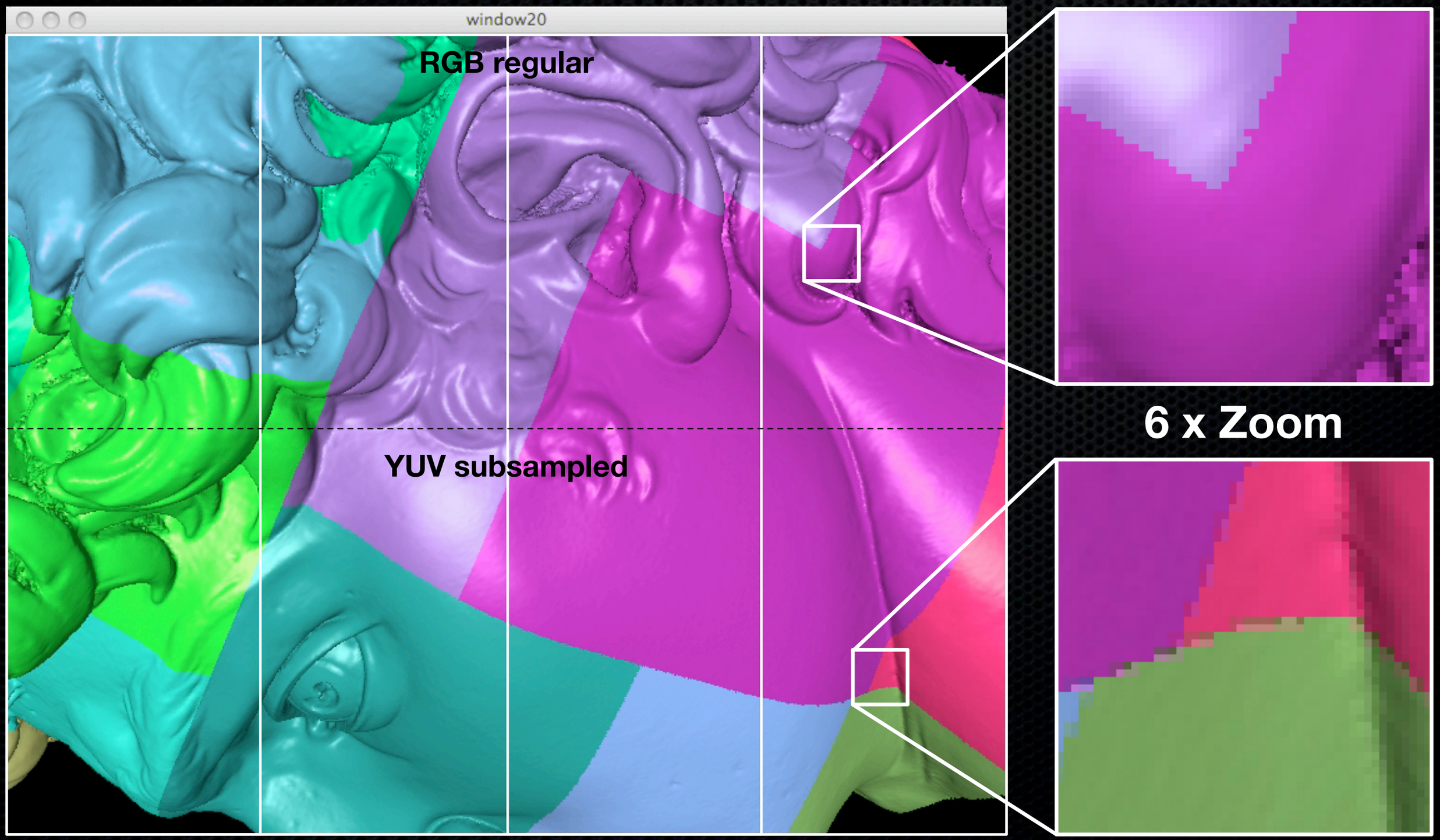
RGB to YUV

Y_1	U_1	V_1	A_1	Y_2	U_2	V_2	A_2
Y_3	U_3	V_3	A_3	Y_4	U_4	V_4	A_4

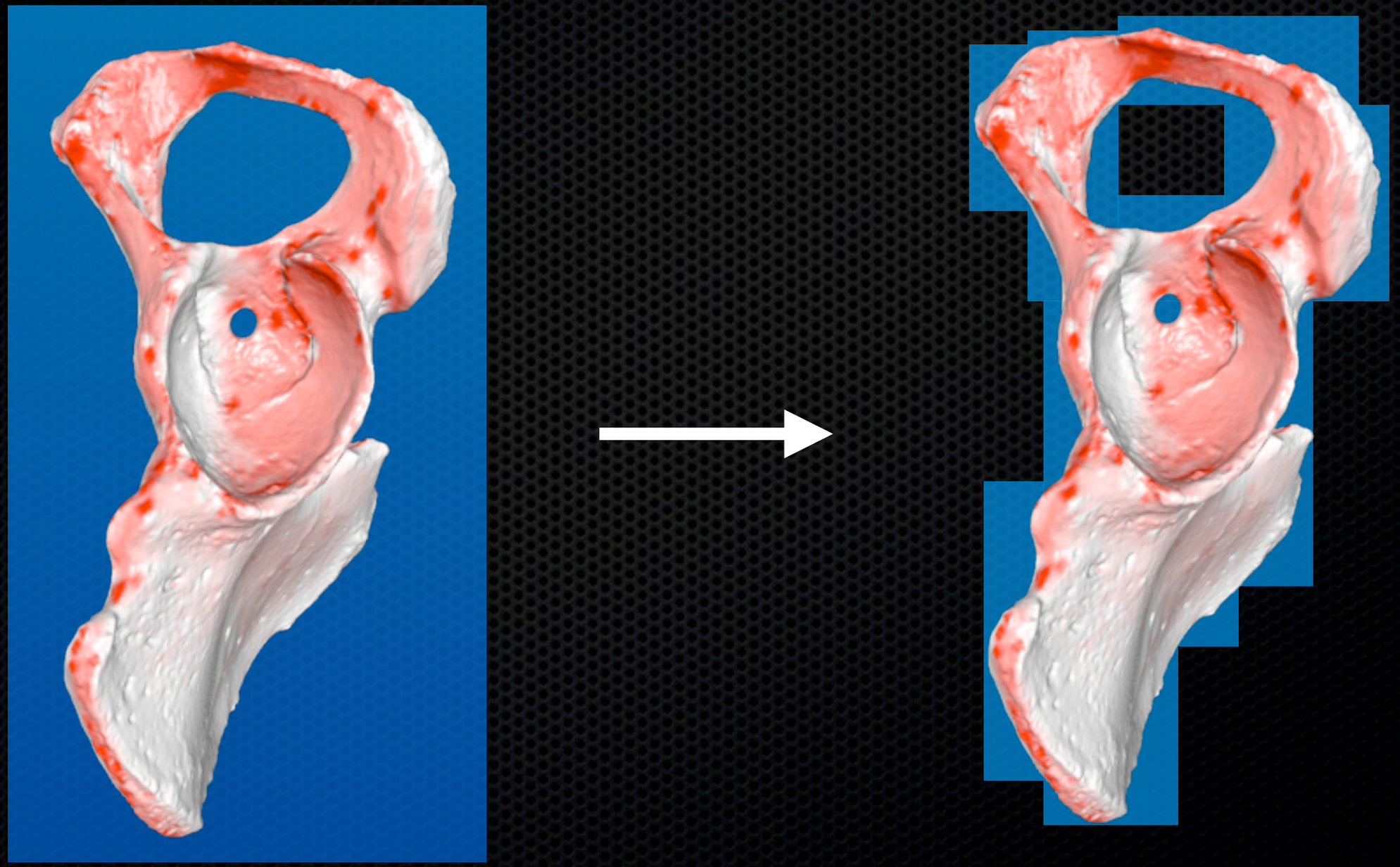
Average

Y_1	Y_2	U_n	A_n
Y_3	Y_4	V_n	A_n

Methods - YUV12 (Example)

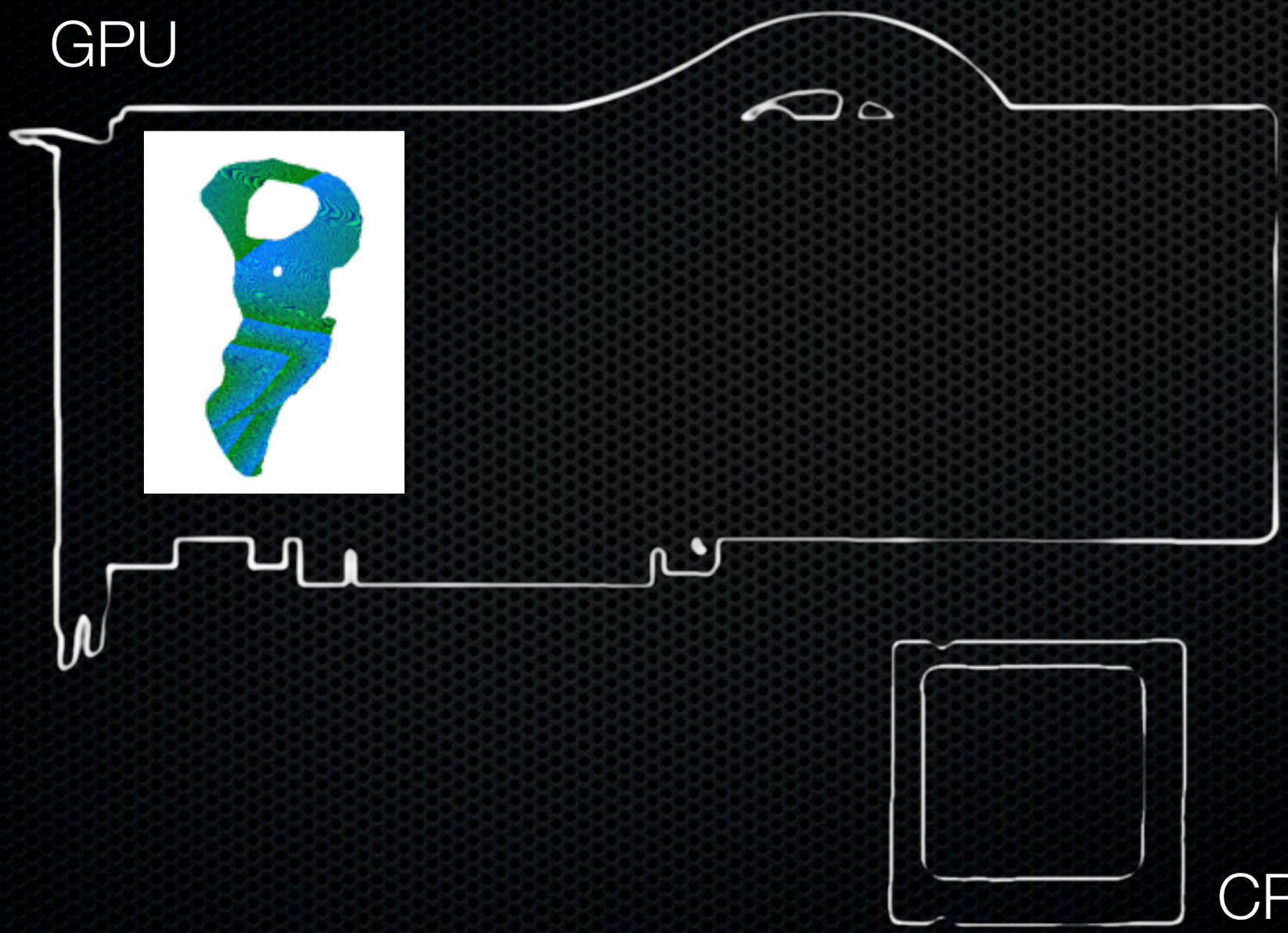


Methods - ROI (Idea)

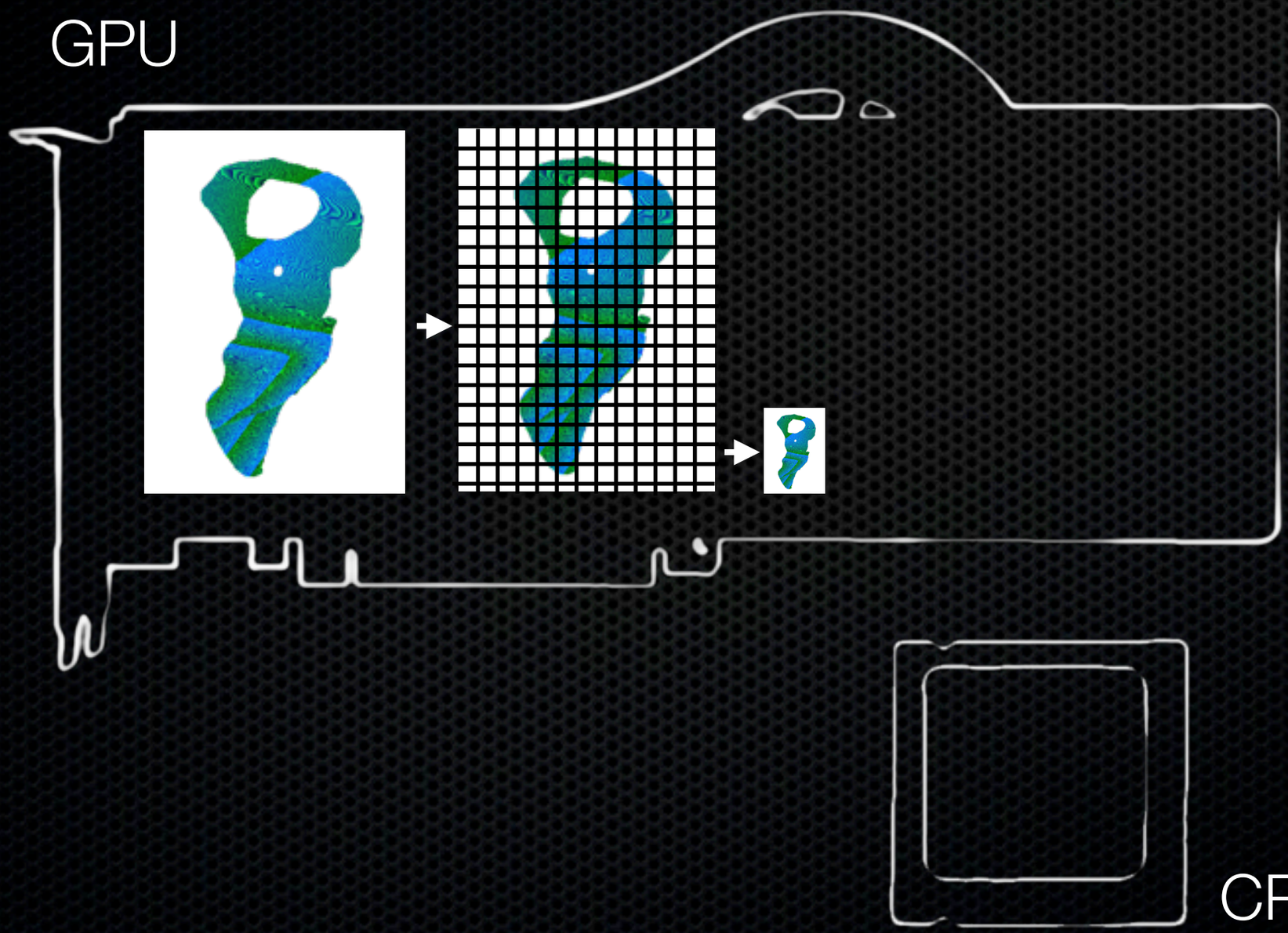


VISUALIZATION AND MULTIMEDIA LAB

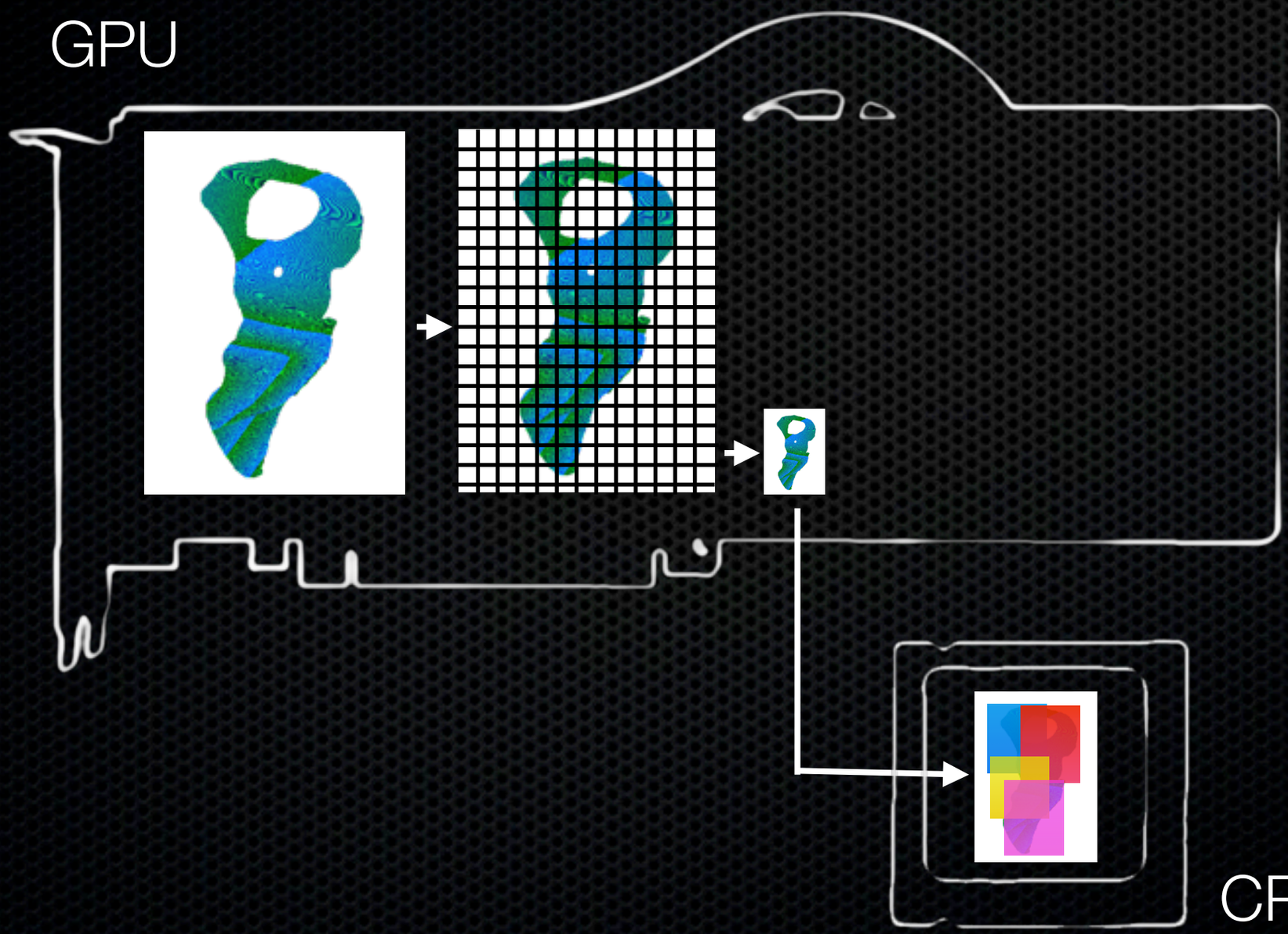
Methods - ROI



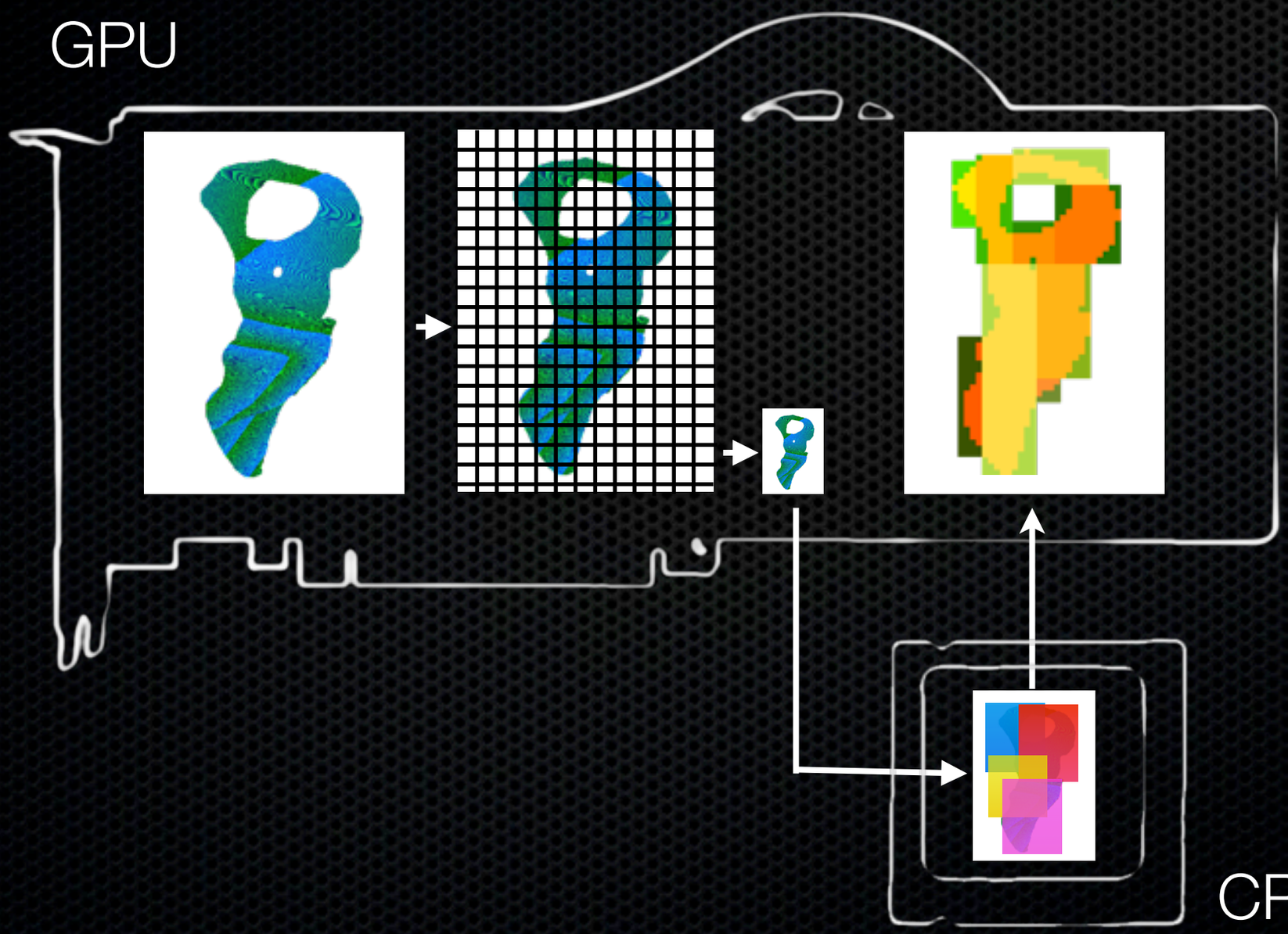
Methods - ROI



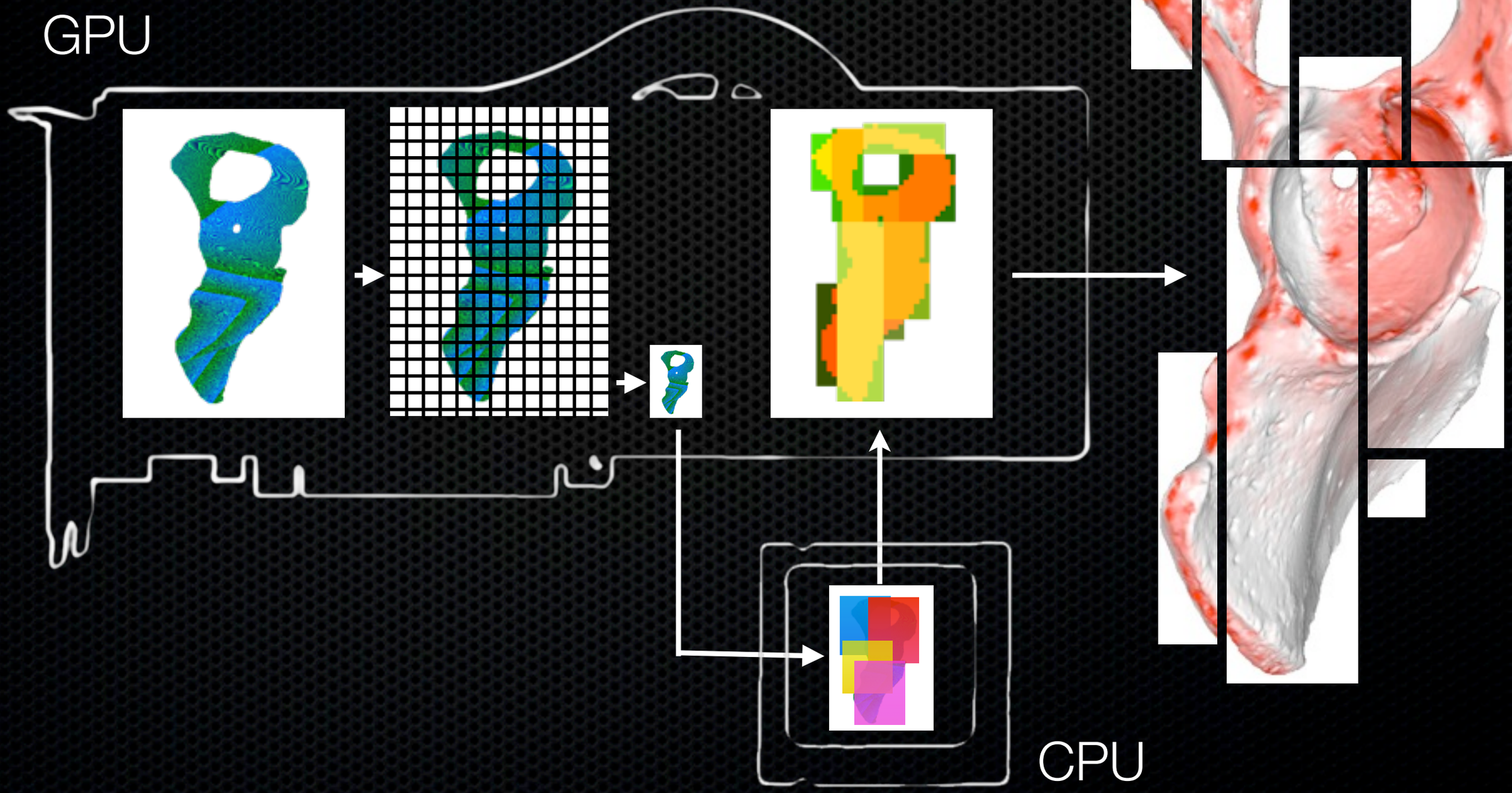
Methods - ROI



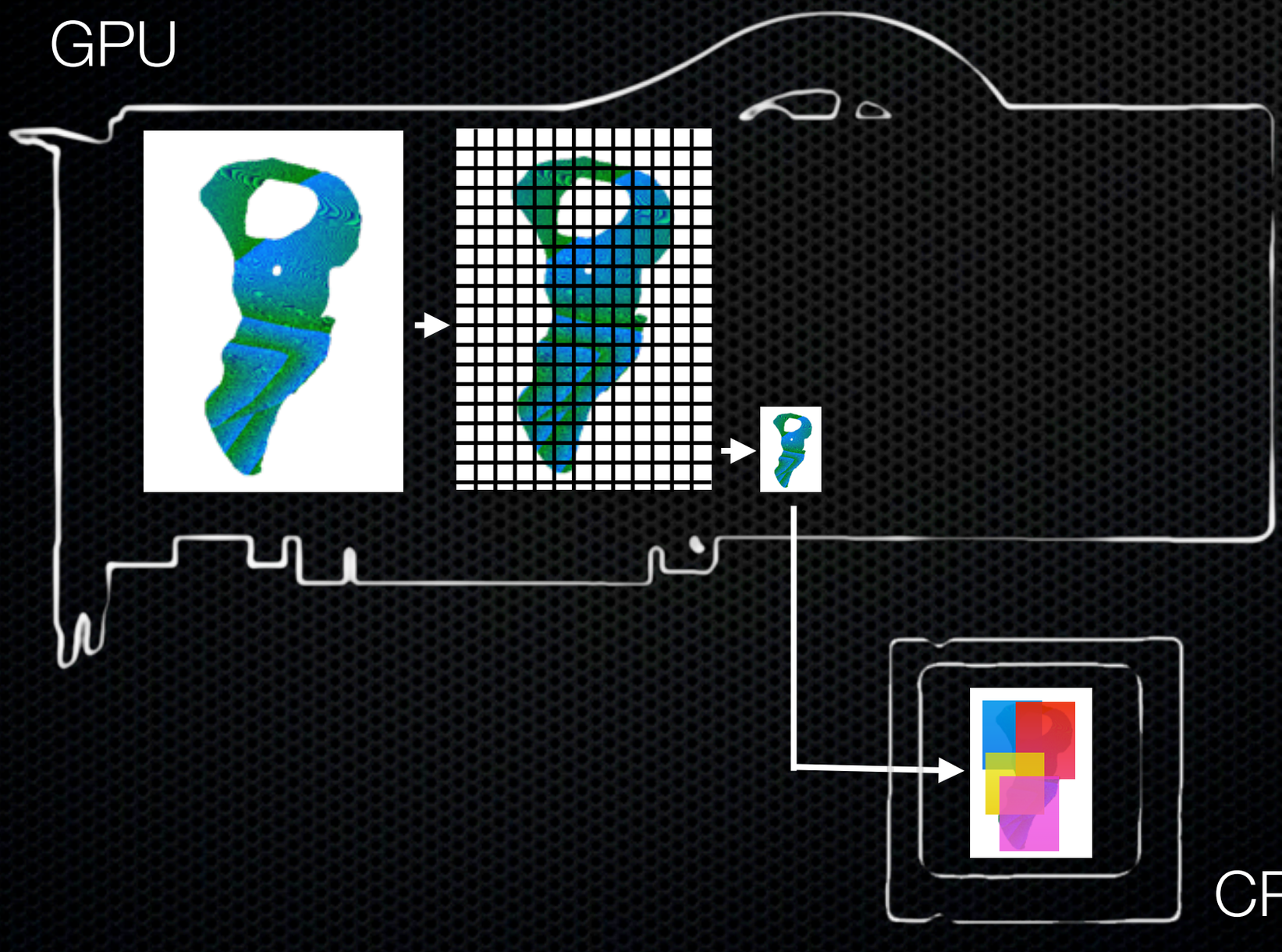
Methods - ROI



Methods - ROI



Methods - ROI



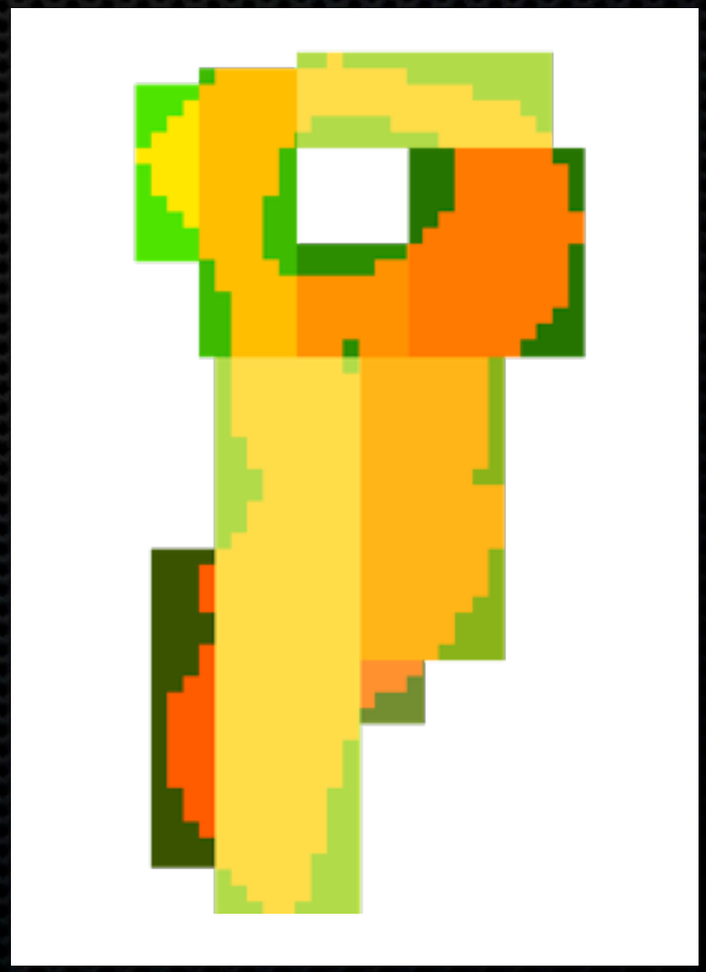
ROI speed:

~600 fps

Methods - ROI

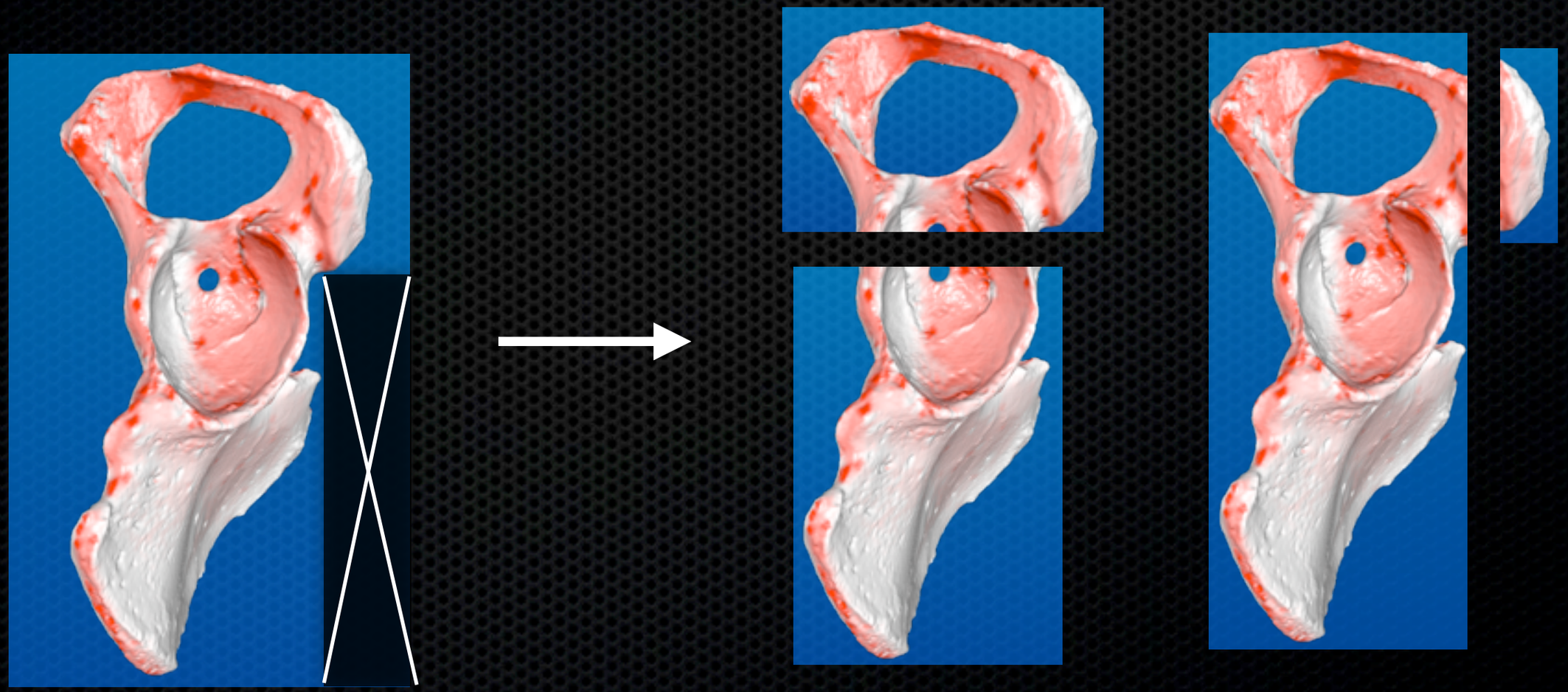
Regions requirements:

- Cover all data
- No overlapping
- Smallest total area

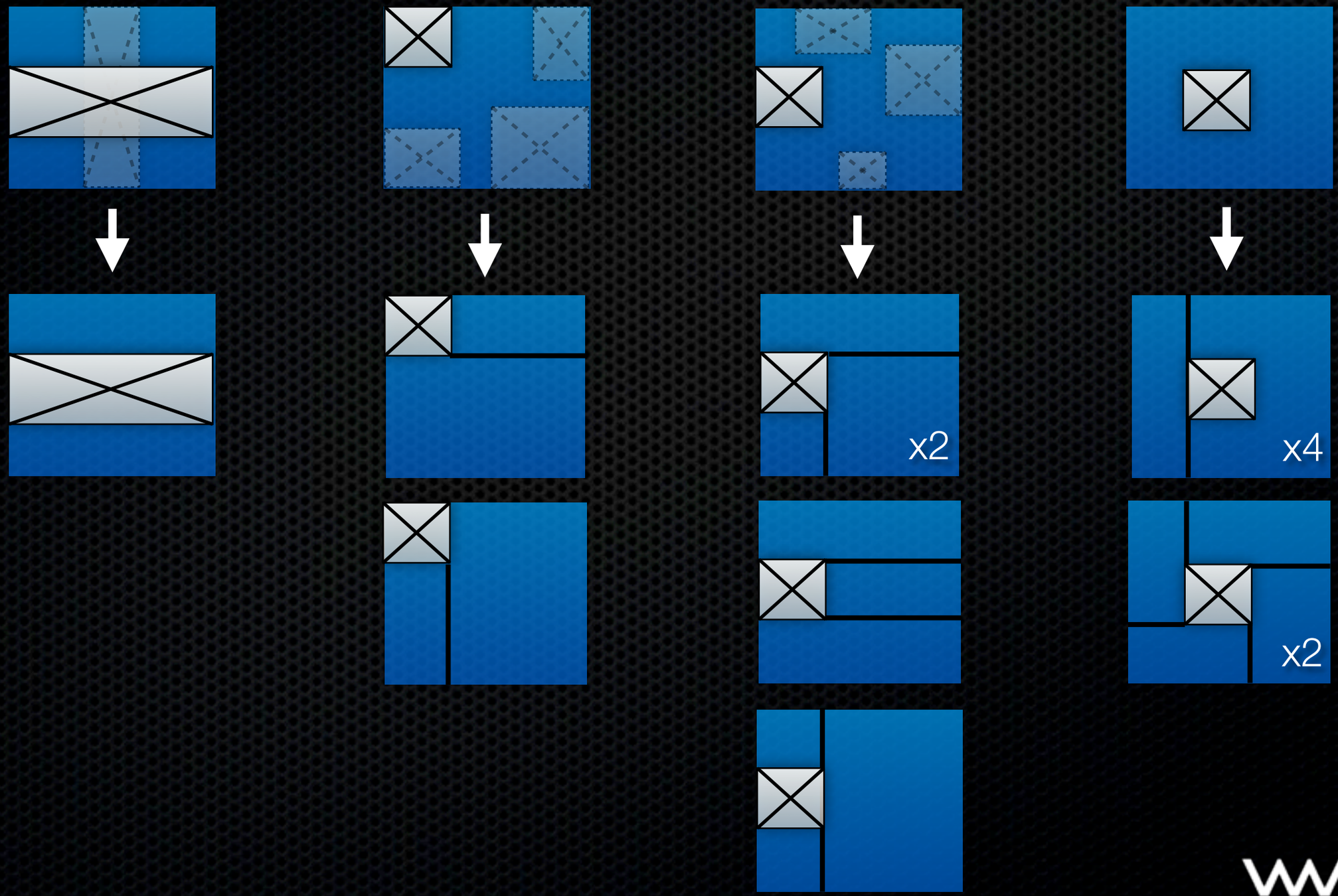


Methods - ROI

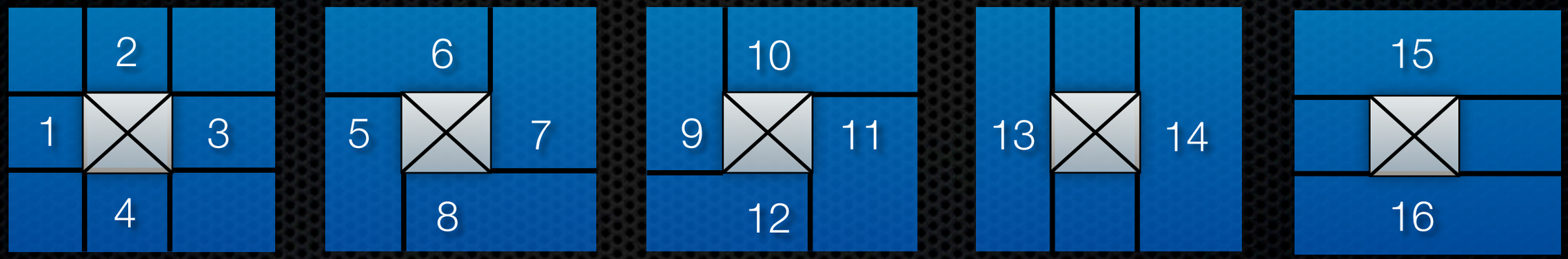
Cut away unused space instead!



Methods - ROI



Methods - ROI



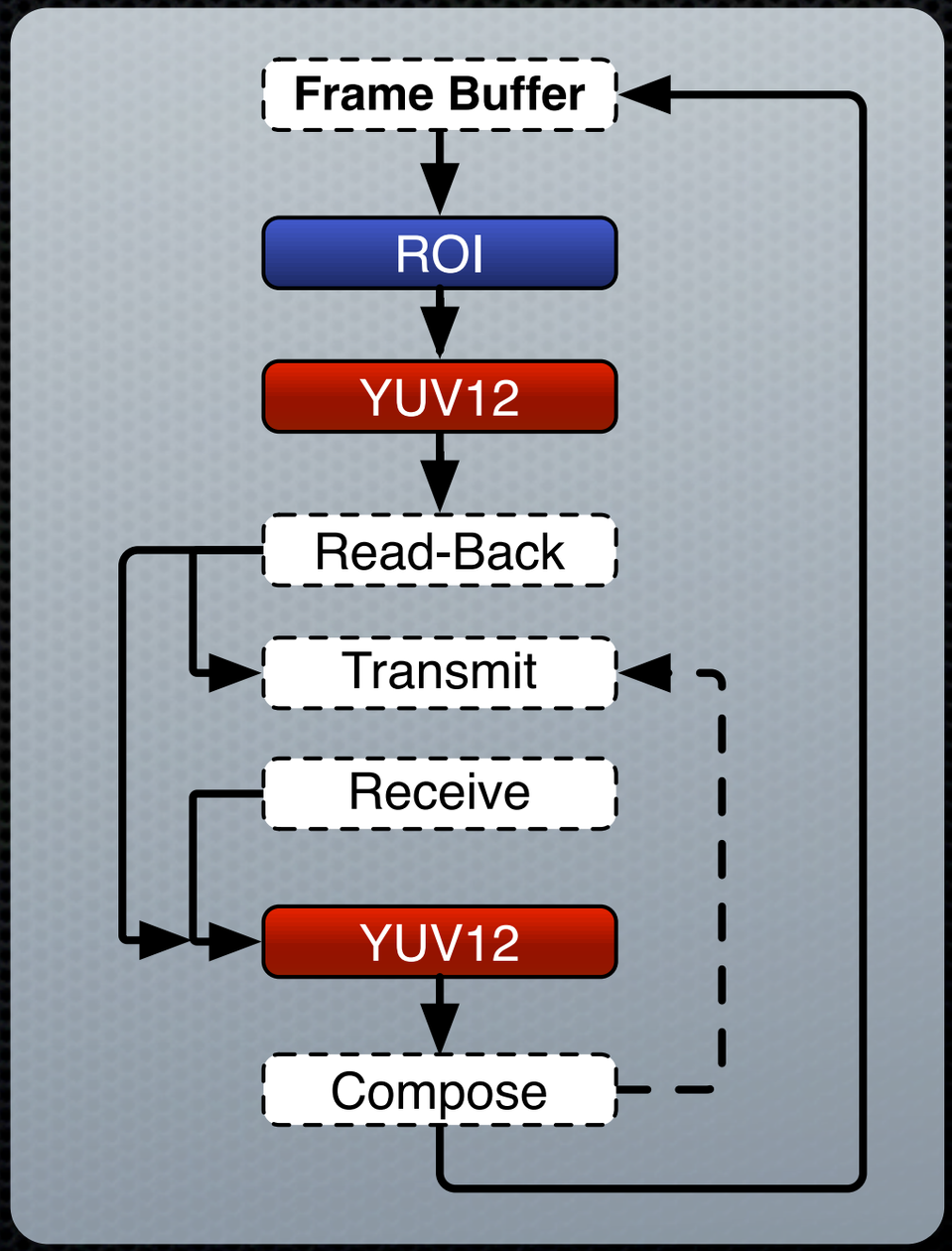
ROI Demonstration

Demo

Methods - Integration

ROI and YUV12 simply integrated as separate layers

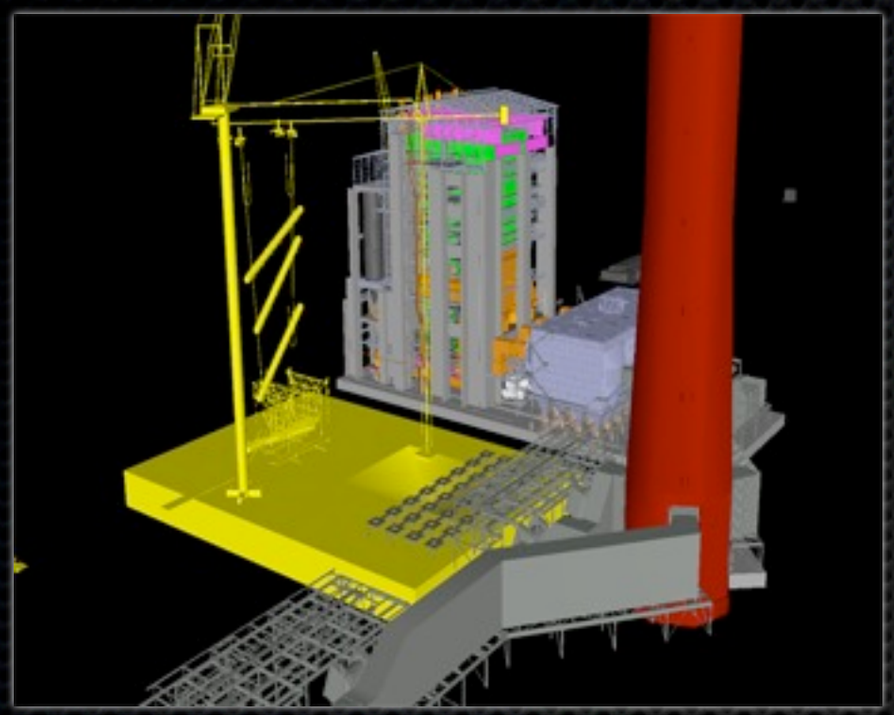
Compatible with other compression methods



Analysis

- Setup
- Throughput and rendering limits
- Sort-first performance
- Sort-last performance
- ROI for sort-last

Models



Power plant

~13 M

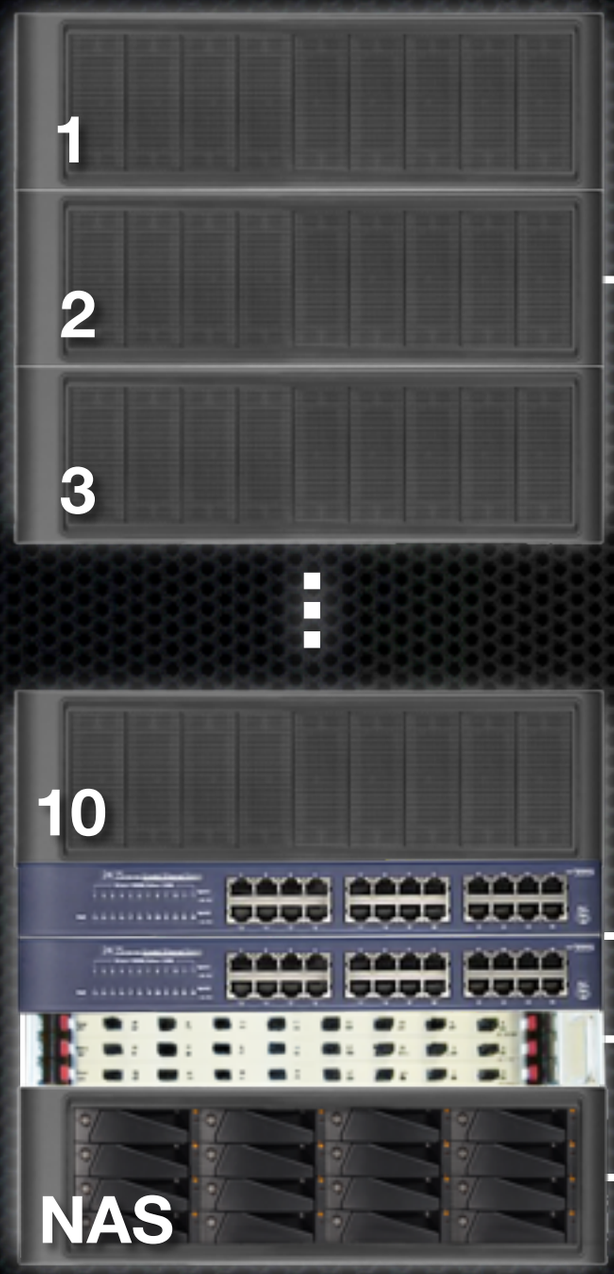
David head

~4 M

David statue

~56 M

Cluster Setup (Hactar)



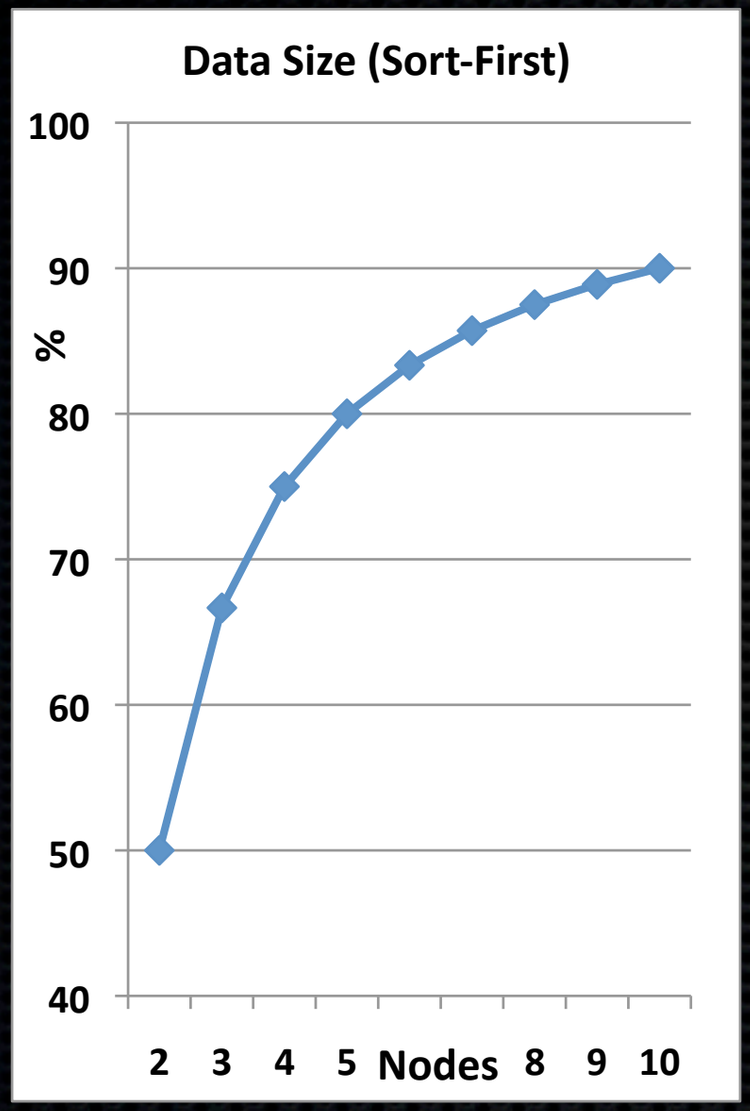
2× 2.2GHz AMD
GeForce 9800 GX2
4 GB RAM

1 Gbit/s (Ethernet)
2 Gbit/s (Myrinet)

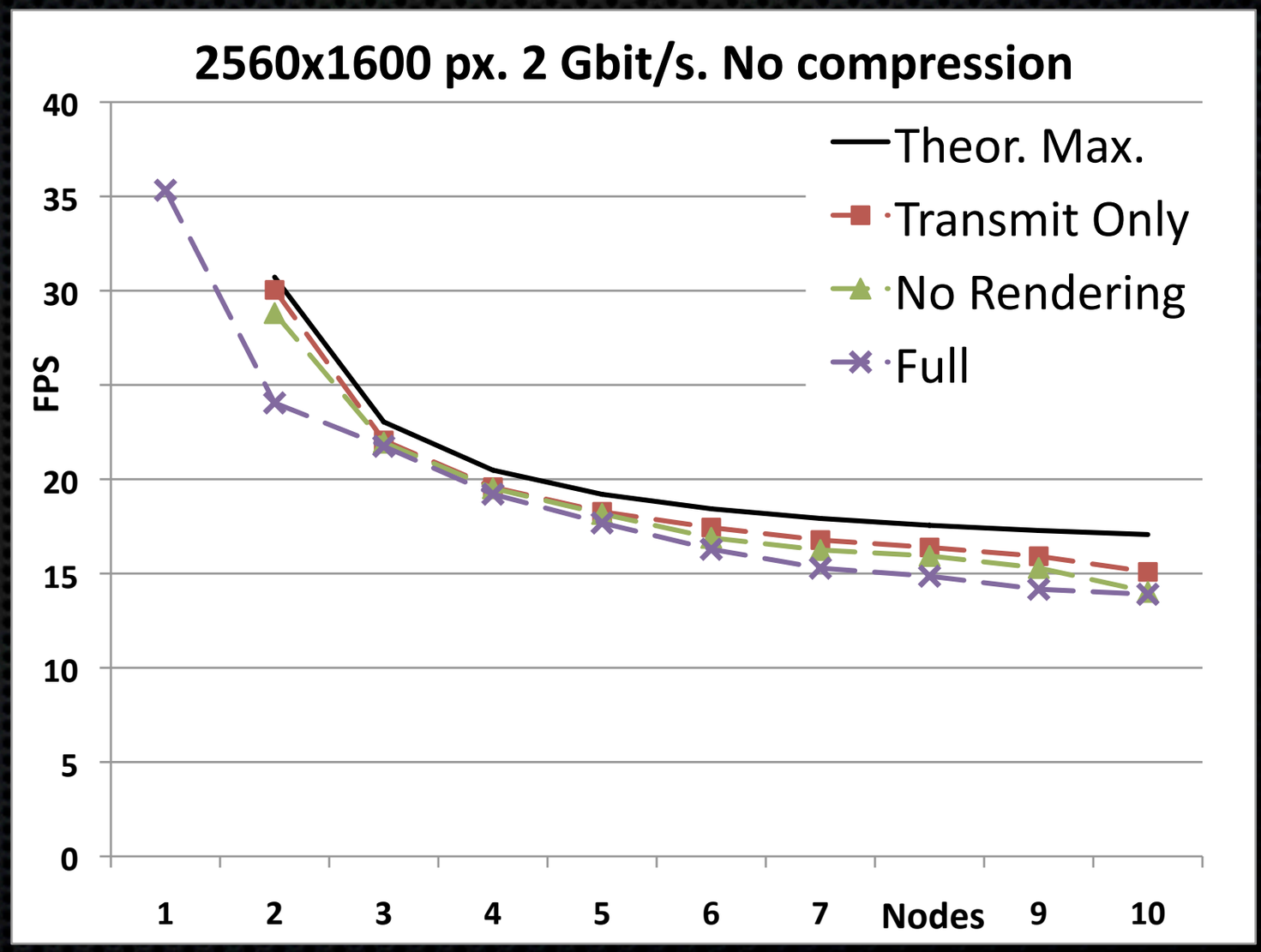
2 TB (4× 1Gbit/s Ethernet)

VISUALIZATION AND MULTIMEDIA LAB

Throughput and Rendering Limits (No Compression)

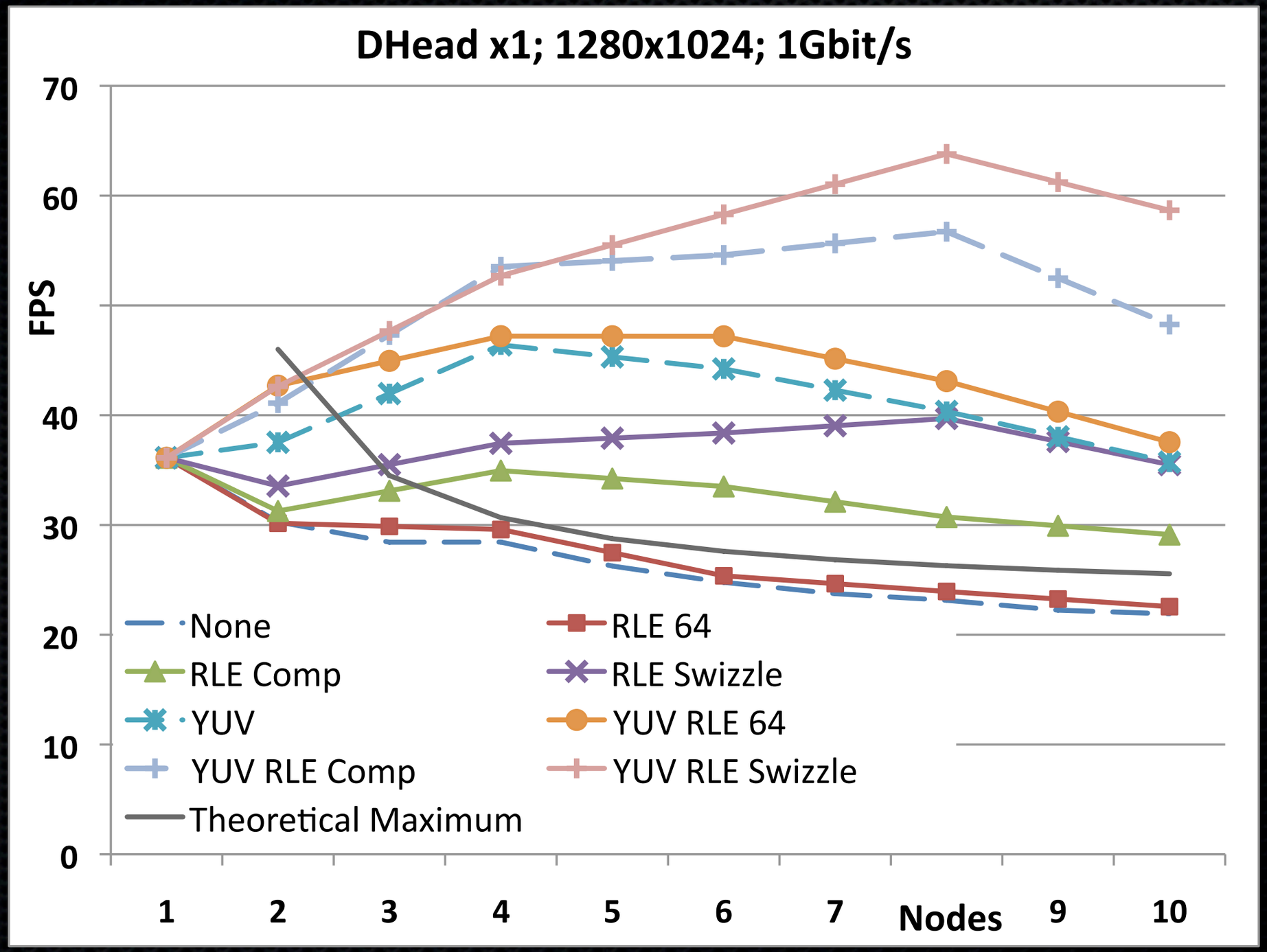


Data to receive

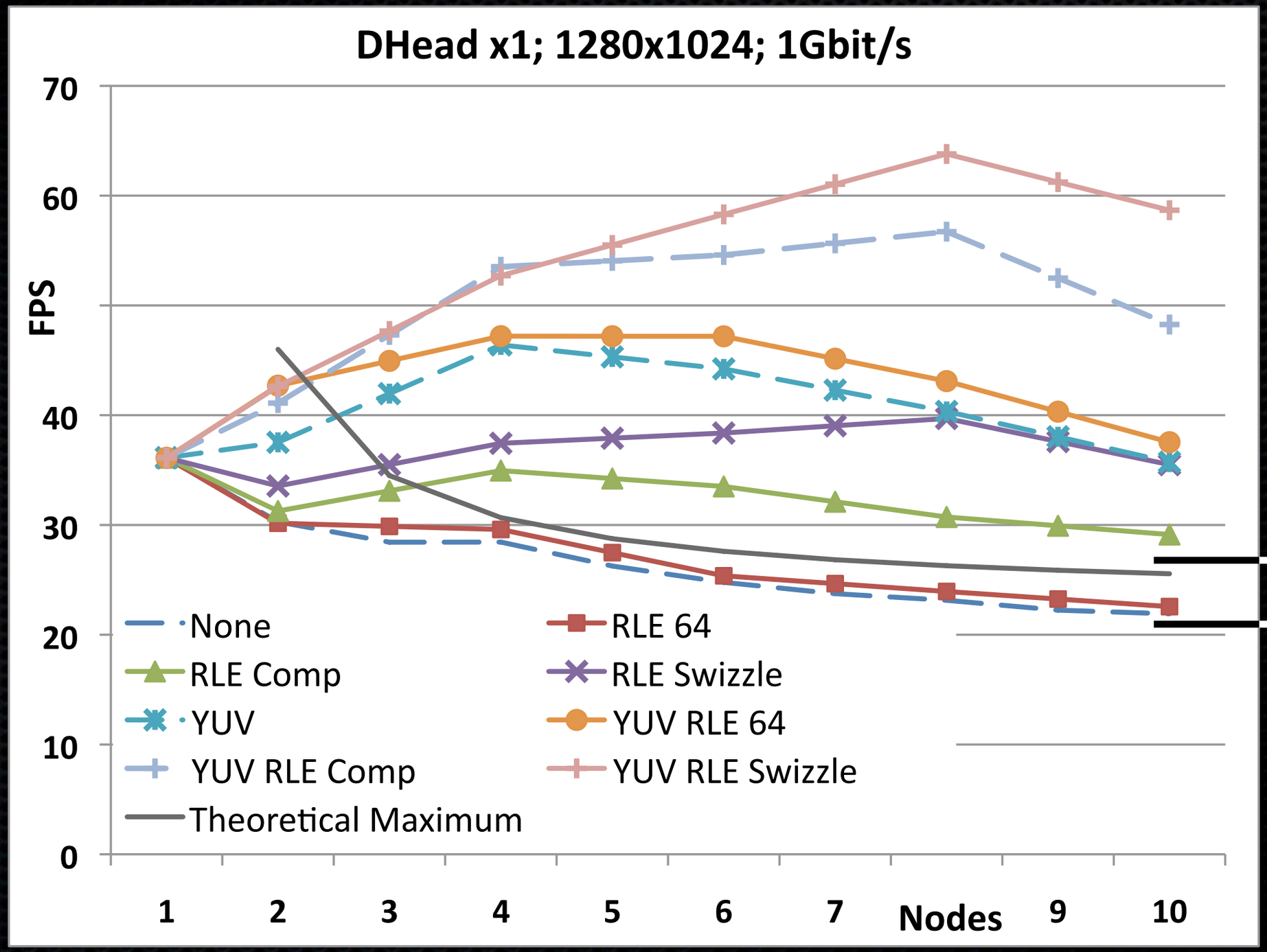


Performance of sort-first

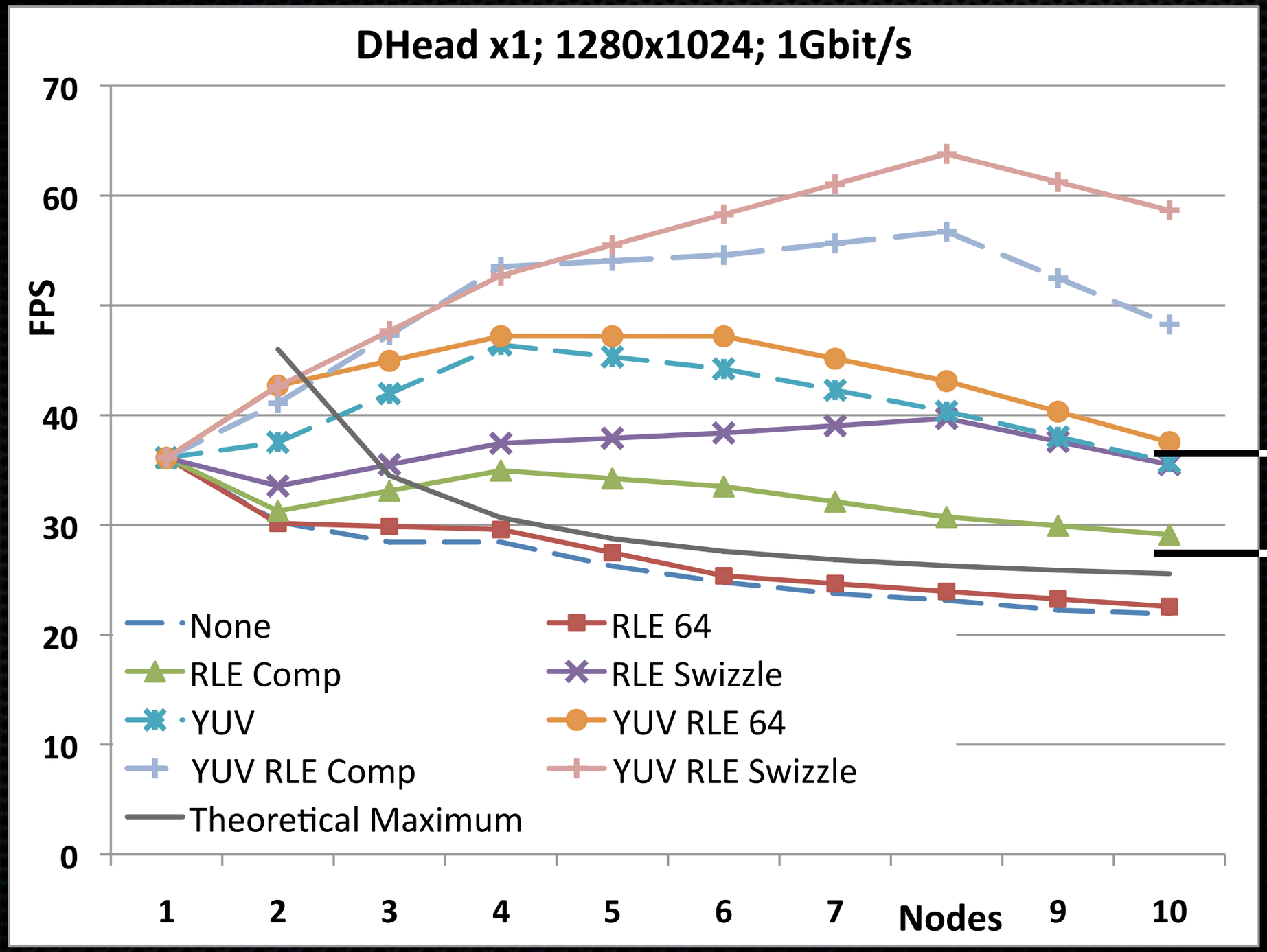
Sort-First Performance



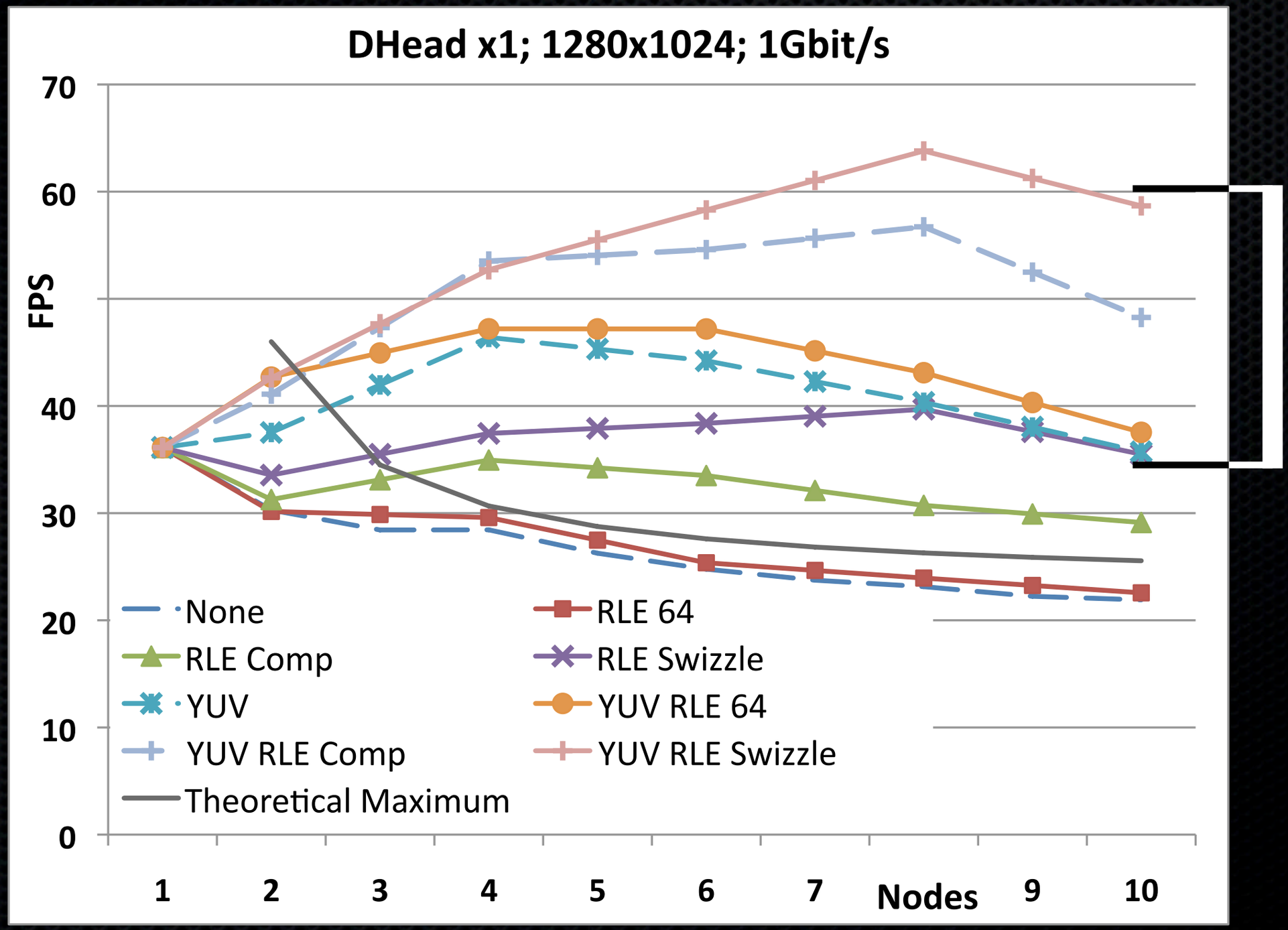
Sort-First Performance



Sort-First Performance

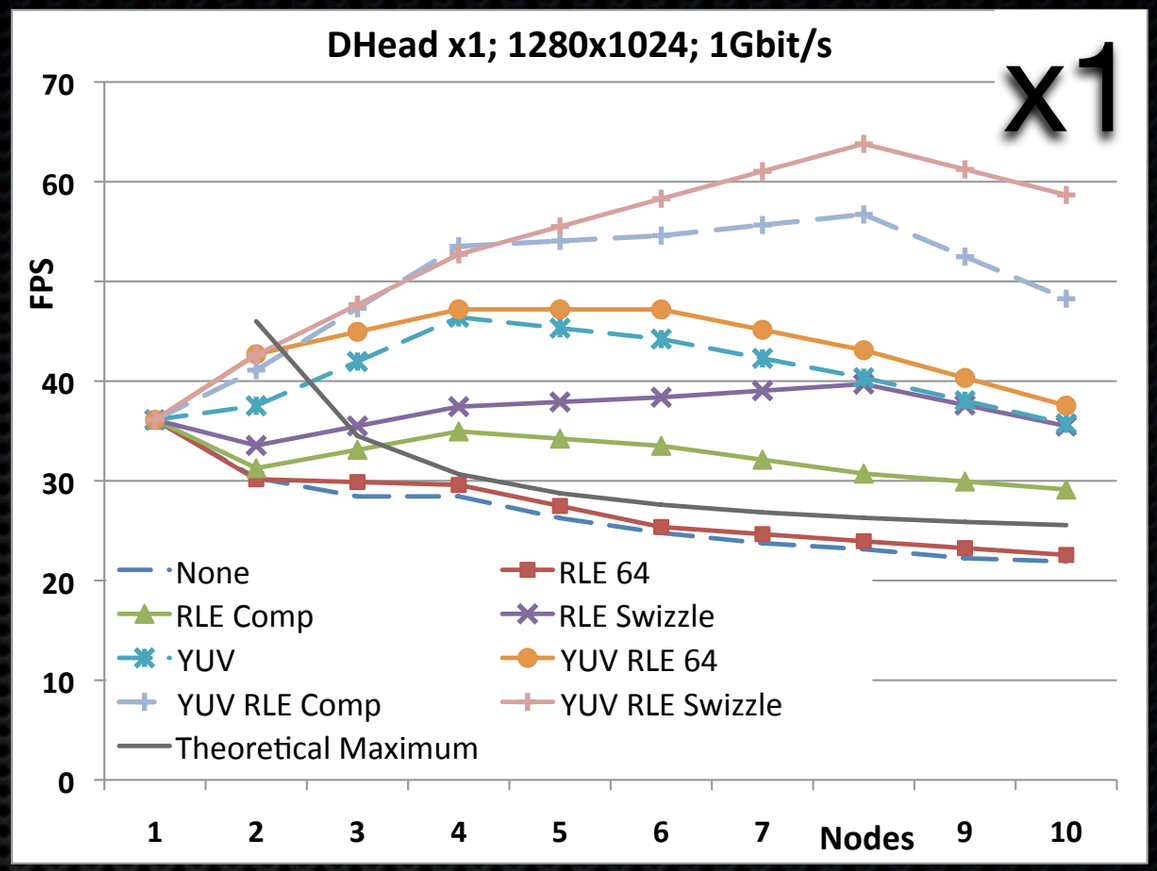


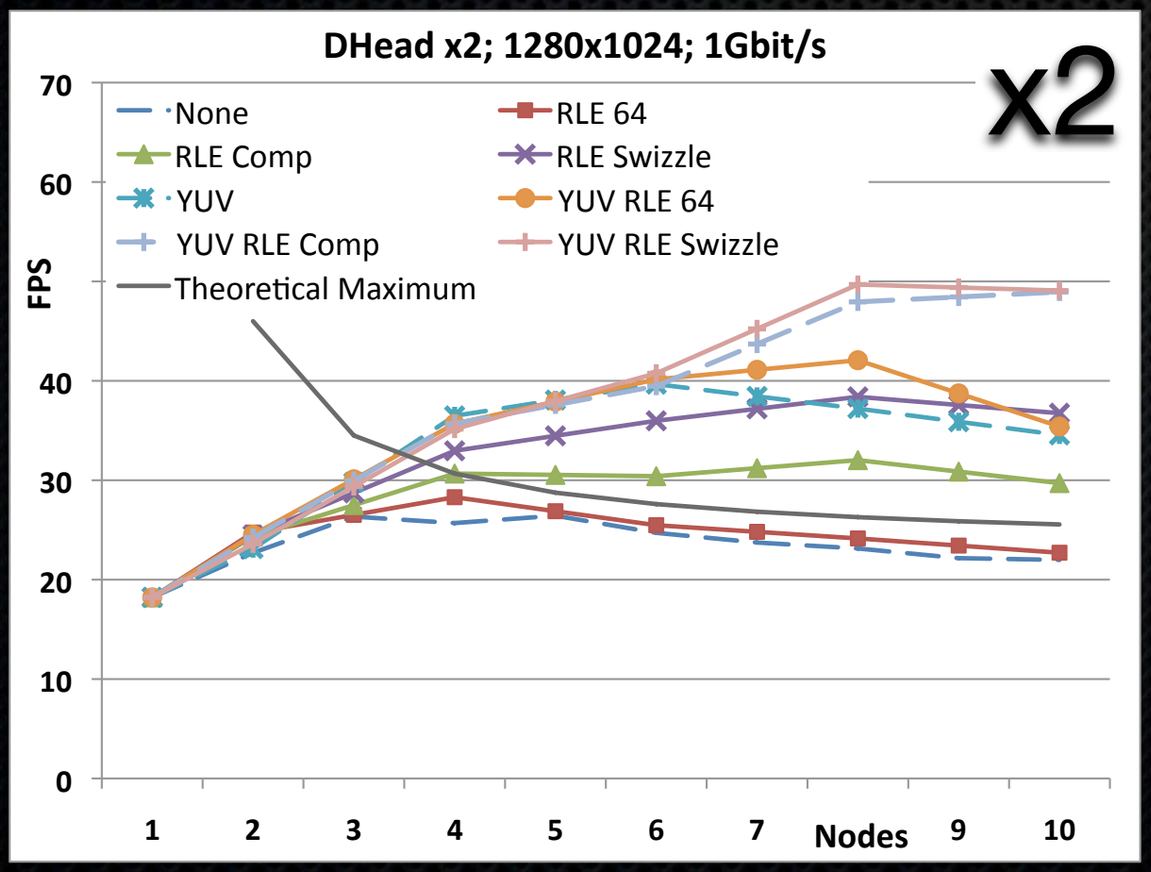
Sort-First Performance

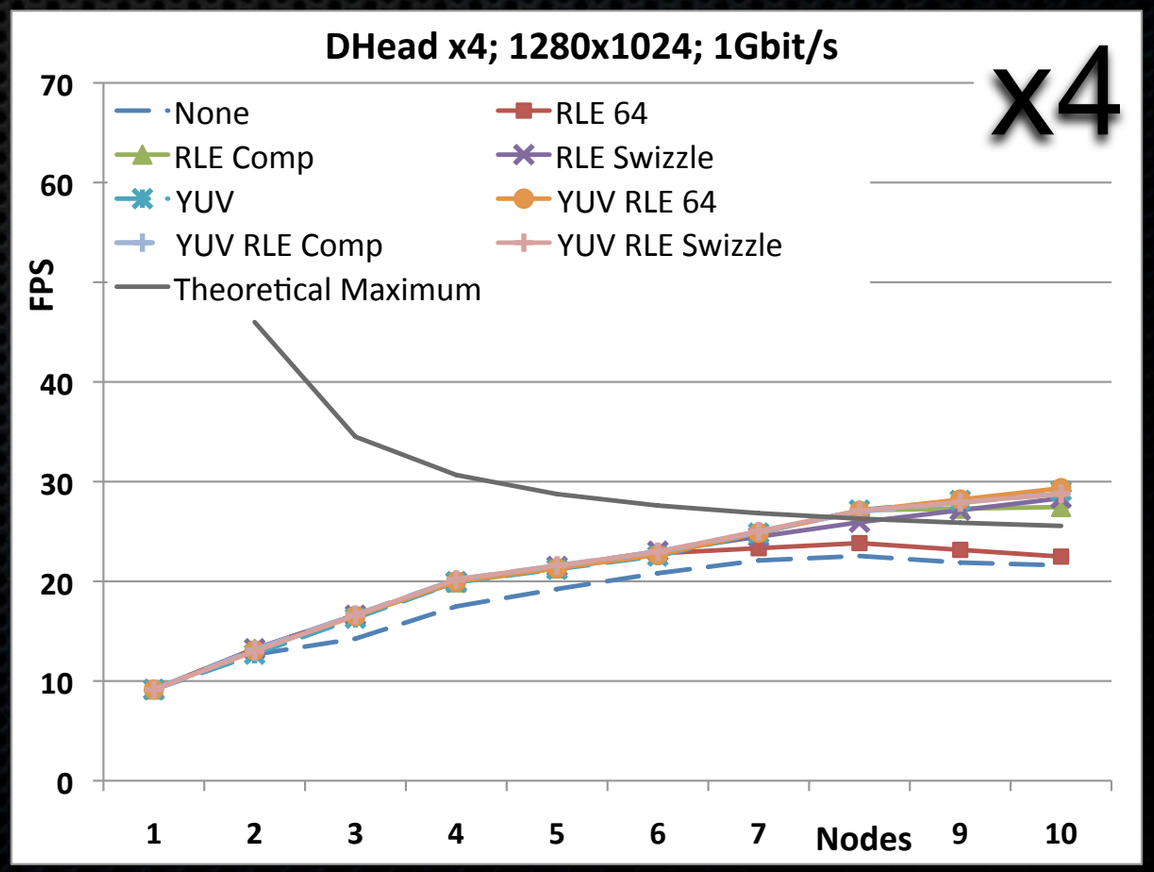


Sort-First Performance

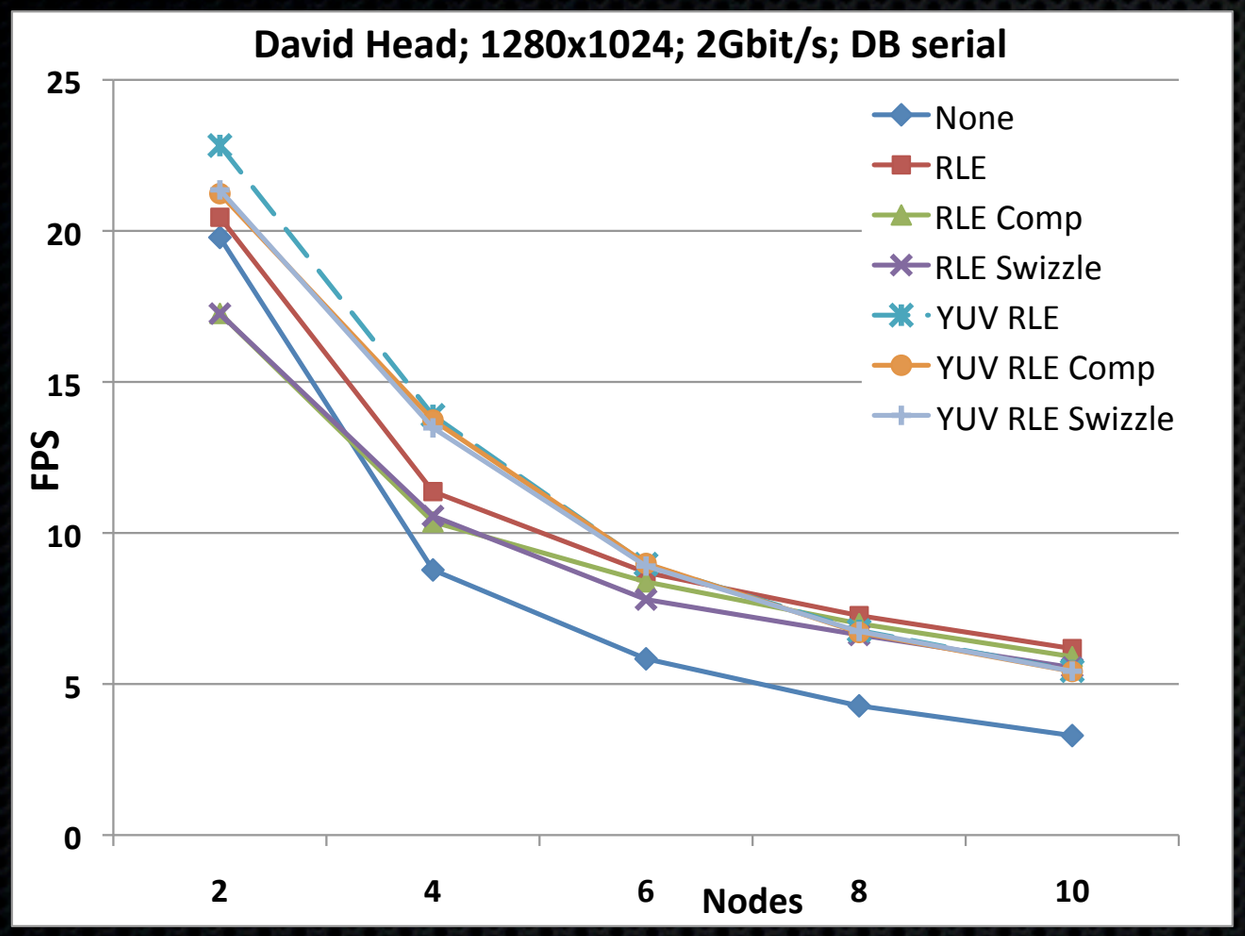
David head model
rendered 1, 2 and 4 times



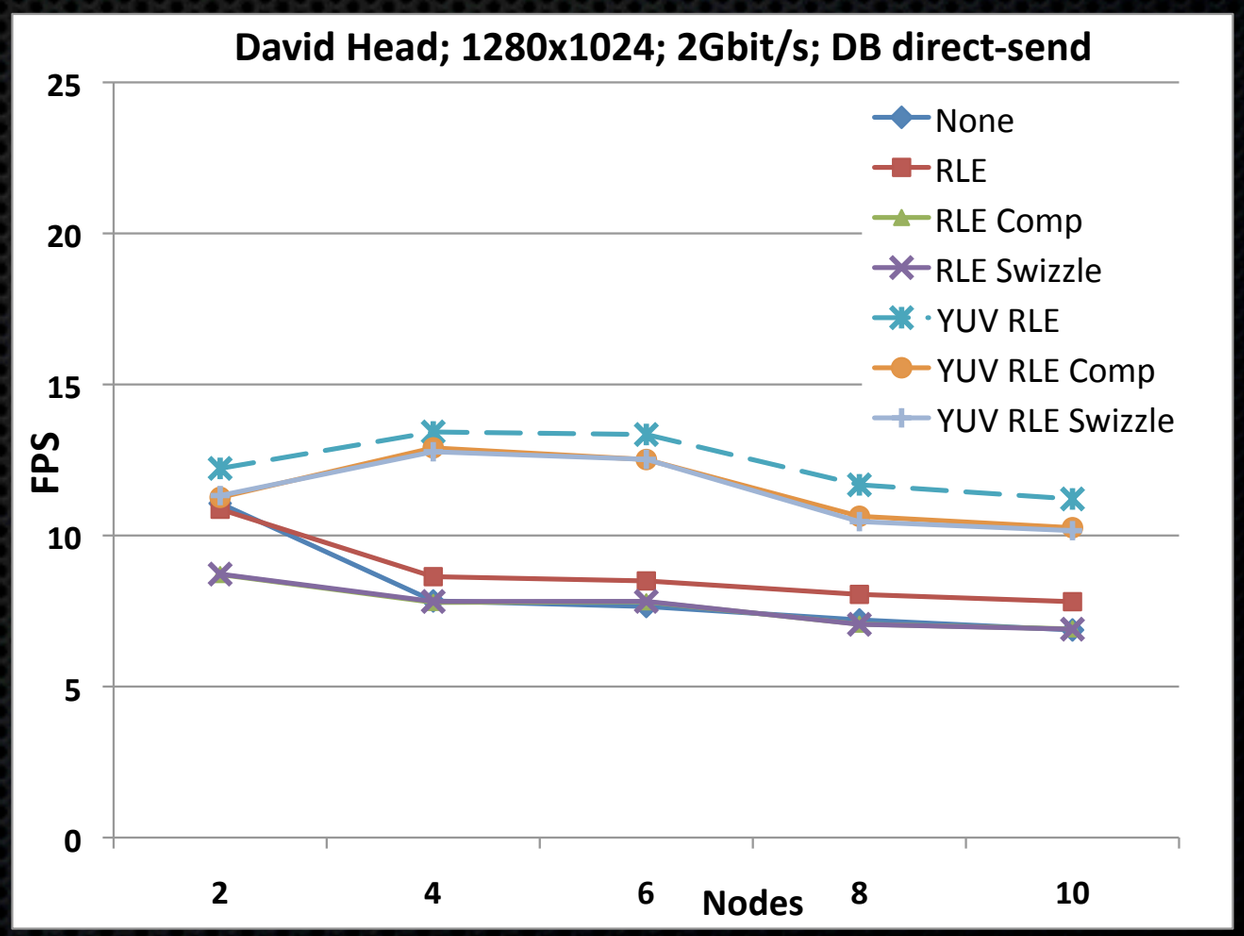




Sort-Last Performance



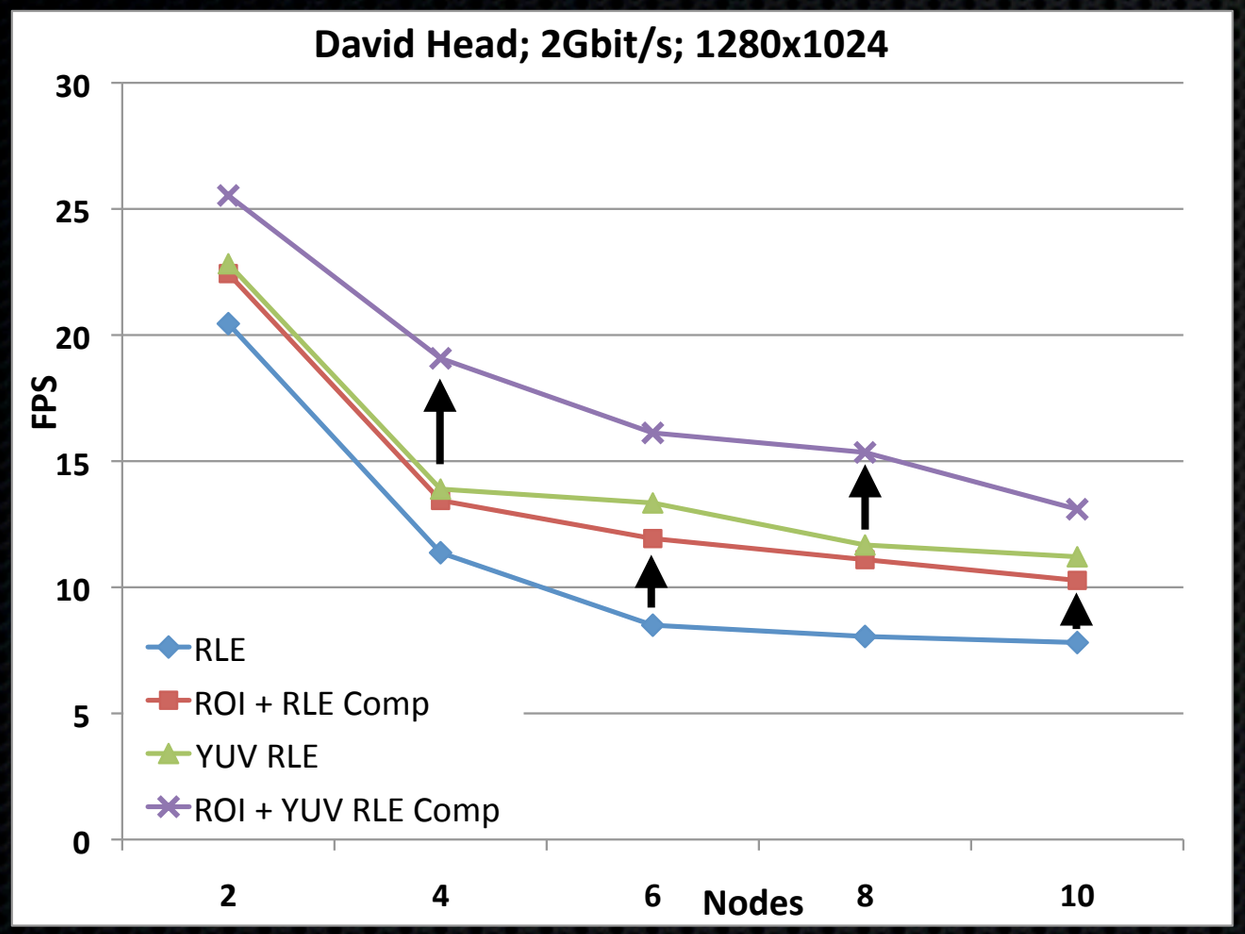
Serial Sort-Last



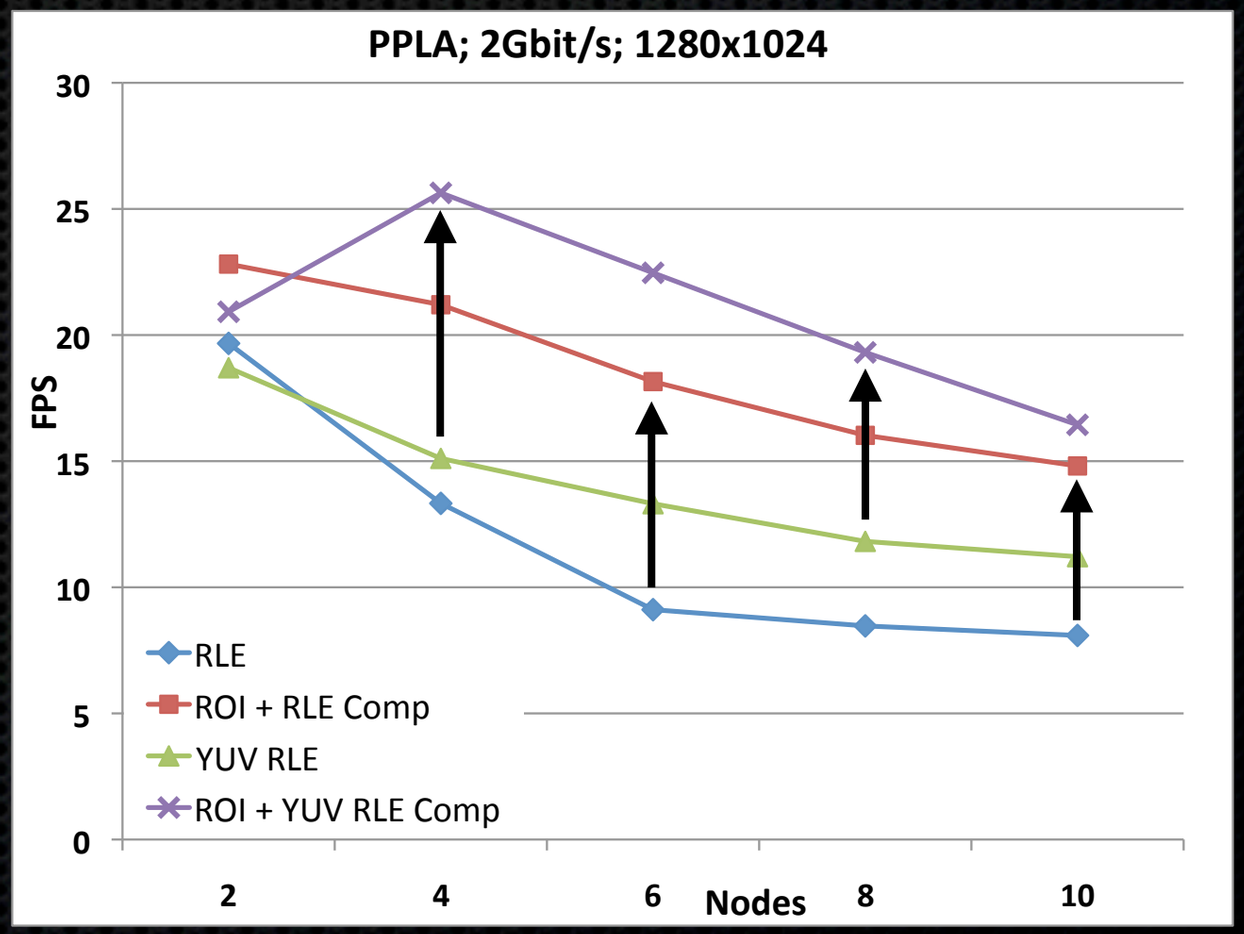
Direct-Send

VISUALIZATION AND MULTIMEDIA LAB

ROI Performance



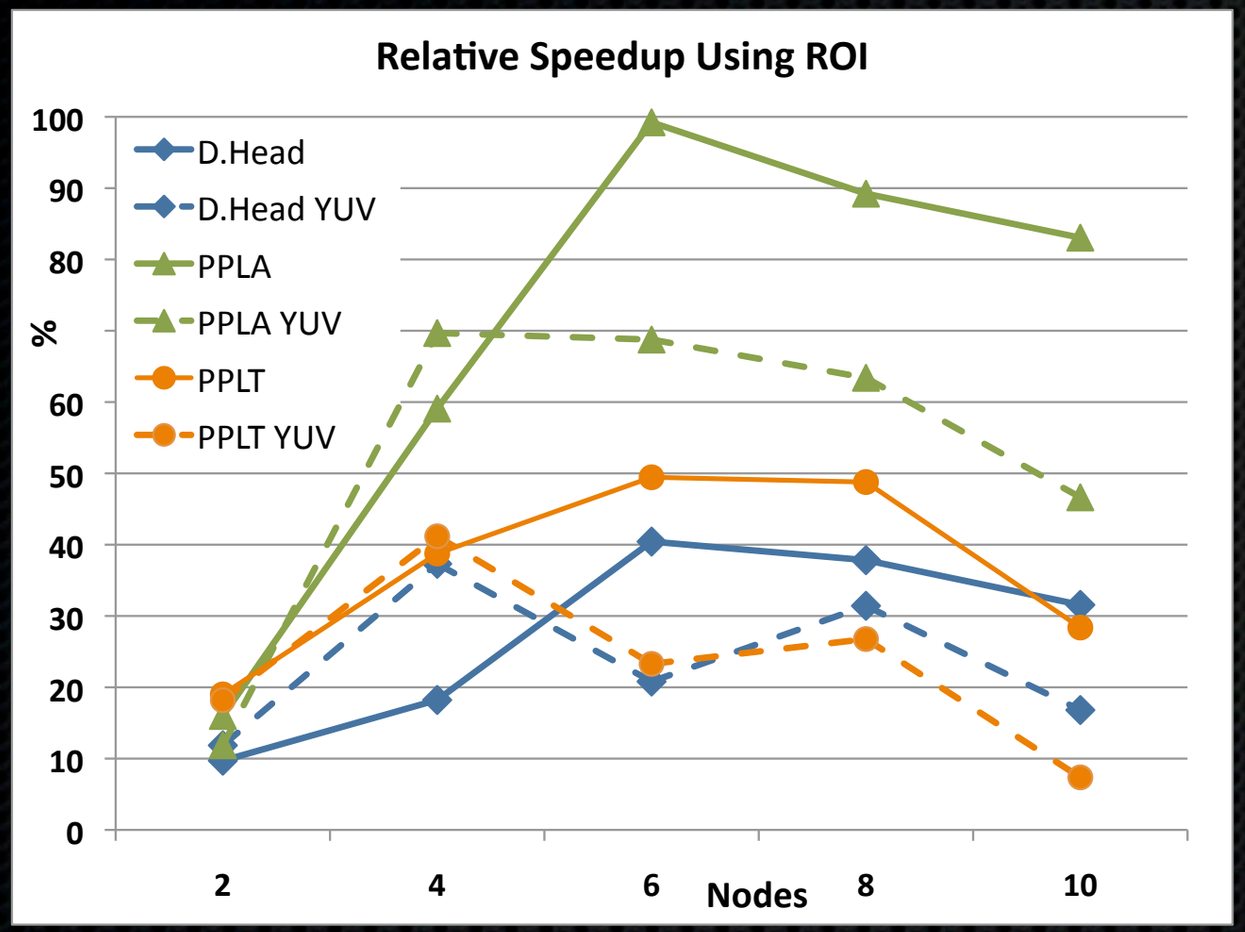
David head



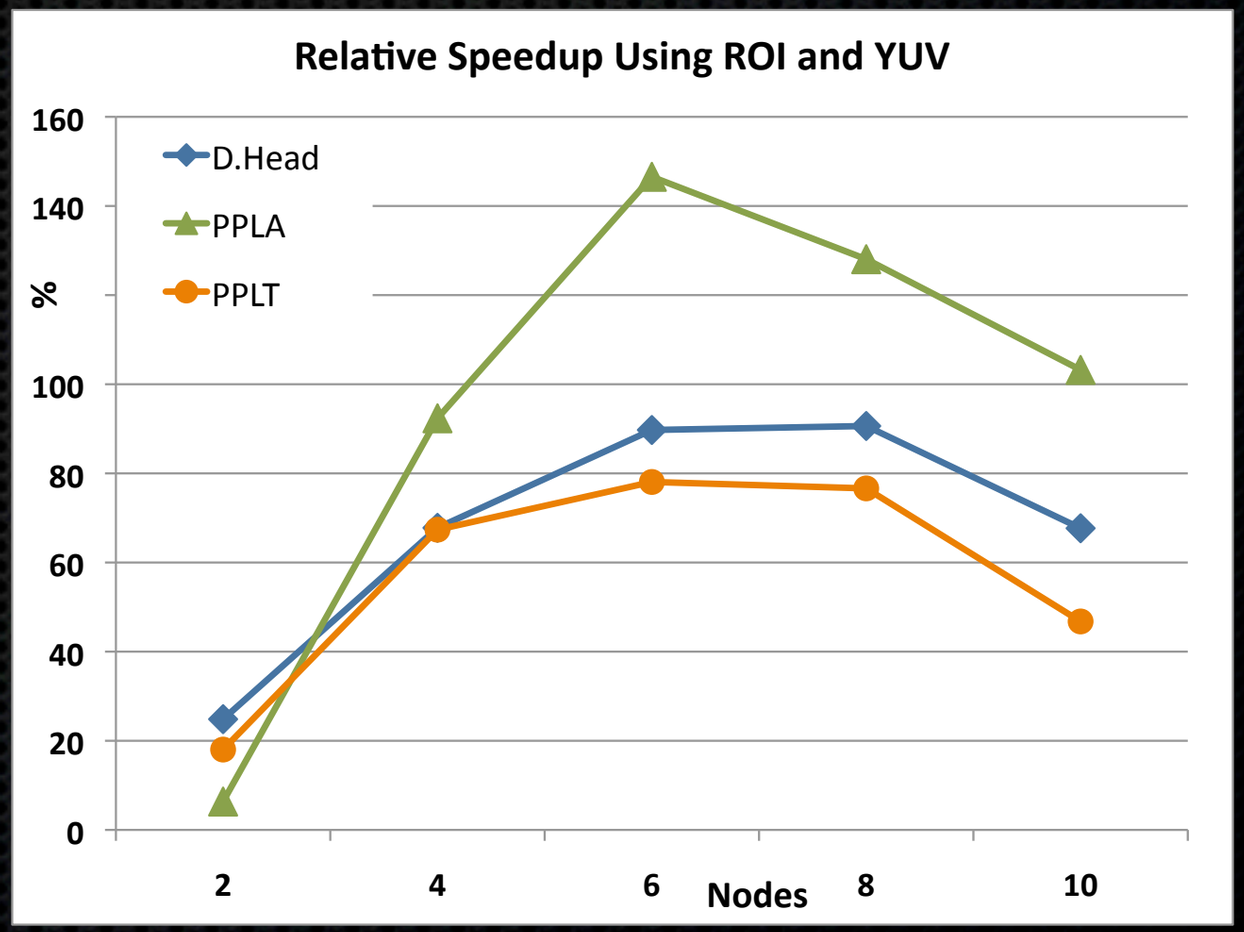
PPLA

VISUALIZATION AND MULTIMEDIA LAB

Relative ROI Performance



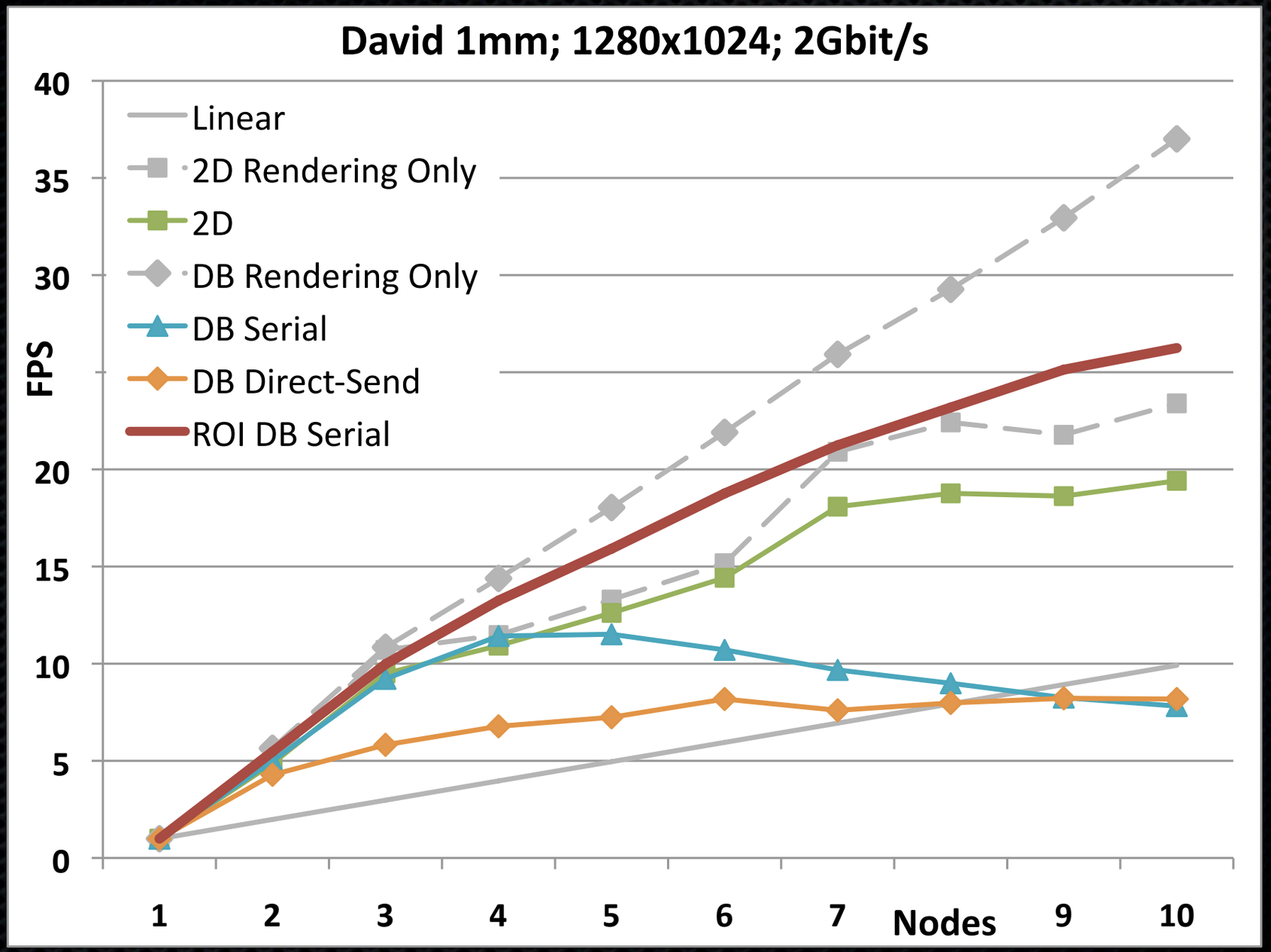
Relative ROI



Relative ROI+YUV

VISUALIZATION AND MULTIMEDIA LAB

ROI Best Case



David statue (56M triangles)

Conclusion

- Several compression techniques are compared in terms of overall rendering performance
- Fast ROI algorithm for sort-last rendering is demonstrated

Thank You!

Questions?

VISUALIZATION AND MULTIMEDIA LAB