Direct Send Compositing for Parallel Sort-Last Rendering

Stefan Eilemann, Renato Pajarola
Visualization and MultiMedia Lab
University of Zurich

20. May 2007

Latest version at http://www.equalizergraphics.com/documents/EGPGV07.pdf

VMML
visualization and multimedia lab
university of zürich

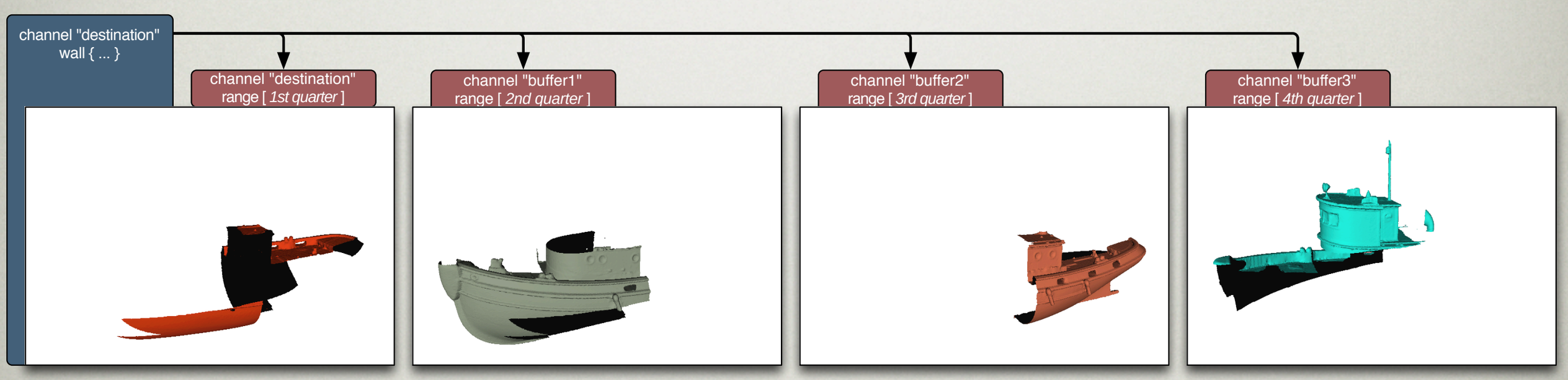
Outline

- Parallel Sort-Last Rendering
- Compositing
- Direct-Send Compositing
- Binary-Swap Compositing
- Implementation
- Results

VMML
visualization and multimedia lab
university of zürich

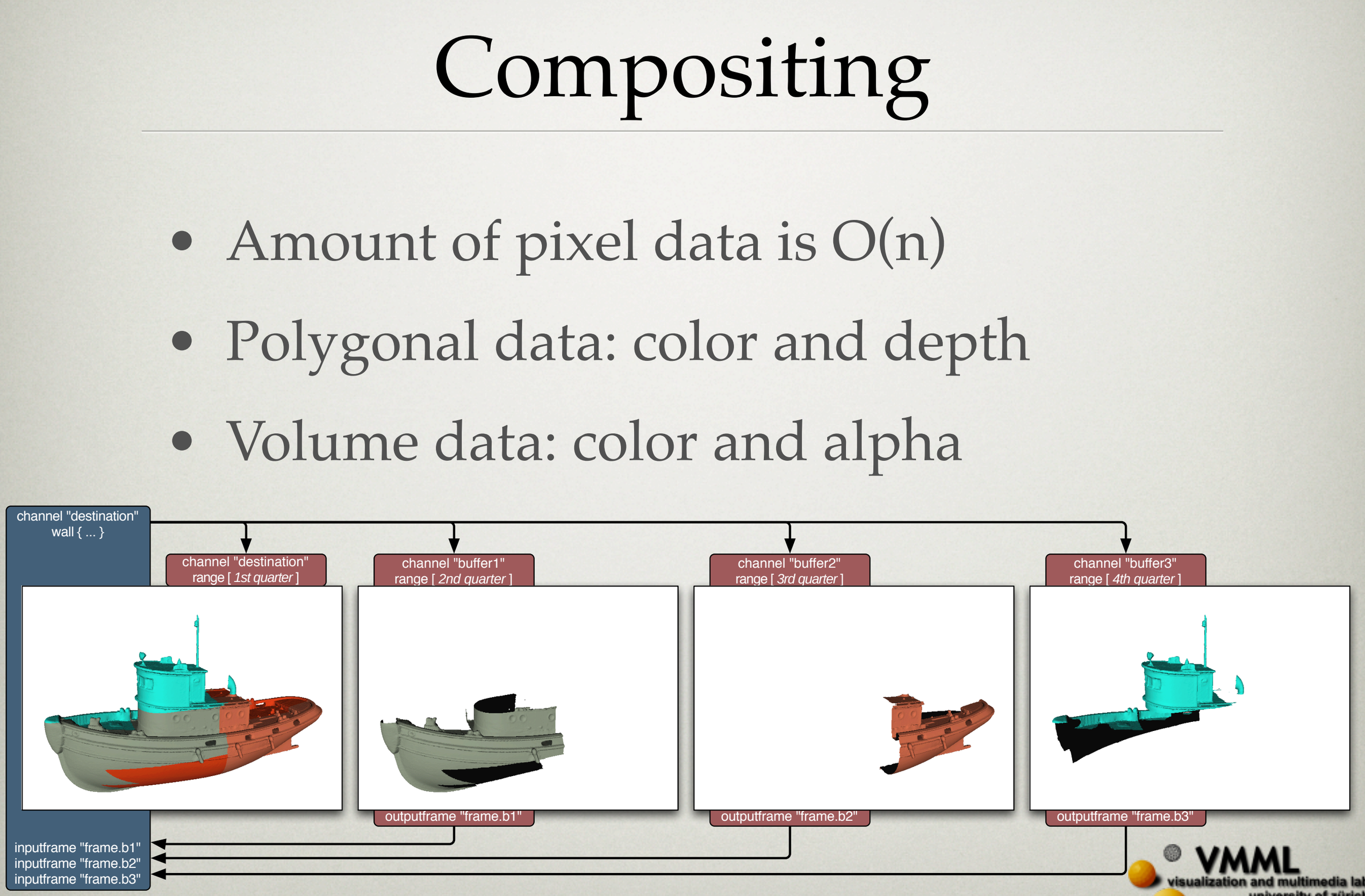
Parallel Sort-Last Rendering

- Run n rendering threads
- Each thread renders 1/n of the data
- Rendering performance scales nicely



Compositing

- Amount of pixel data is O(n)
- Polygonal data: color and depth
- Volume data: color and alpha

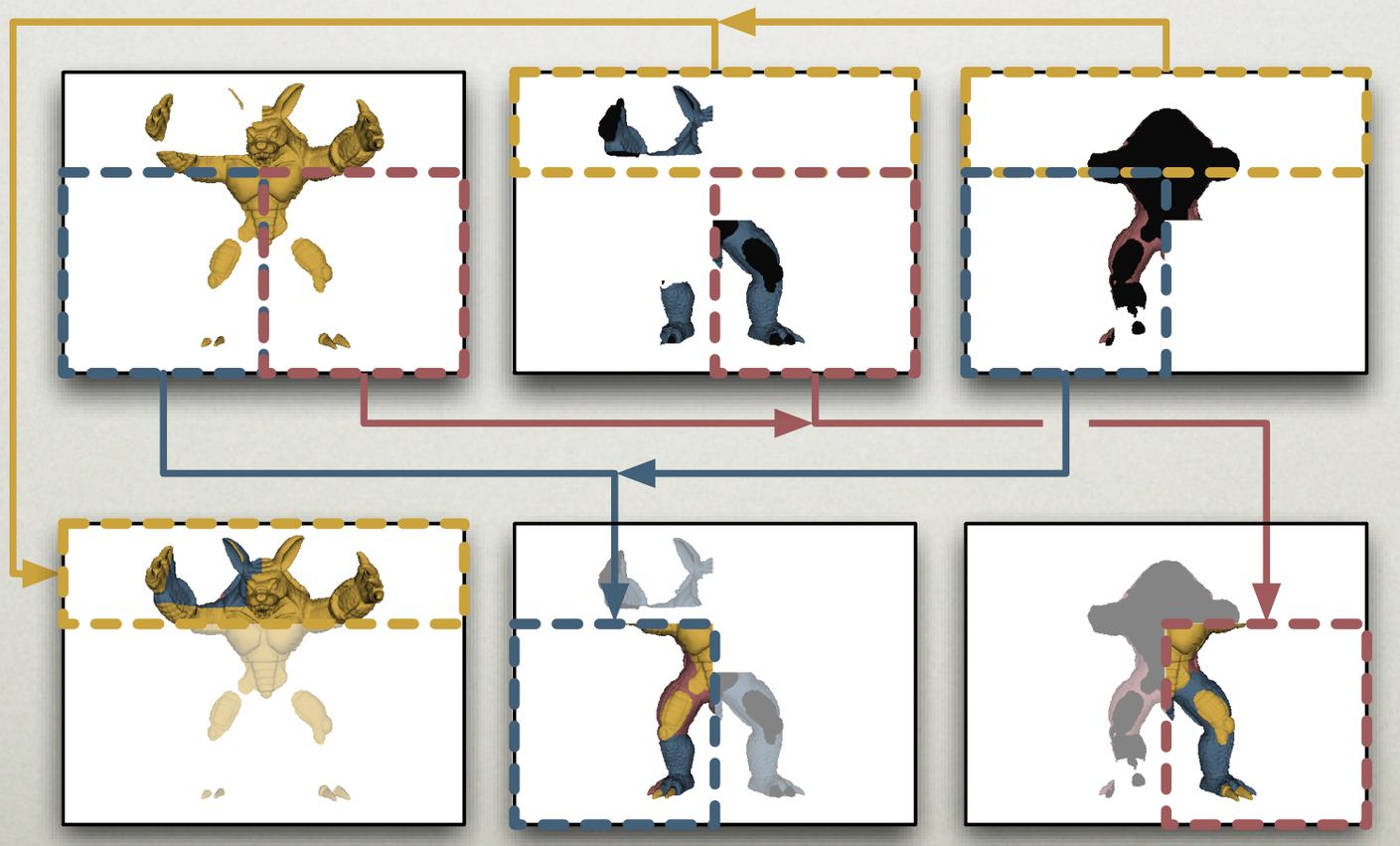


VMML
visualization and multimedia lab
university of zürich

Direct-Send Compositing

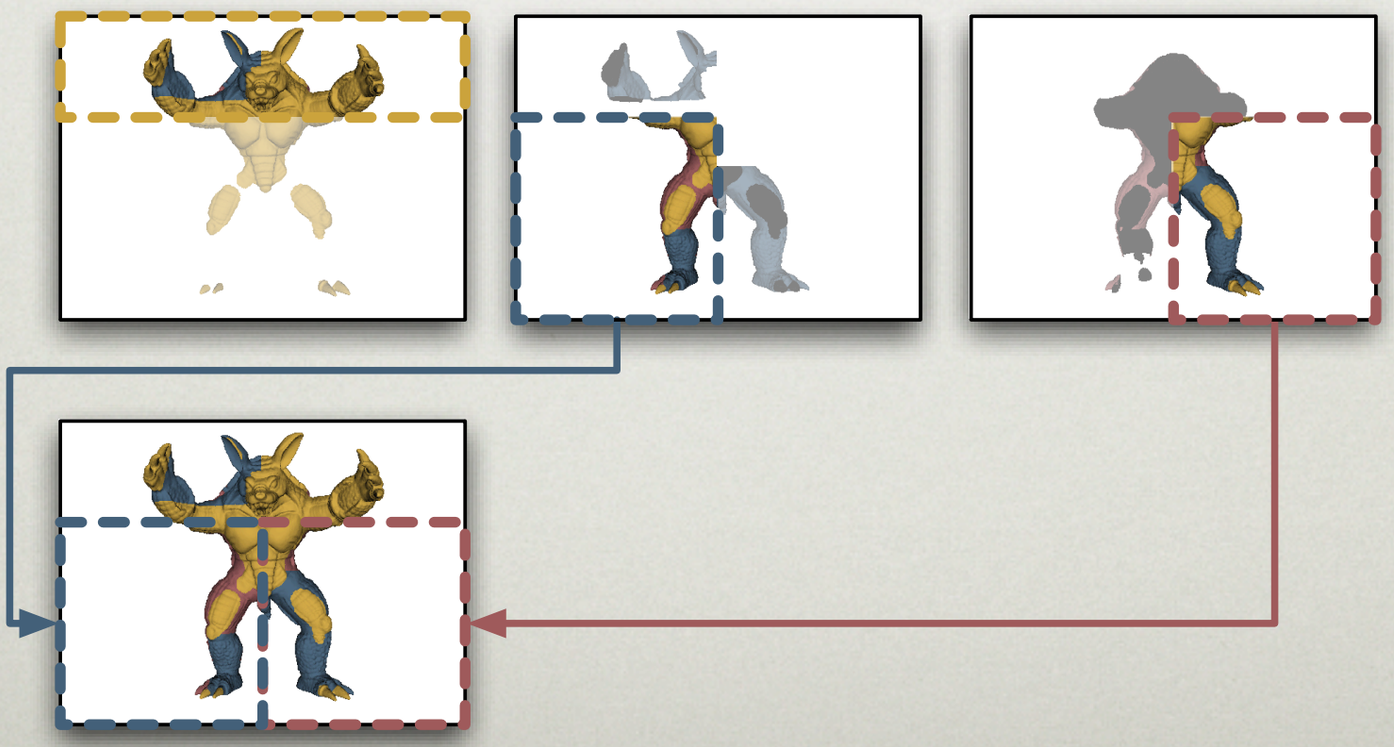
- Use all nodes for compositing
- Amount of data per node is O(1)

1. Each node composites one tile



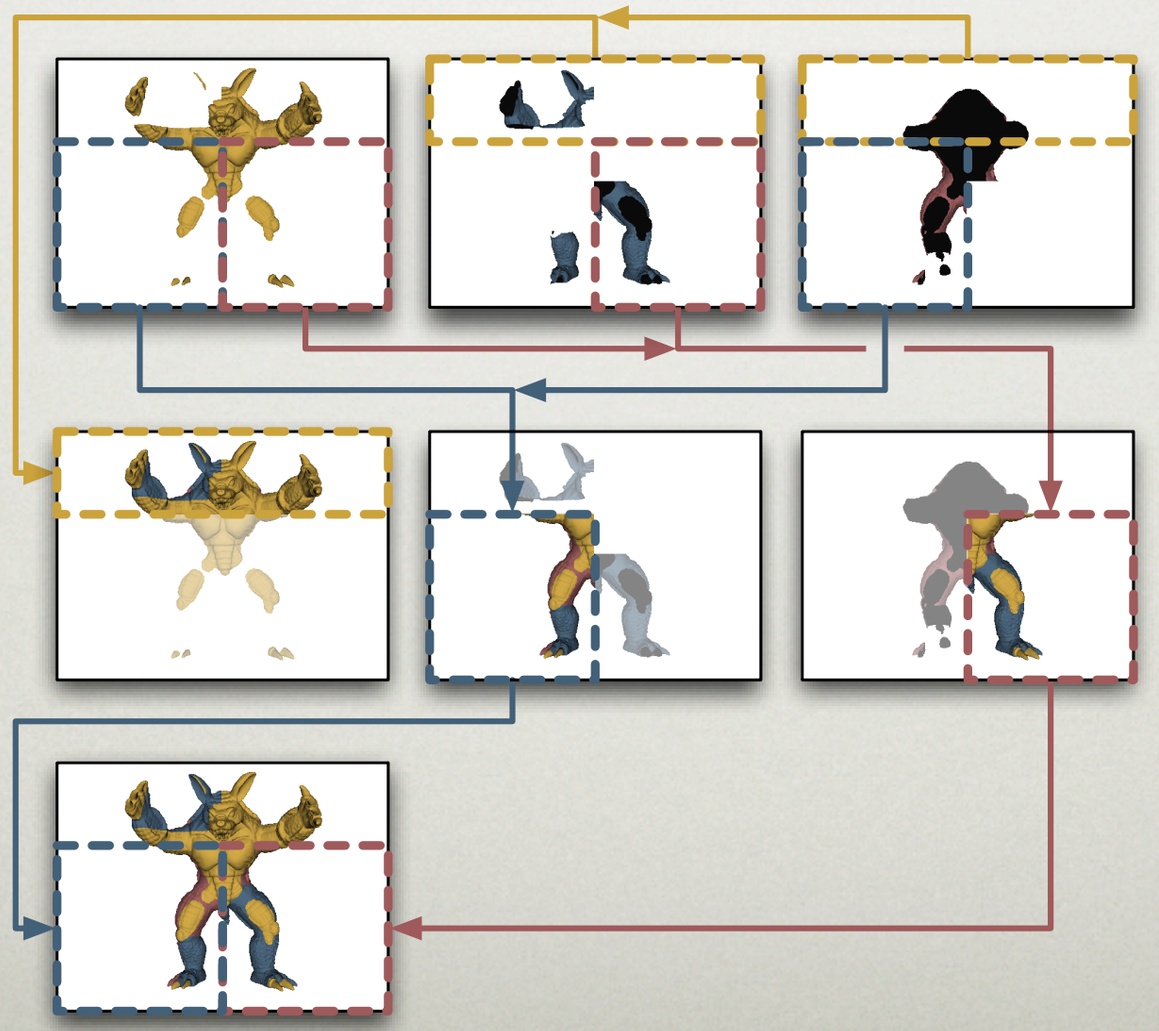
Direct-Send Compositing (2)

2. Gather composited tiles
   - Color information only - like sort-first
   - Destination channel's tile is in-place



Direct-Send Compositing (3)

- Any number of nodes
- Two synchronization points



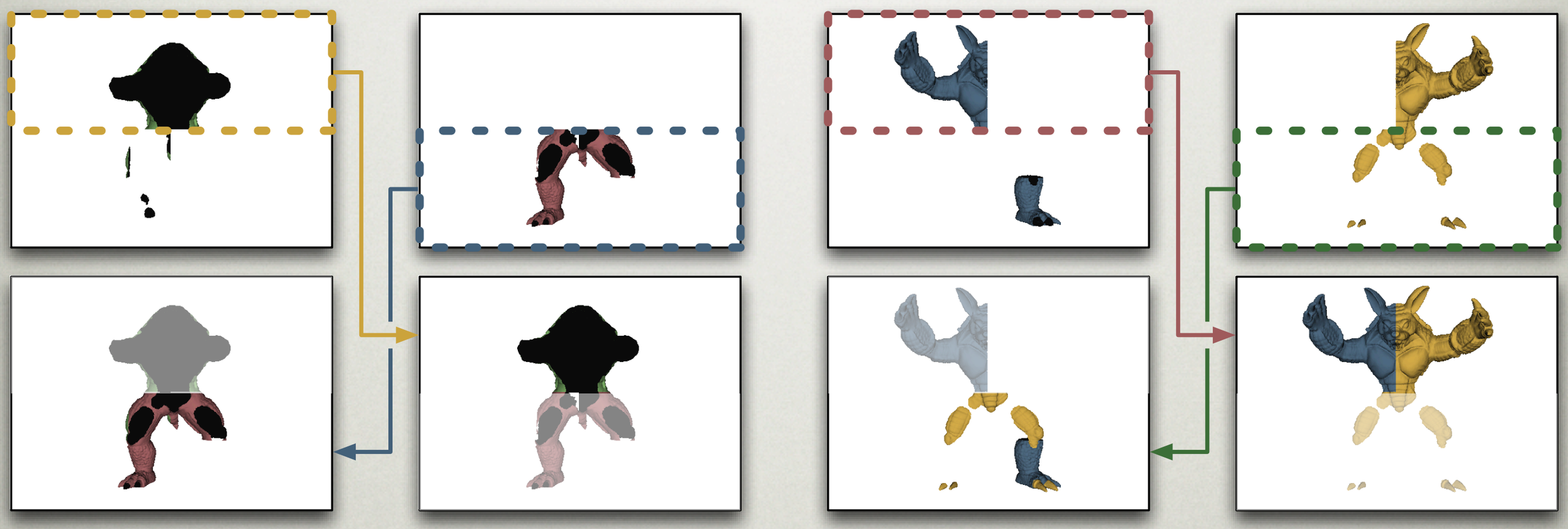
Direct-Send Compositing

Demo

VMML
visualization and multimedia lab
university of zürich

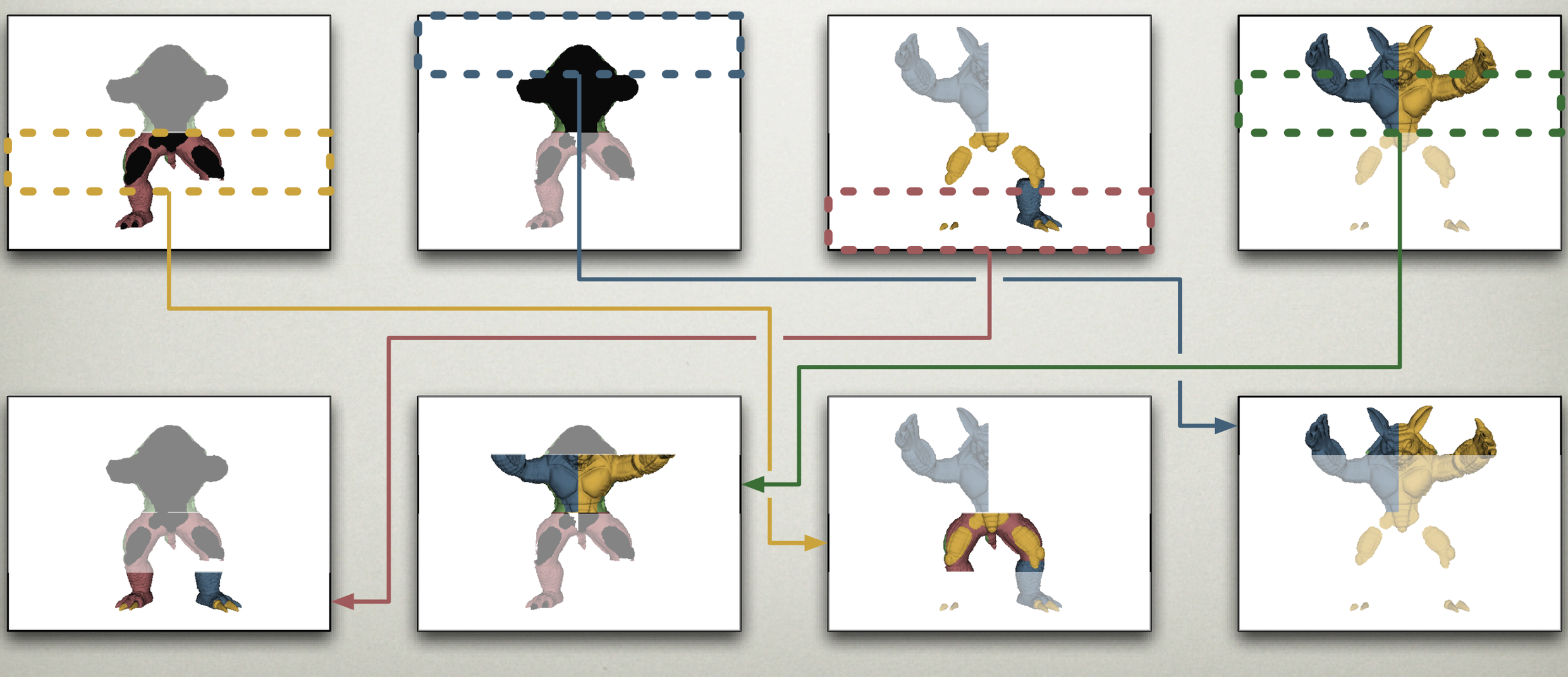
Binary-Swap Compositing

1. Swap half of framebuffer with partner
2. Composite



Binary-Swap Compositing (2)

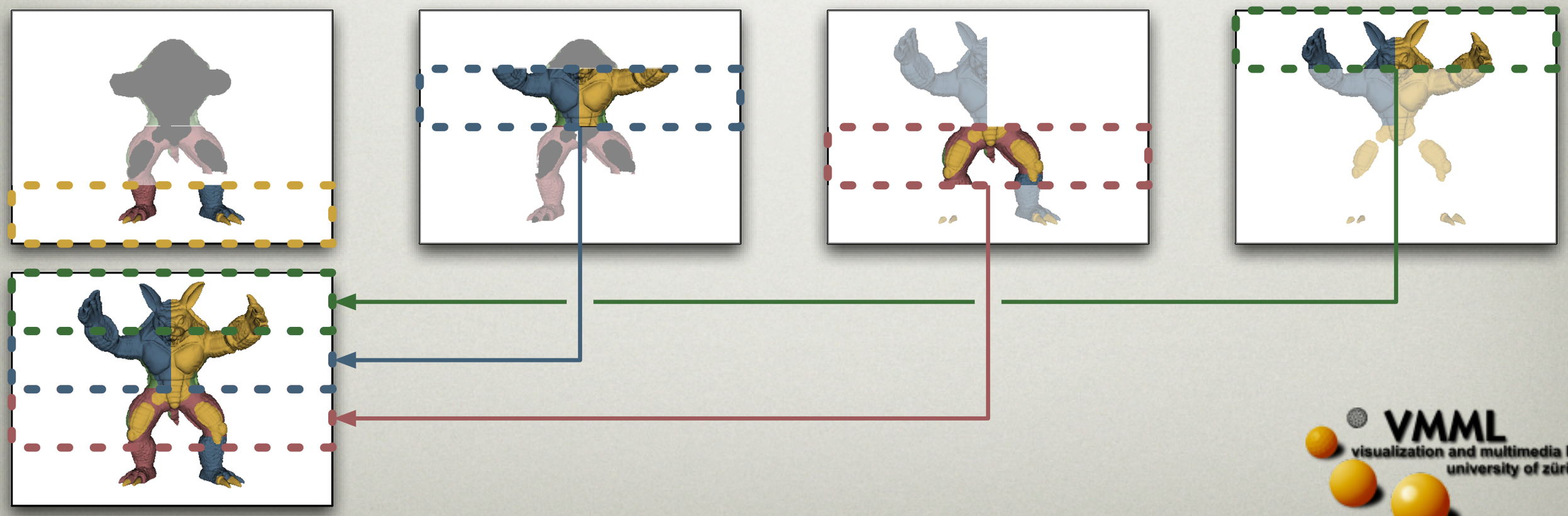
3. Swap half of ‘half’ with another partner
4. Repeat until tiles are complete



Binary-Swap Compositing (3)

5. Gather color tiles

- Power-of-two number of nodes
- Log2(n) synchronization points



Binary-Swap Compositing

Demo

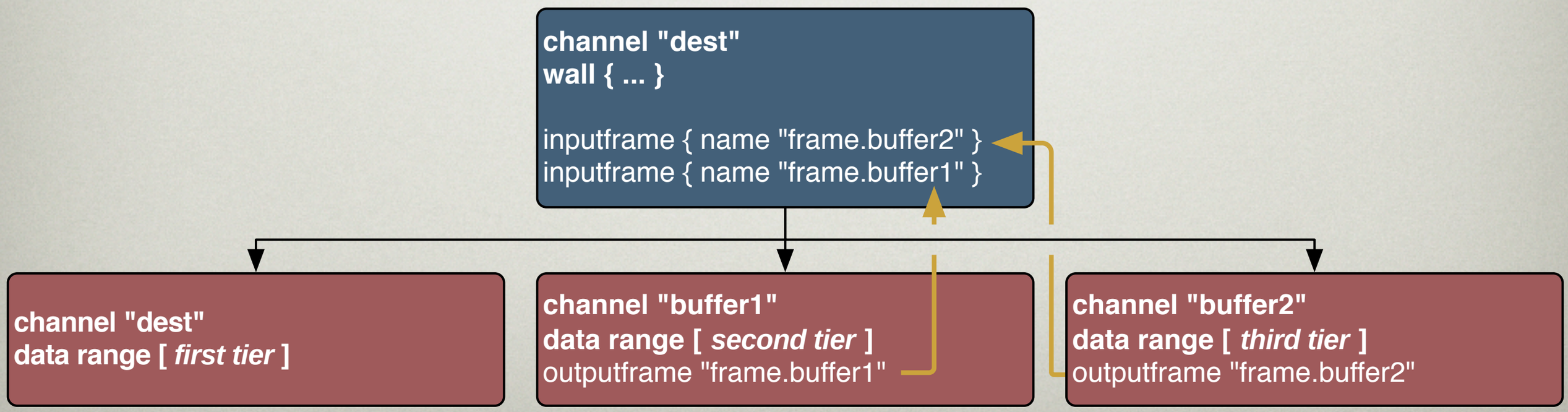
Implementation

- Part of Equalizer, an open source parallel rendering framework
- Application provides OpenGL code
- Configured using compound tree
- Compositing based on output and input frames
- Compositing algorithm not hard-coded!

VMML
visualization and multimedia lab
university of zürich

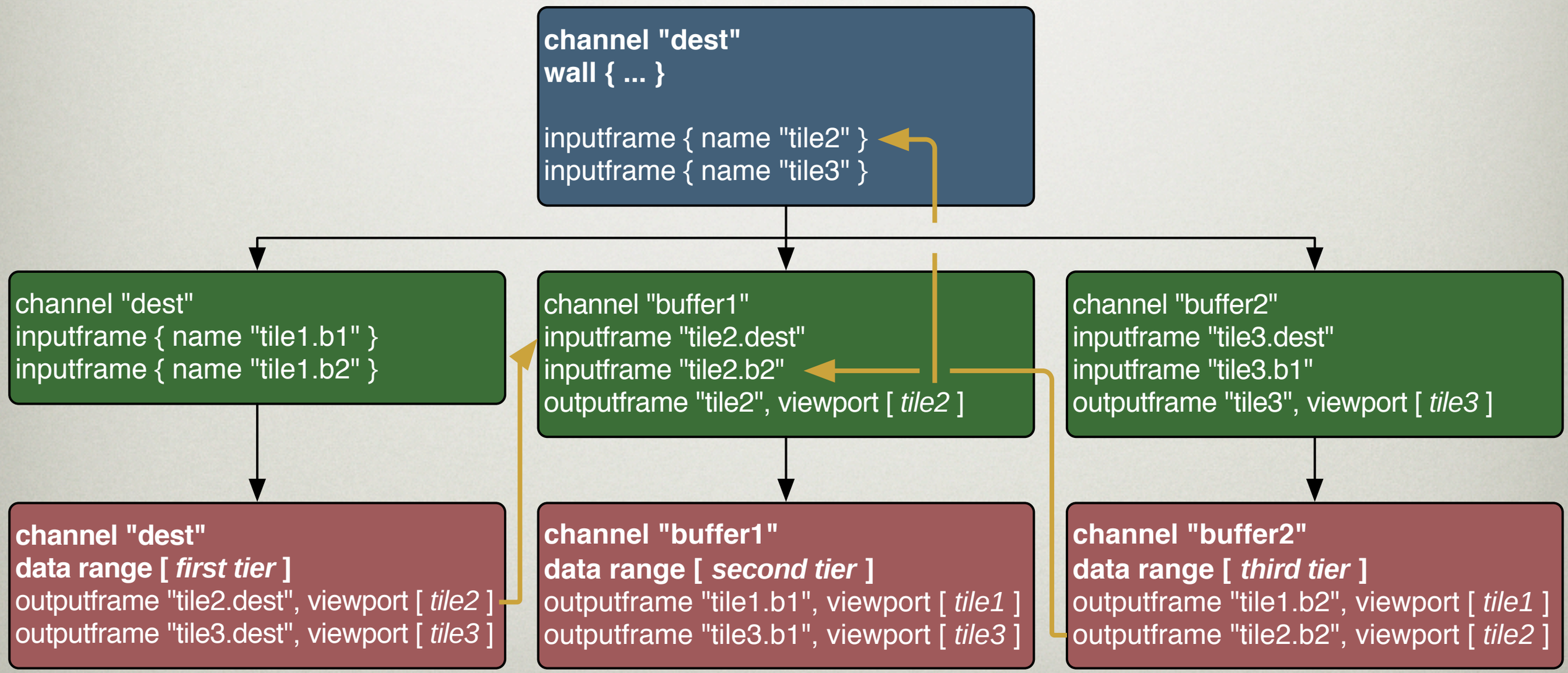
Implementation (2)

- Each compound uses a channel
- Root channel defines destination view
- Leaf compounds render for destination
- Sort-Last serial assembly tree:



Implementation (3)

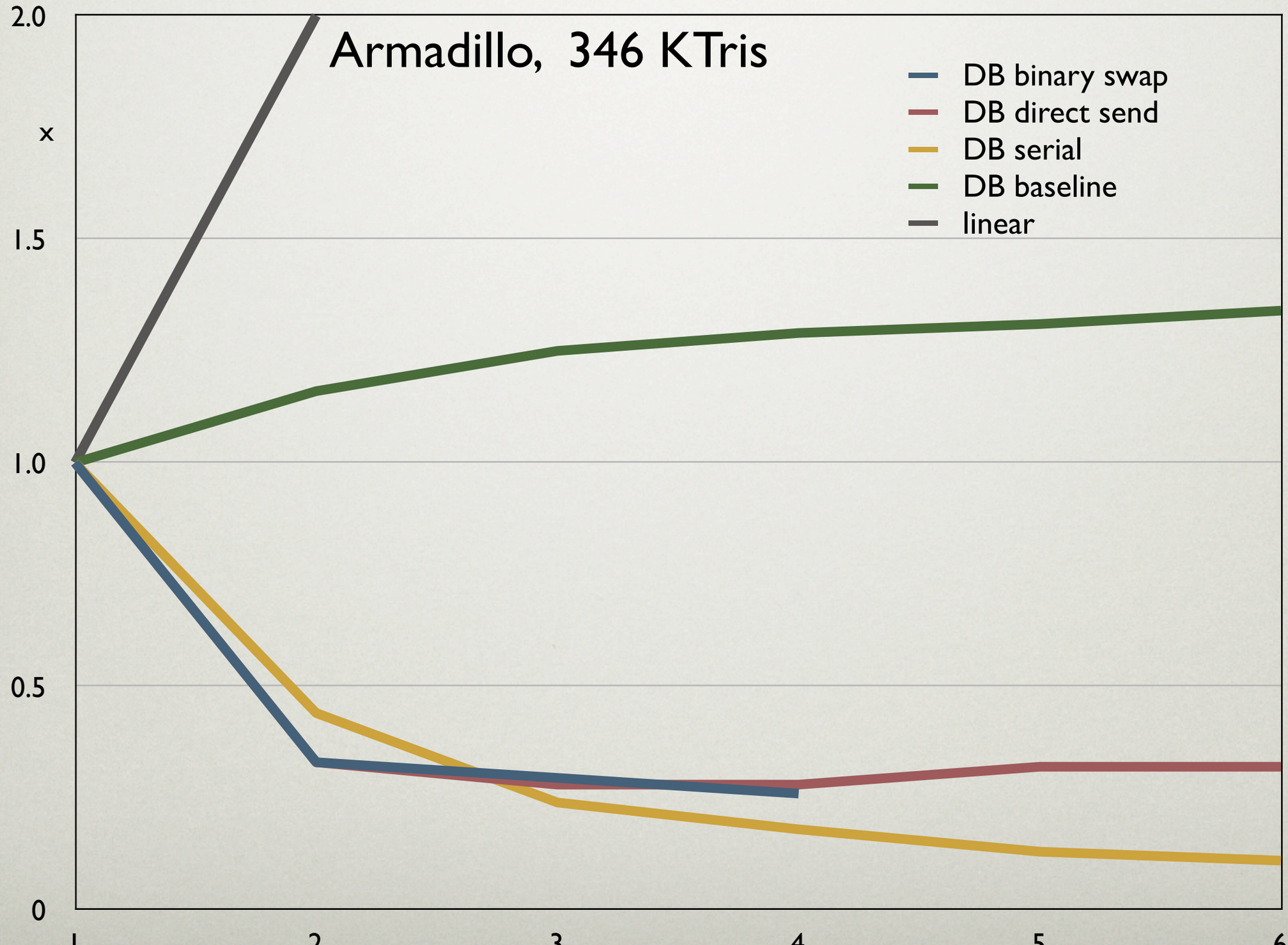
- Direct-Send compound tree:



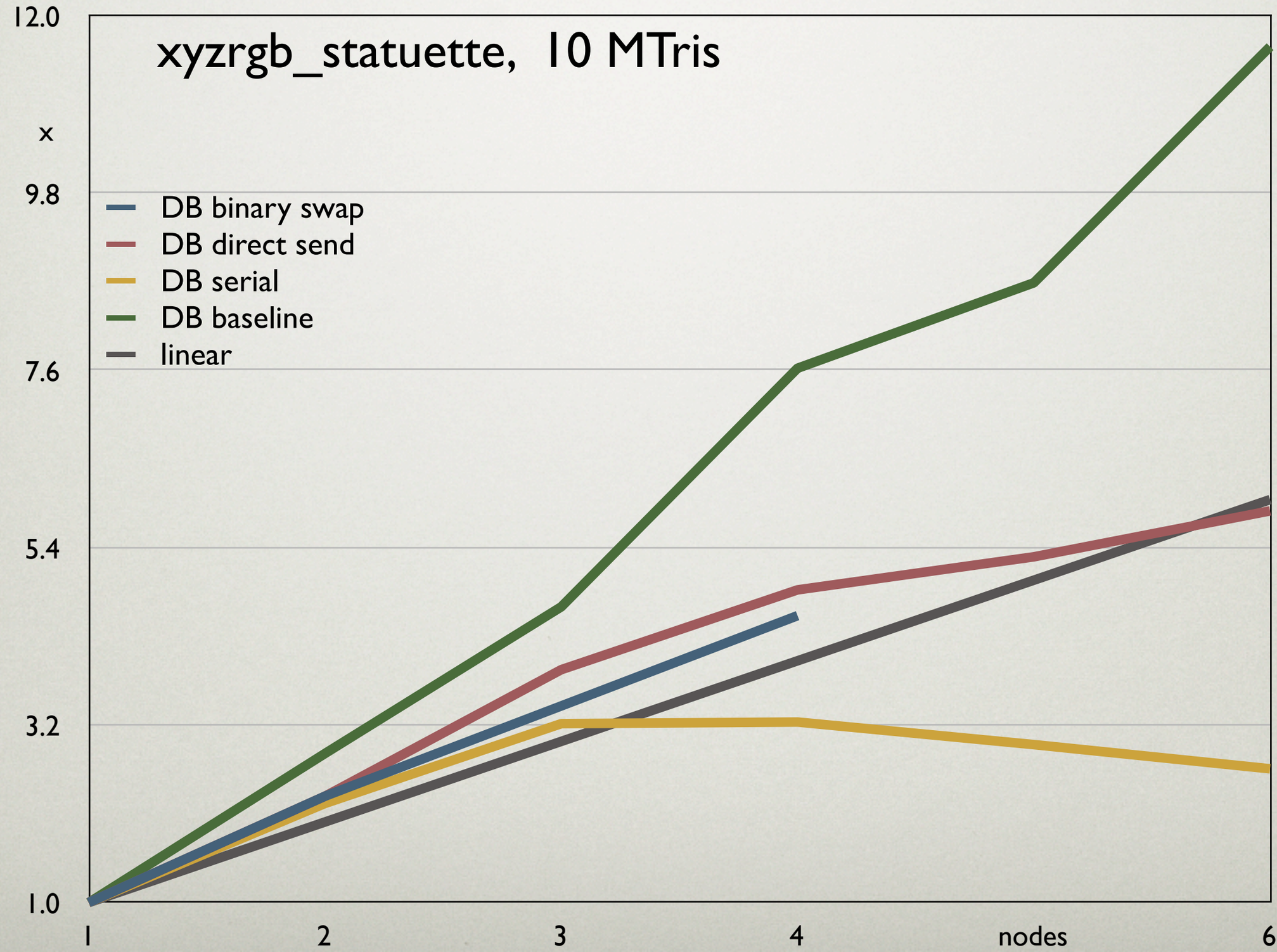
Results

- Hactar:
  - Six dual-Opteron nodes, 4 GB RAM
  - One Geforce 7800GTX per node
  - Gigabit Ethernet
- Polygonal Data: eqPly example
- Window size: 1280x800

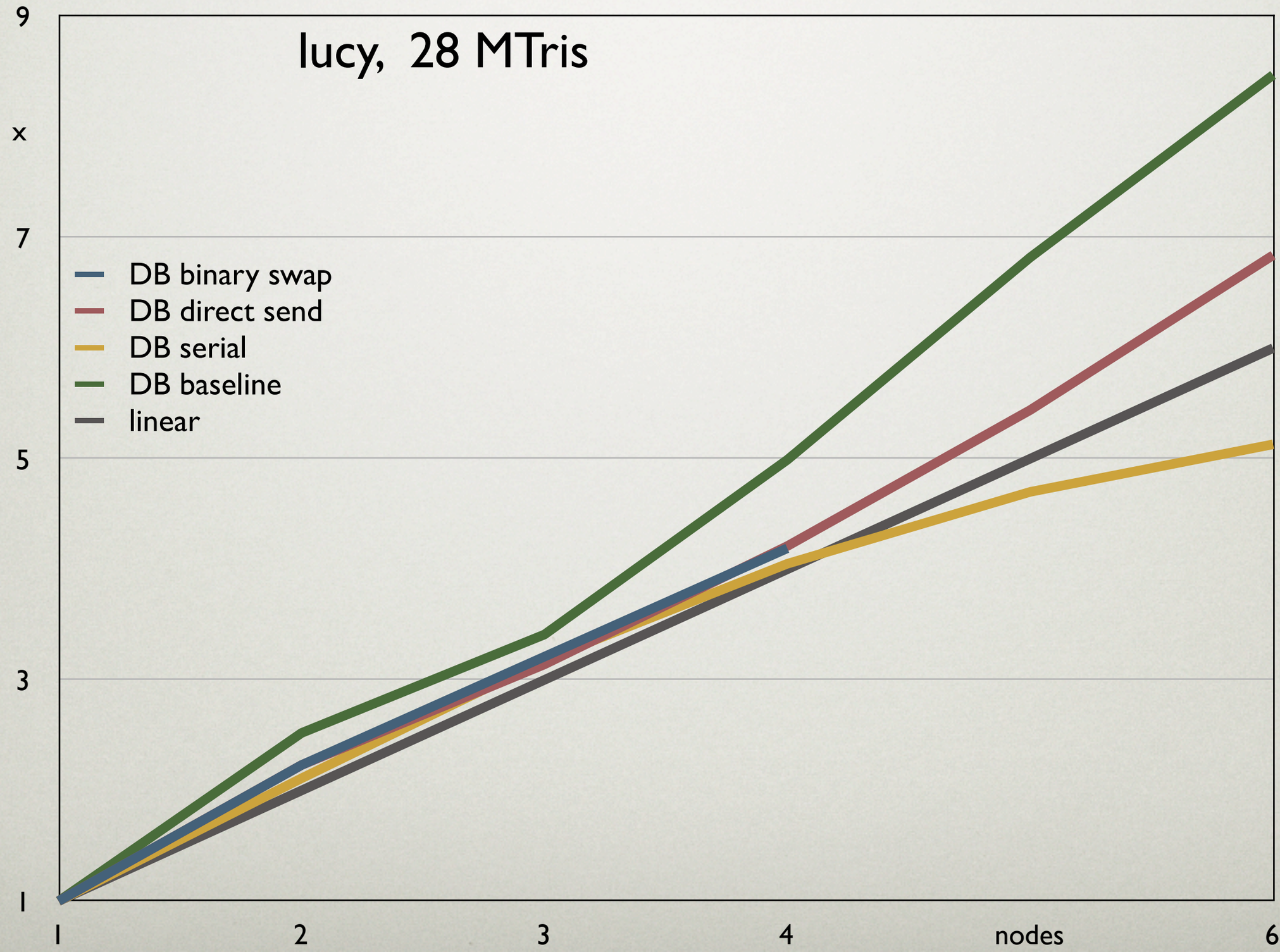
Small Model


Armadillo, 346 KTris
DB binary swap
DB direct send
DB serial
DB baseline
linear
2.0
x
1.5
1.0
0.5
0
1
2
3
4
5
6

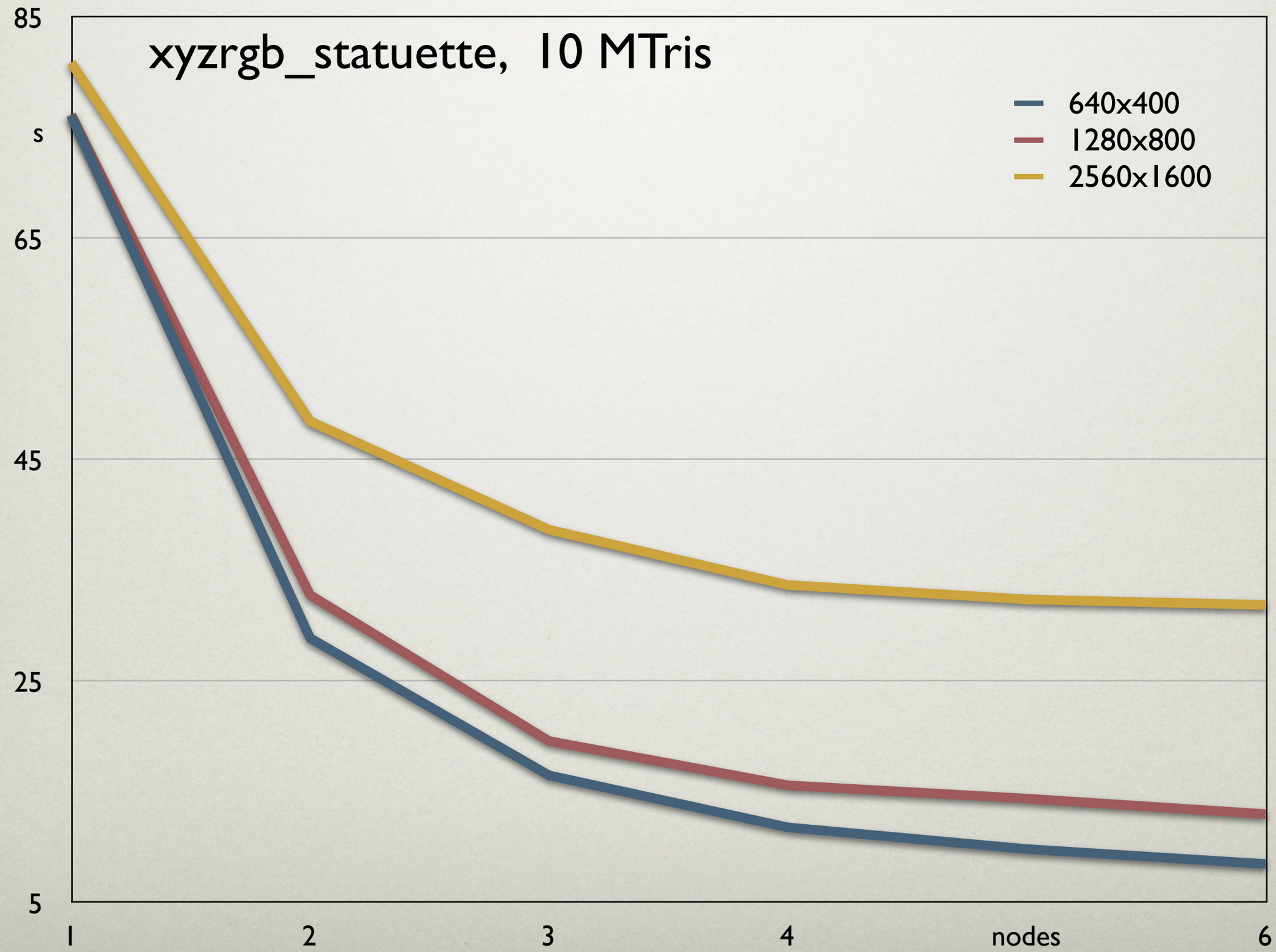
Medium Model



Large Model


lucy, 28 MTris
9
x
7
5
3
1
DB binary swap
DB direct send
DB serial
DB baseline
linear
1
2
3
4
nodes
6

Viewport Size



Results

- Direct-Send performs better or equal to binary-swap
- Direct-Send is more flexible
- Implementation behaves as expected
- Implementation is flexible for future algorithms and extensions

Future Work

- Full Equalizer Performance Evaluation
- Optimizations (ROI, SDP)
- 3D kd-Tree, VBO's for eqPly
- Volume Renderer

VMML
visualization and multimedia lab
university of zürich

Last Words

- Website: www.equalizergraphics.com
- Linux, Windows, Mac OS X
- LGPL license
- Open standard for scalable graphics
- **Demo: tomorrow 5pm at CSCS**
- Questions?

VMML
visualization and multimedia lab
university of zürich