

Equalizer

Get the best performance from your
visualization cluster

Outline

- High-performance visualization
 - Transparent and semi-transparent solutions
 - Programming interfaces
- Equalizer
 - Components
 - Features

What is HPV ?

- High-Performance Visualization - like HPC but for interactive 3D applications
- Address the demand to visualize huge data sets using COTS clusters
- Issue is to *scale* rendering performance using multiple GPU's and CPU's

HPC Analogy

	HPC	HPV
What?	Parallelize computation across multiple CPU's	Parallelize 3D rendering across multiple GPU's and CPU's
How?	Mostly non-interactive batch processing	Highly interactive, real-time rendering
Hardware	Cluster or Supercomputers typically using fast interconnects	Graphic Cluster, Supercomputers, display hardware, input devices

HPV Today

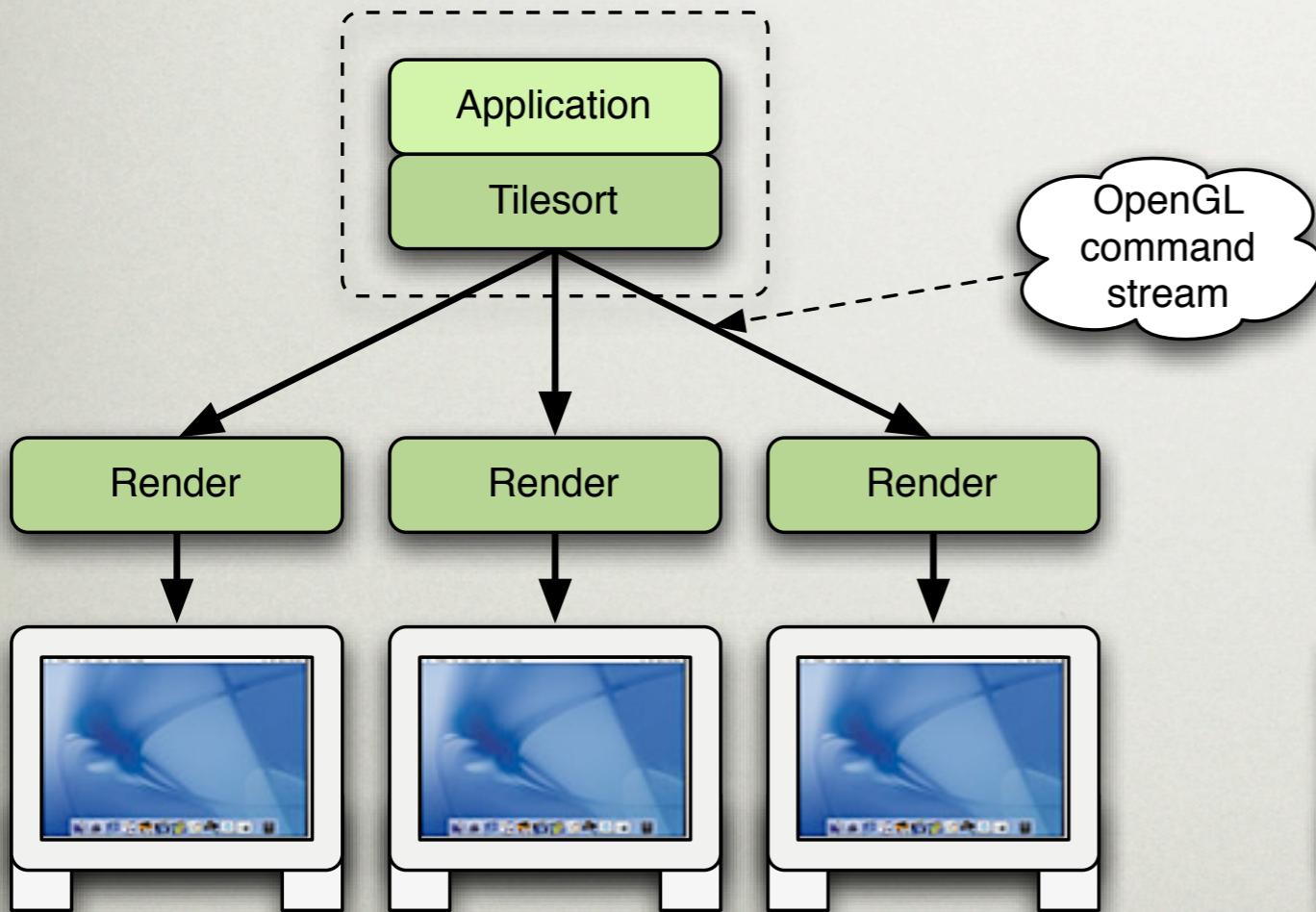
- Transparent and semi-transparent solutions
- Programming interfaces
- Equalizer

HPV Transparent Solutions

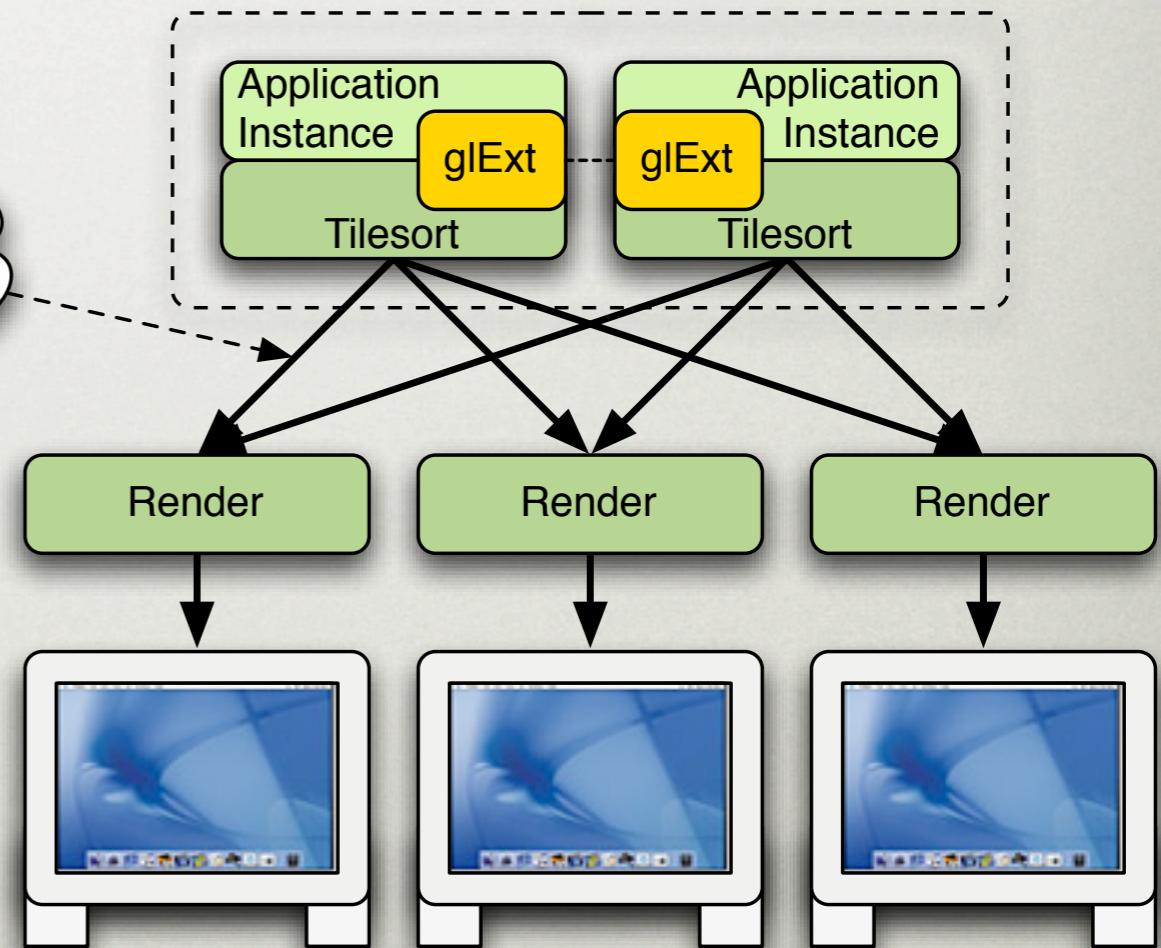
- Chromium, ModViz VGP, OMP, ...
 - Operate on OpenGL command stream (HPC analogy: auto-parallelizing compilers)
 - Provide programming extensions for improved performance and scalability (semi-transparent)

HPV Transparent Solutions

Transparent



Semi-Transparent



HPV Programming Interfaces

- ScaleViz, Vega Prime, VTK
 - Impose overall programming model and data structure (HPC analogy: CFD codes)
 - Best for developing new applications
- Cavelib, VRJuggler, MPK
 - Limited to HPV-critical areas of the code (HPC analogy: MPI, PVM)
 - Best for porting existing applications

Equalizer

A Programming Interface

and

Resource Management System

for

Scalable Graphics Applications

Equalizer Programming Interface

Applications are written against a *client library* which abstracts the interface to the execution environment

- Minimally invasive programming approach
- Abstracts multi-processing, synchronisation and data transport
- Supports distributed rendering and performs frame compositing

Resource Management System

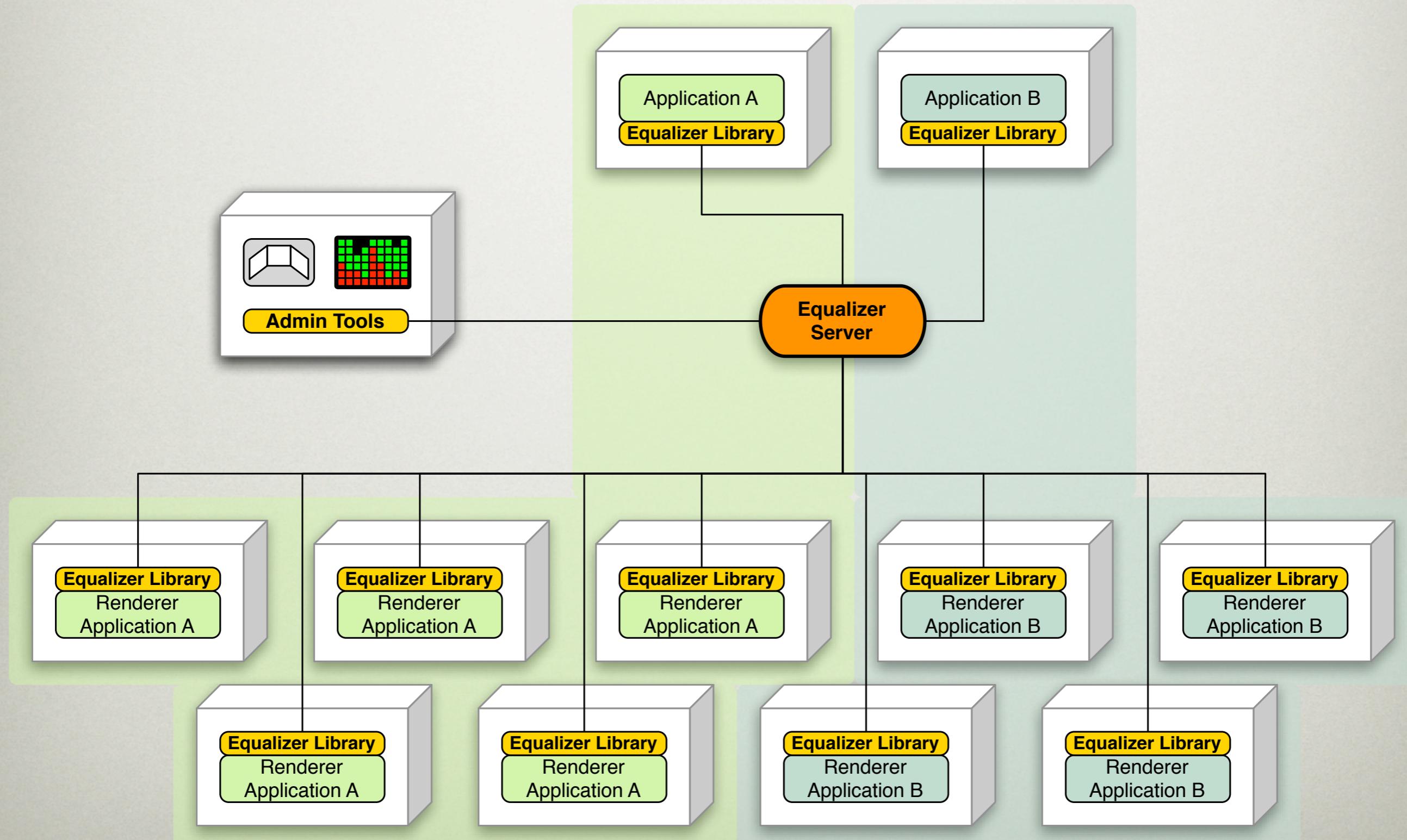
Applications are deployed by a *server* which balances the resource usage across the system

- Centralizes the setup for all applications
- Configures application and deploys render clients
- Dynamic load-balancing of the cluster resources

Equalizer Tools

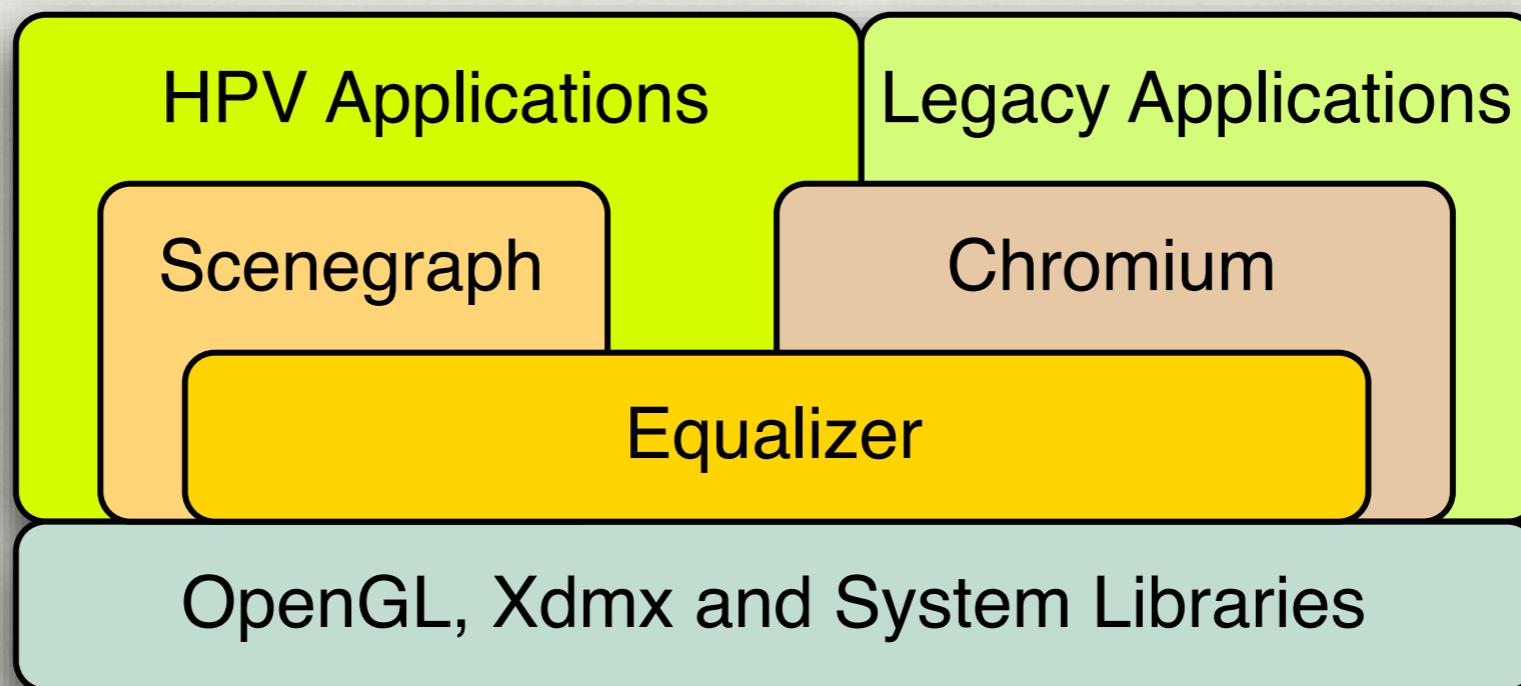
- Configuration
 - Dynamic reconfiguration of the server
- Monitoring
 - Resource usage, bandwidth utilization, statistics
- Profiling
 - Execution flow, data distribution

Equalizer Components

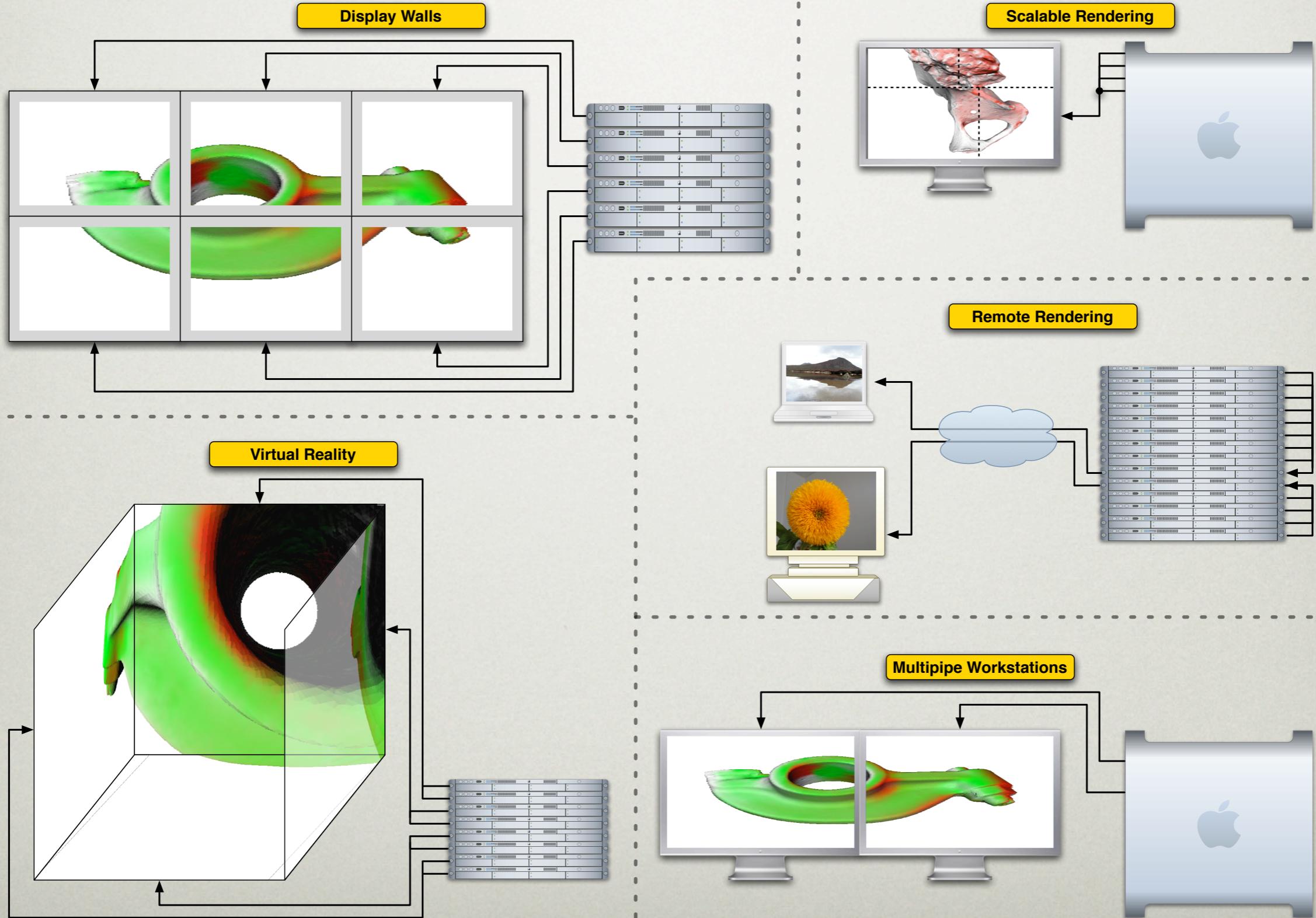


Equalizer Vision

- Xdmx: X11 single virtual screen
- Chromium: OpenGL single virtual screen
- Equalizer: Scalable rendering engine



Selected Use Cases



Equalizer Features

- Runtime configuration
- Runtime scalability
- Minimally invasive
- Asynchronous execution
- Clusters and SSI
- Open Source (LGPL)

Runtime Configuration

- Hierarchical resource description:
Node → Pipe → Window → Channel
 - Node: single system of the cluster
 - Pipe: graphic card
 - Window: drawable and context
 - Channel: view
- Resource usage: compound tree

Runtime Scalability

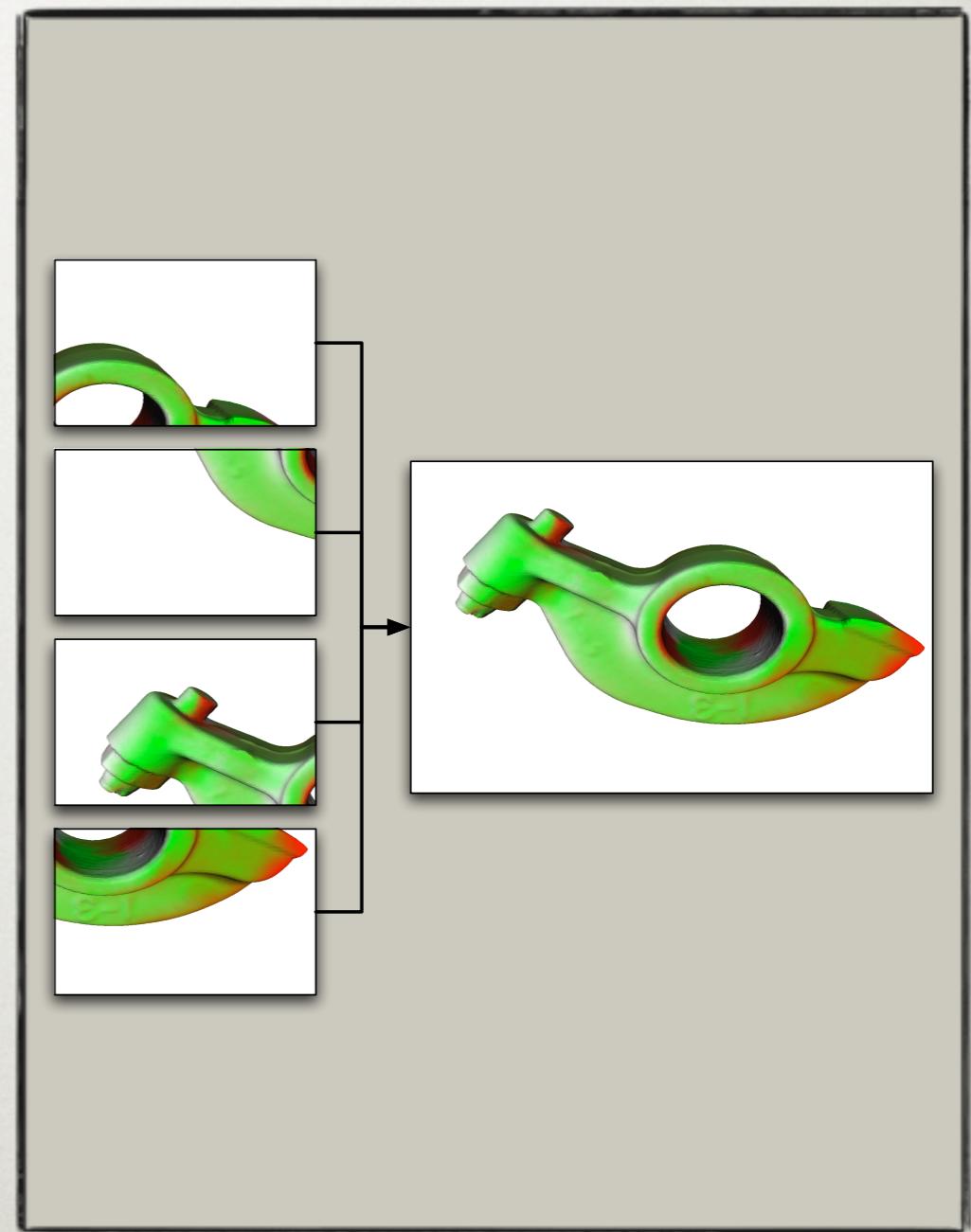
- Parallel execution of the application's rendering code
- One thread per graphic card
- Decomposition of rendering for one view
- Server can choose and adapts configurations based on system resources, topology and load

Runtime Scalability

- Sort-first (2D), sort-last (DB), time-multiplex (DPlex), stereo (eye)
- Flexible configuration of decomposition and recomposition
- Later: automatic loadbalancing
- Supports compositing hardware
- Hardware-specific optimizations

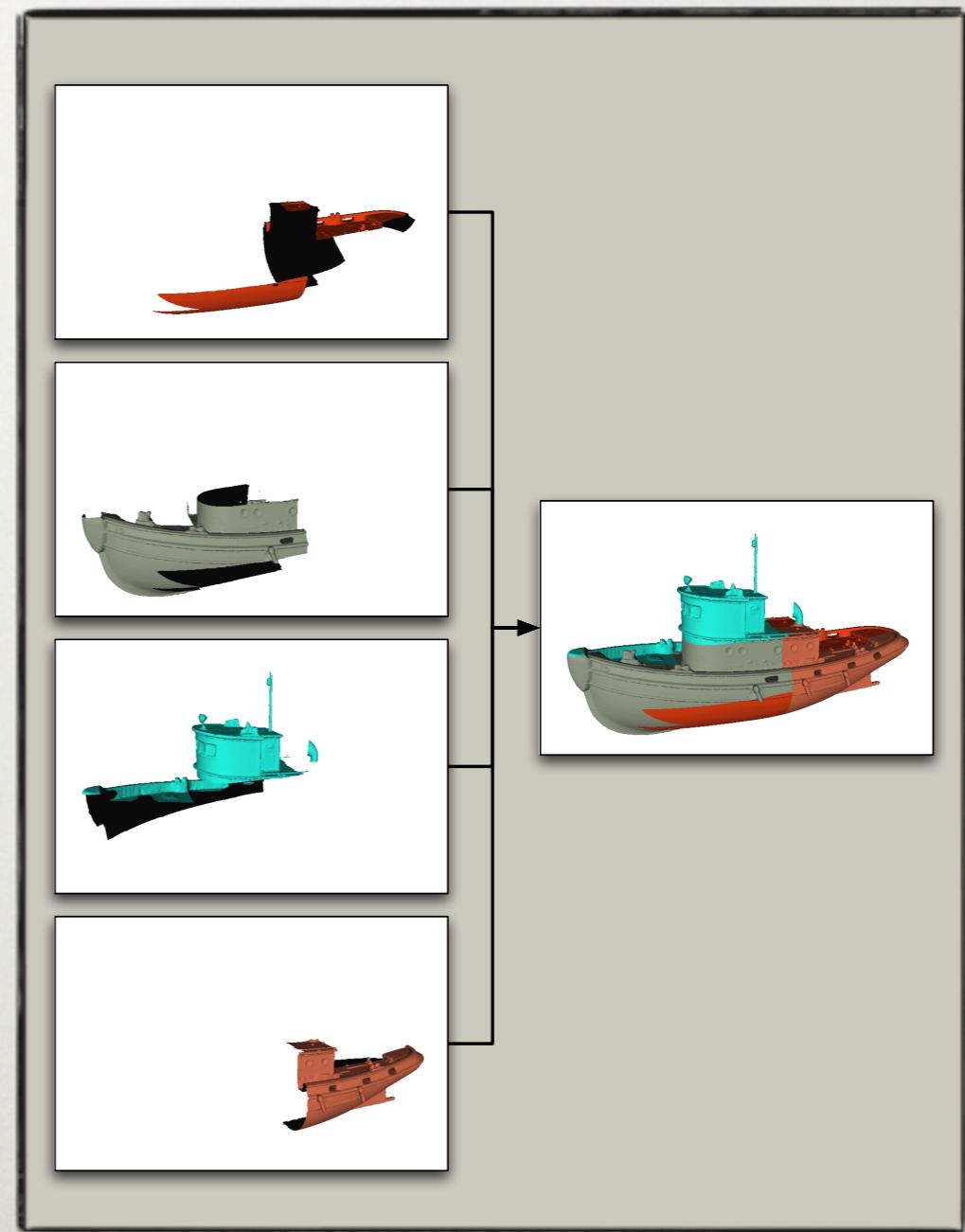
2D / Sort-First

- Scales fillrate
- Scales geometry when used with efficient view frustum culling
- Parallel overhead due to primitive overlap limits scalability



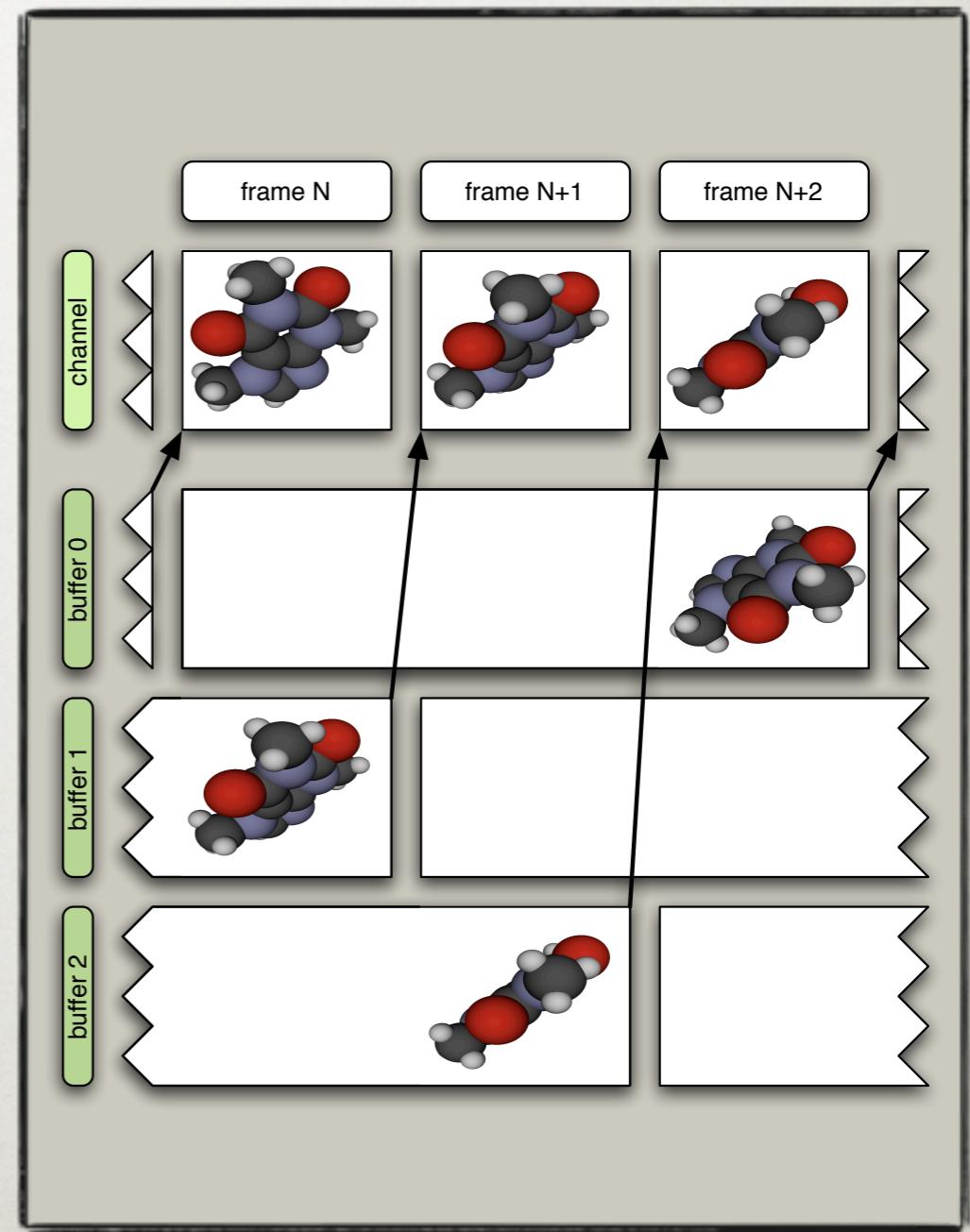
DB/Sort-Last

- Scales all aspects of rendering pipeline
- Application needs to be adapted to render subrange of data
- Recomposition relatively expensive



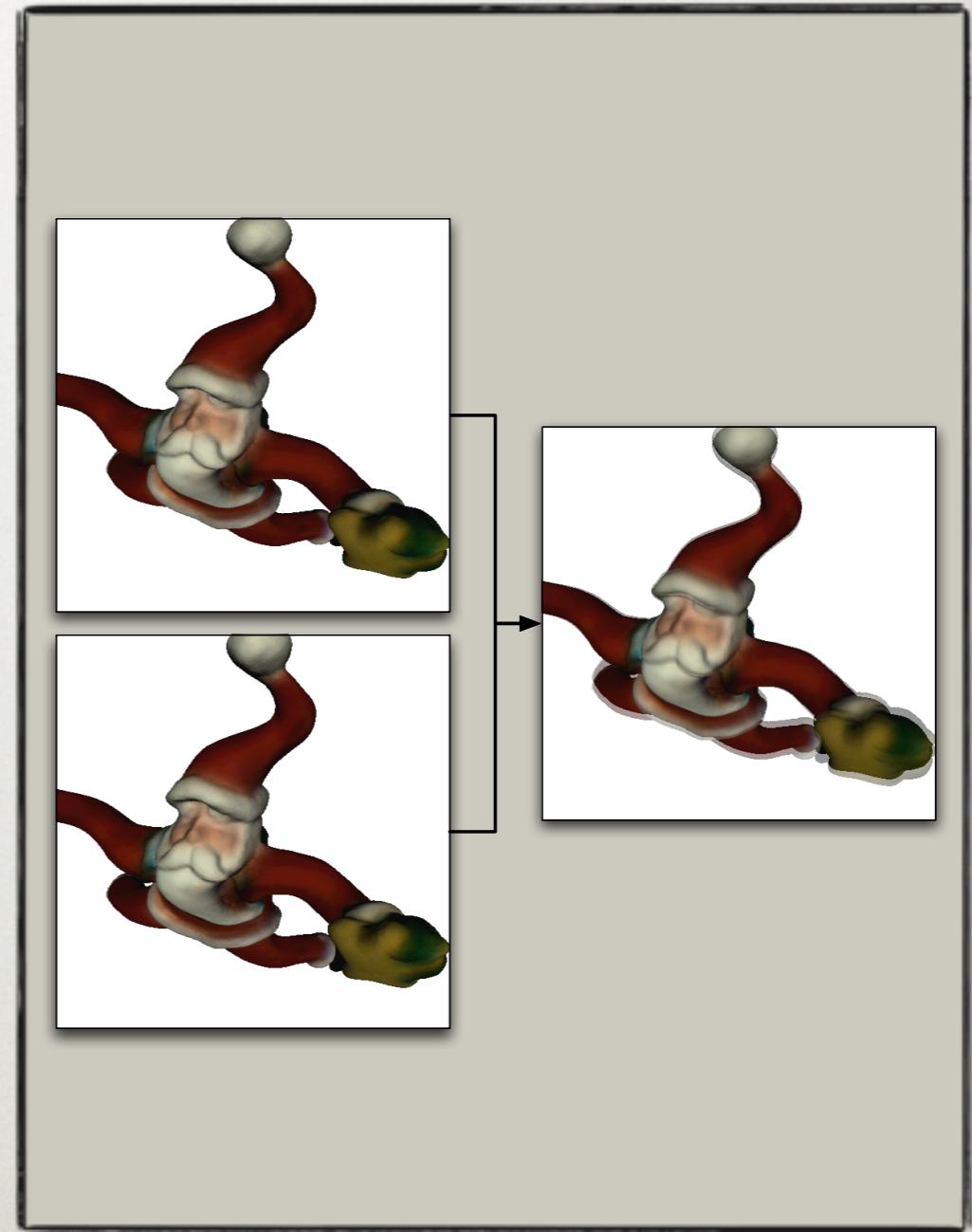
DPlex / Time-Multiplex

- Good scalability and loadbalancing
- Increased latency may be an issue
- Not yet implemented



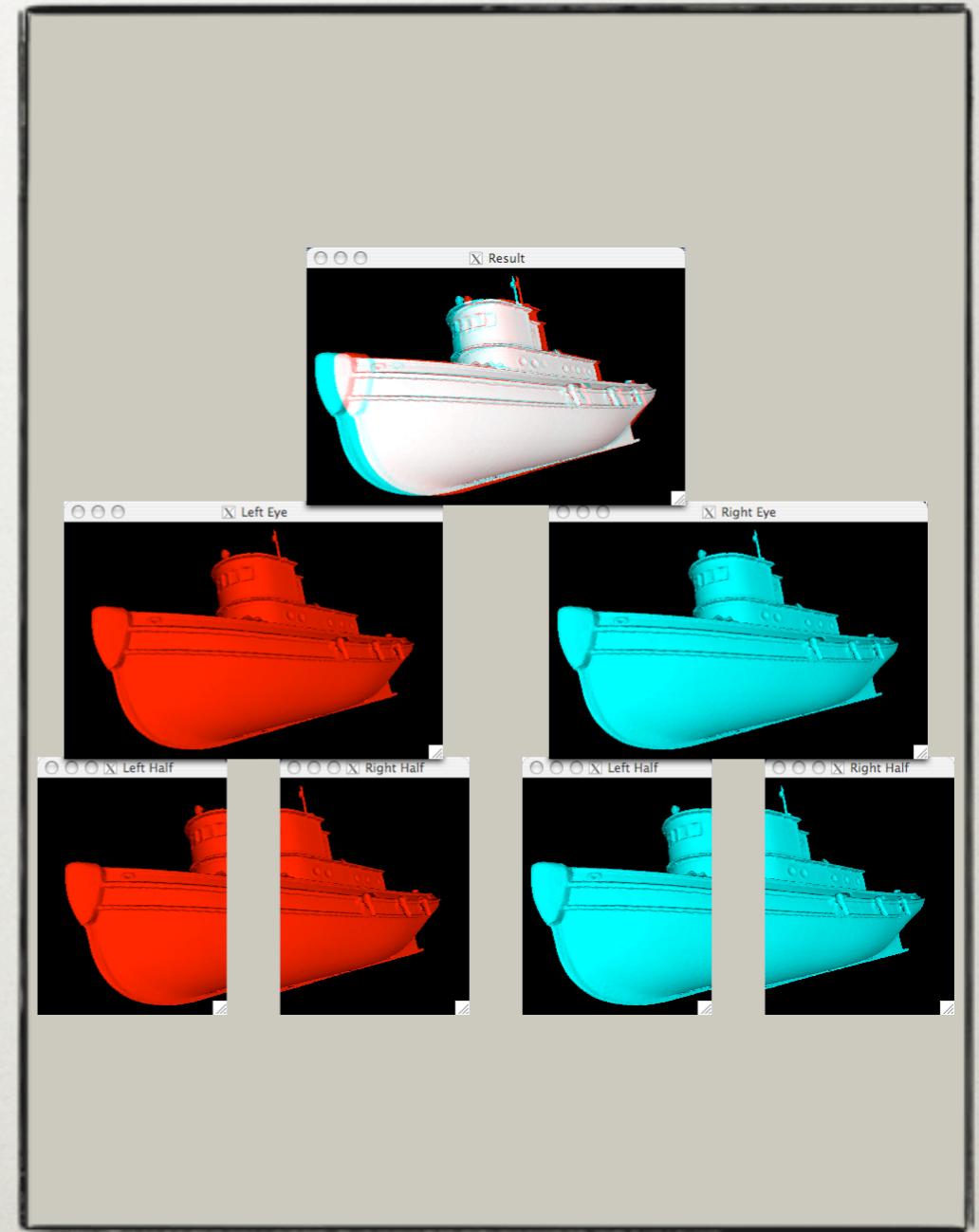
Eye Compound

- Stereo rendering
- Active, passive and anaglyphic stereo
- Excellent loadbalancing
- Limited by number of eye views



Multilevel Compounds

- Compounds allow any combination of modes
- Combine different algorithm to address and balance bottlenecks
- Flexible configuration of recombination algorithm



Minimally Invasive

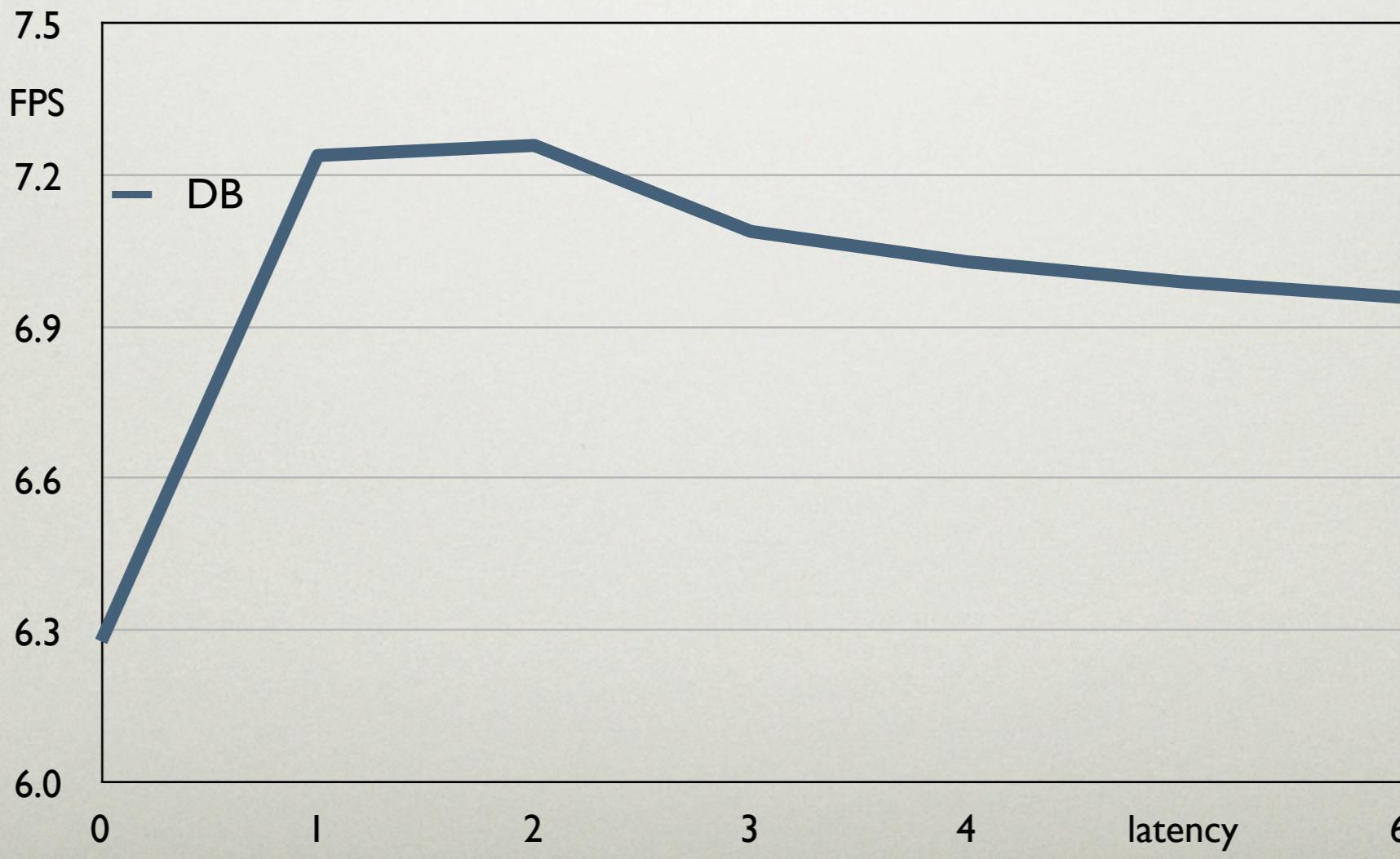
- “Make everything as simple as possible, but not simpler.” -- Albert Einstein
- Porting is as easy as possible
- Work is limited to visualization-relevant parts
- Simpler approaches compromise performance

Asynchronous Execution

- Source channels start rendering the next frame early
- Hides imbalance in load distribution
- Only visible channels belonging to the same view are synchronized
- Greatly improves scalability on bigger pipe counts

Asynchronous Execution

- Example: 5-node sort-last, direct-send
- 15% speedup, no loadbalancing



SSI and Clusters

- Supercomputers are just tightly integrated clusters
- Equalizer runs on both architectures
- Overall model is the same
- Shared memory systems allow additional optimisations
- Porting for SSI simpler than full port

Open Source

- LGPL license
- Open standard for parallel rendering
- User-driven development
- Latest version available on:
www.equalizergraphics.com
- [Get in touch](#)