Equalizer

Enabling High-Performance Visualization

Outline

- High-Performance Visualization
  - Overview
  - Current State
- Equalizer Project Description
  - Overview
  - Components and Features
  - Programming Model

High Performance Visualization ?

- HPV: High-Performance Computing for Interactive 3D Applications
- Parallel Rendering using multiple GPU's and CPU's
- Emerging Approach to visualize huge Amounts of Data using COTS Clusters

HPC vs. HPV Overview

	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 single systems, fast interconnects	Cluster or single systems with multiple graphic cards, fast interconnects

HPC vs. HPV Programming Tools

	HPC	HPV
Transparent Solutions	Auto-parallelizing compilers, Mosix, DSM systems	OpenGL intercept libraries (Chromium, ModViz)
Frameworks	MPI, PVM	n/a
Solutions	CFD tools, ...	OpenSceneGraph, Vega Prime, VTK, ...

HPV today (I)

- Transparent and Semi-Transparent Solutions
  - Intercept OpenGL Command Stream
  - Limited Performance and Scalability
  - Chromium, ModViz

HPV today (II)

- Multi-Pipe Frameworks
  - Provide Execution Framework
  - Help Application to achieve ideal Scalability
  - No Cluster-Capable, scalable Toolkit available
  - OpenGL Multipipe SDK (SSI only)
  - Cavelib (Multi-Frustum only)

HPV today (III)

- Scenegraph-Based Solutions
  - Specialized for a certain Purpose
  - Impose Rendering Model on Application
  - Allow ideal Scalability and Performance
  - Can use Multipipe Framework
  - OpenSceneGraph, Vega Prime, VTK

Equalizer Overview

- A Framework for Parallel, Scalable Visualization
- Minimally Invasive Execution Framework for Multipipe Rendering
- Focus on Scalability and Performance
- Integration with Chromium provides Transition Path for existing Applications

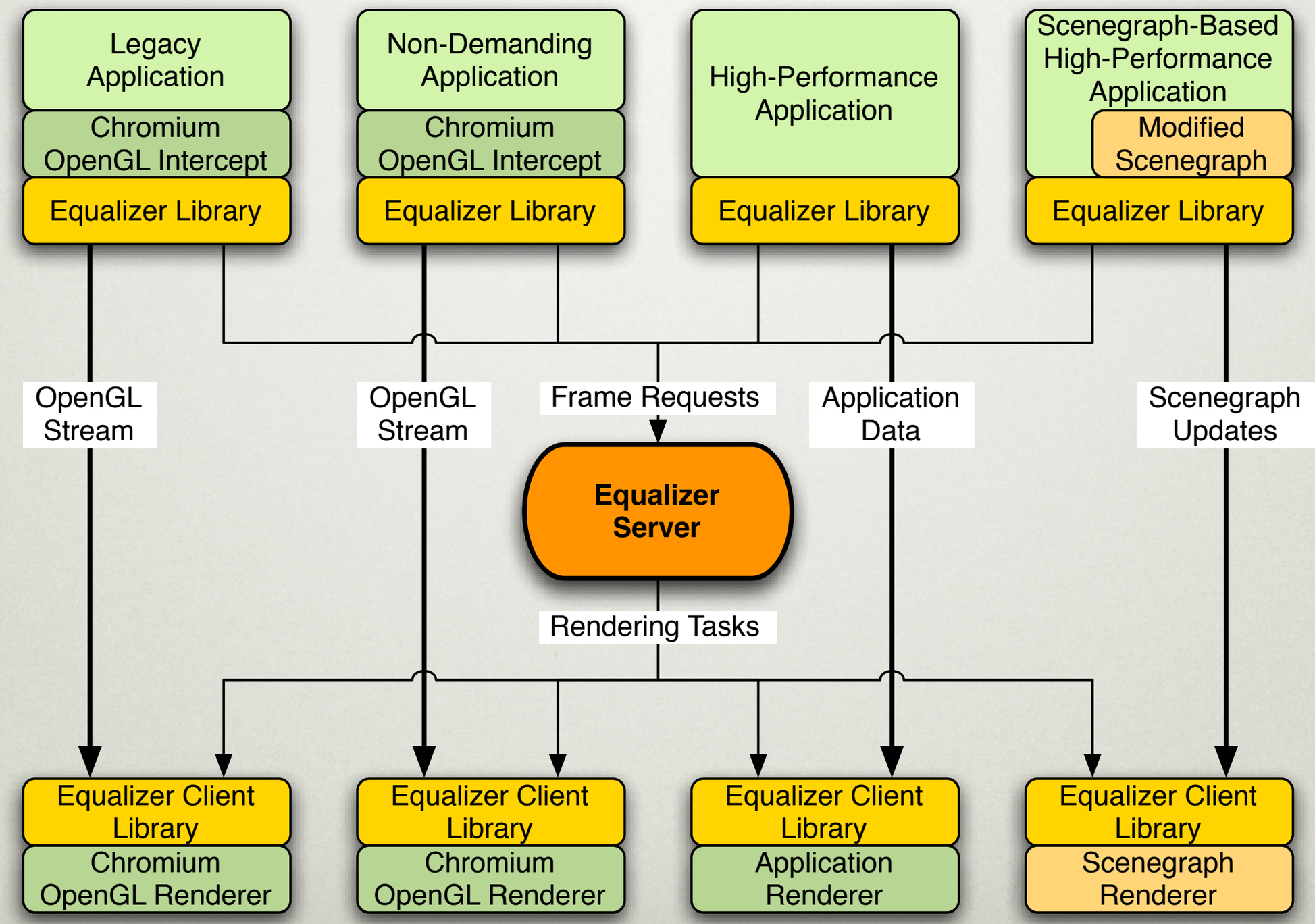
Equalizer Components

- Configuration Server
  - Manages Cluster Resources
  - Configures Clients
- Client Library
  - Provides Server Interface
- Administrative Tools
  - Configuration, Debugging and Performance Tools

Visualization Cluster Components

- Xdmx: Single Virtual Screen for X11 (2D)
- Chromium: 3D for Single Virtual Screen
  - Legacy and Non-Demanding Applications
- Equalizer: ‘MPI for Graphics’
  - Scenegraphs are layered on top of Equalizer
- All three have to be integrated seamlessly

Equalizer Components



Equalizer Features

- Runtime Configuration
- Runtime Scalability
- Asynchronous Execution
- Remote Visualization
- SSI and Clusters
- Open Source

Runtime Configuration

- Systemwide Configuration Server
  - One Setup for all Applications
  - Centralized Resource Allocation
  - Systemwide Optimisations and Load Distribution
  - Deploys Applications' Render Client
- ASCII Configuration Files
- Application is unaware of Configuration

Runtime Configuration

- Hierarchical Resource Description
  - Node: Single System of the Cluster
    - Pipe: Graphic Card
      - Window: OpenGL Drawable and Context
        - Channel: OpenGL View

Runtime Scalability

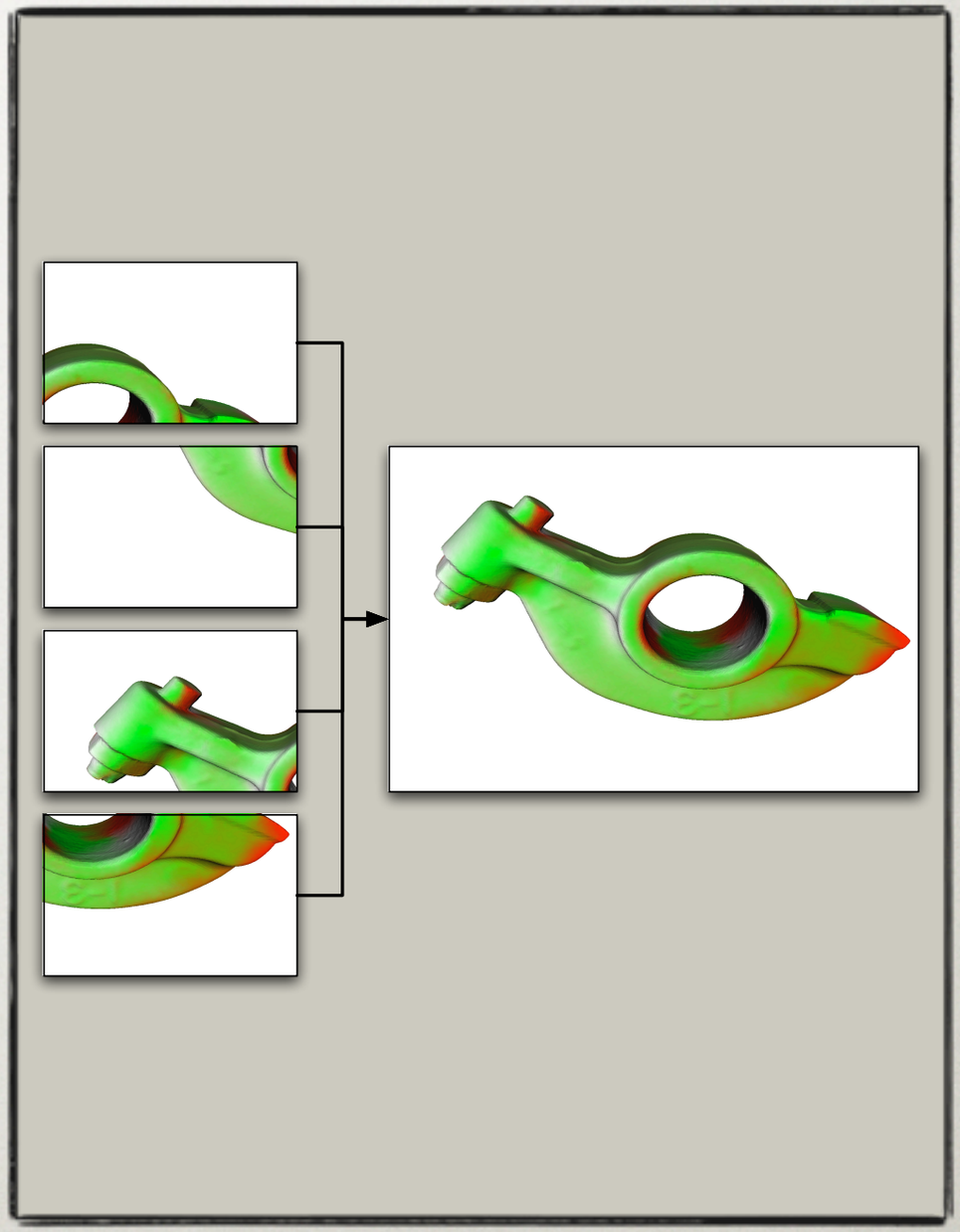
- Parallel Execution of the Applications' Rendering Code
- One Thread per Graphic Card
- Decomposition of Rendering for one View
- Server chooses and adapts Configuration based on System Resources, Topology and Load

Runtime Scalability

- Modes: 2D, DB, DPlex, Eye
- Supports Compositing Hardware
- Automatic Loadbalancing
- Readback Optimisations

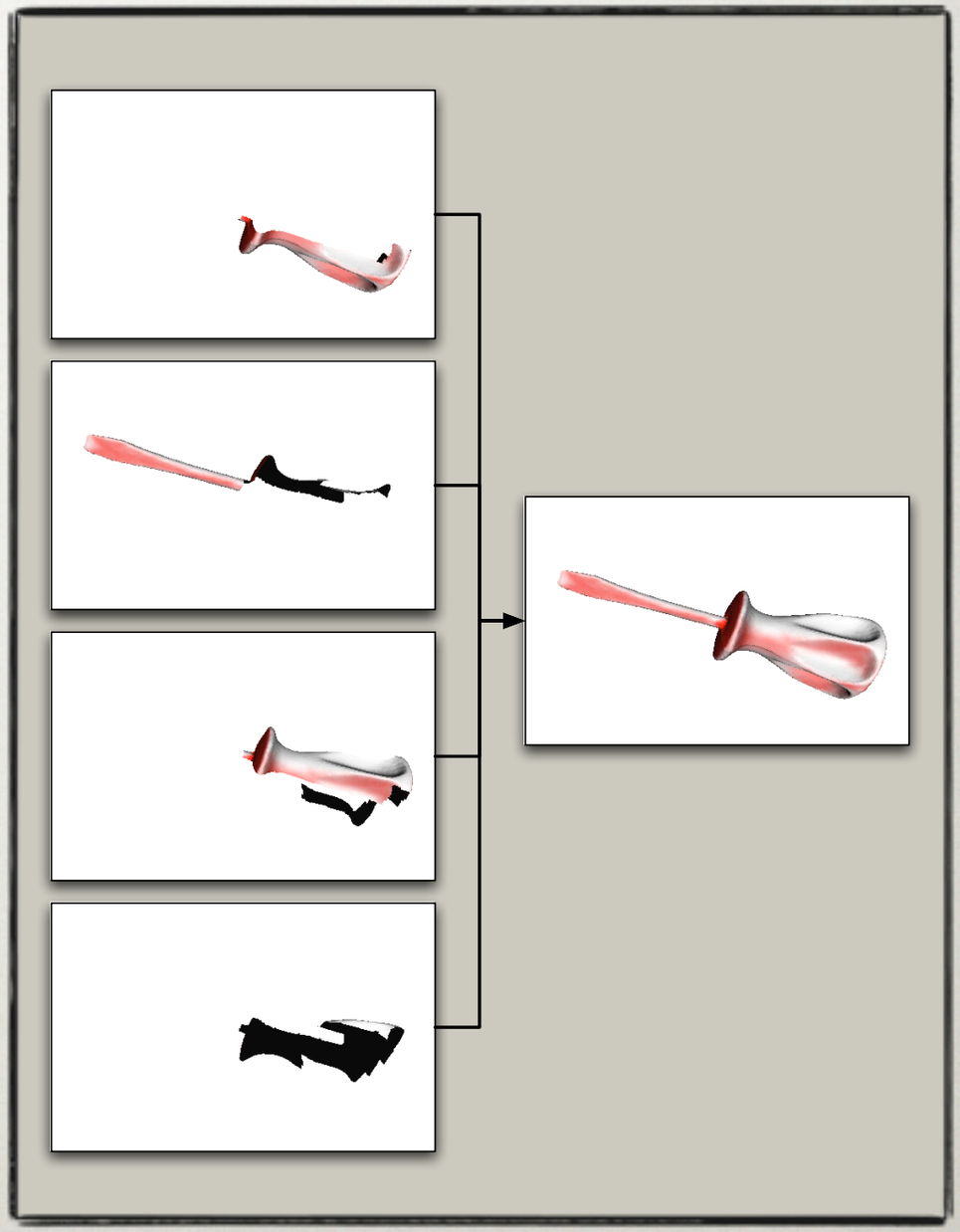
Runtime Scalability
2D Compound

- Sort-First
- Scales Fillrate
- Application needs good View Frustum Culling



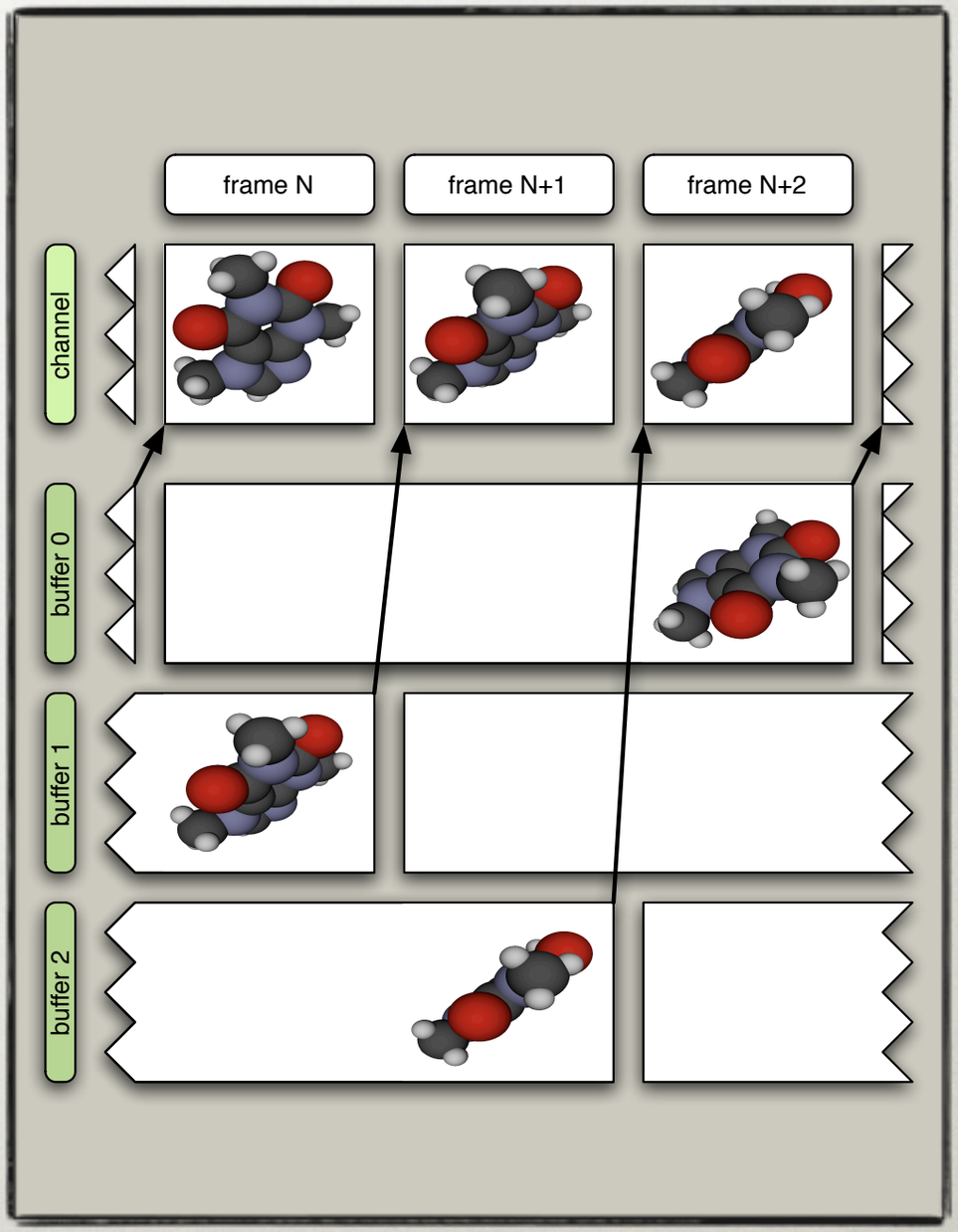
Runtime Scalability DB Compound

- Sort-last
- Good Task Distribution
- Application needs to be adapted to render Subrange of Data



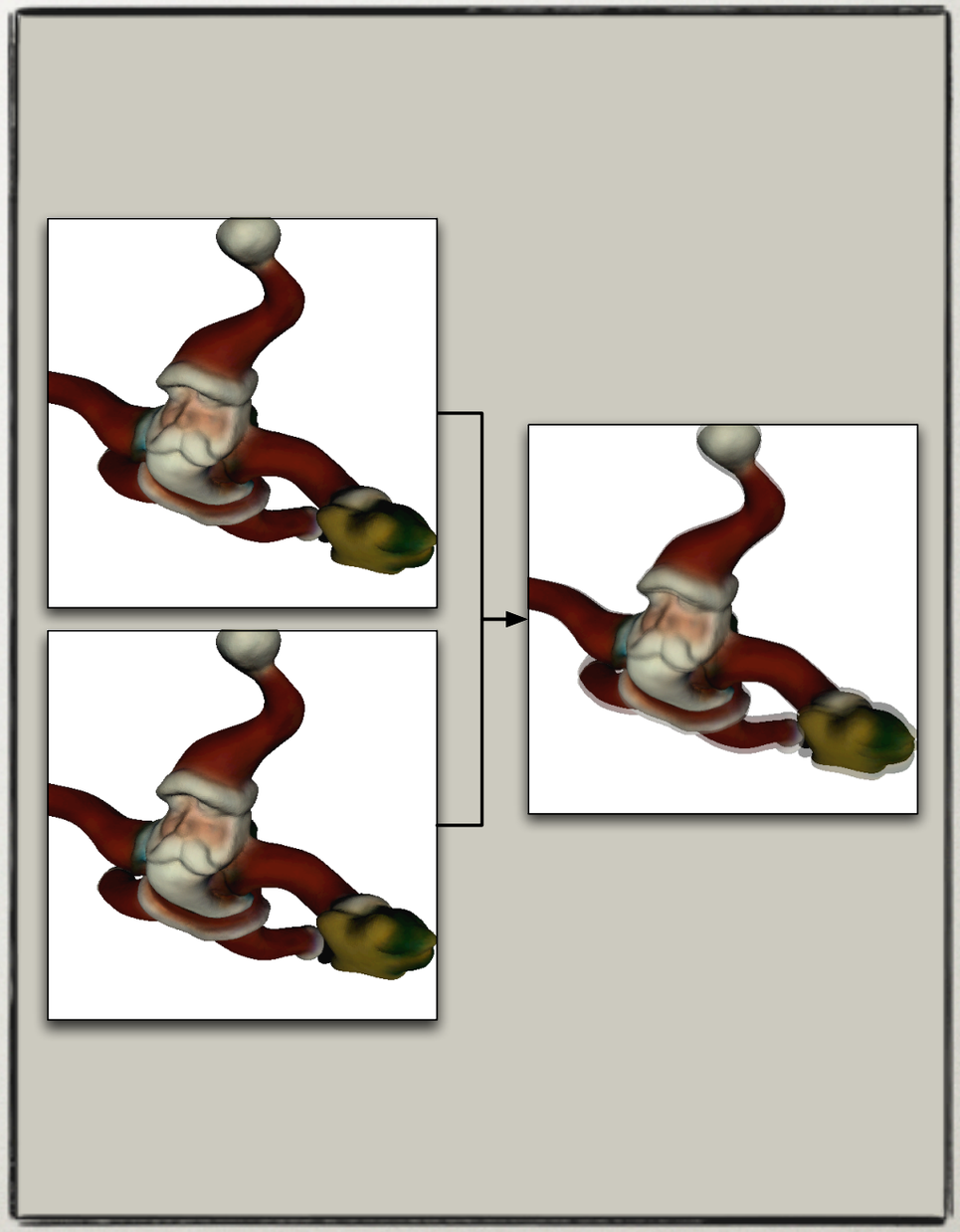
Runtime Scalability DPlex Compound

- Time-Multiplex
- Good Scalability and Loadbalancing
- Increased Latency may be an Issue



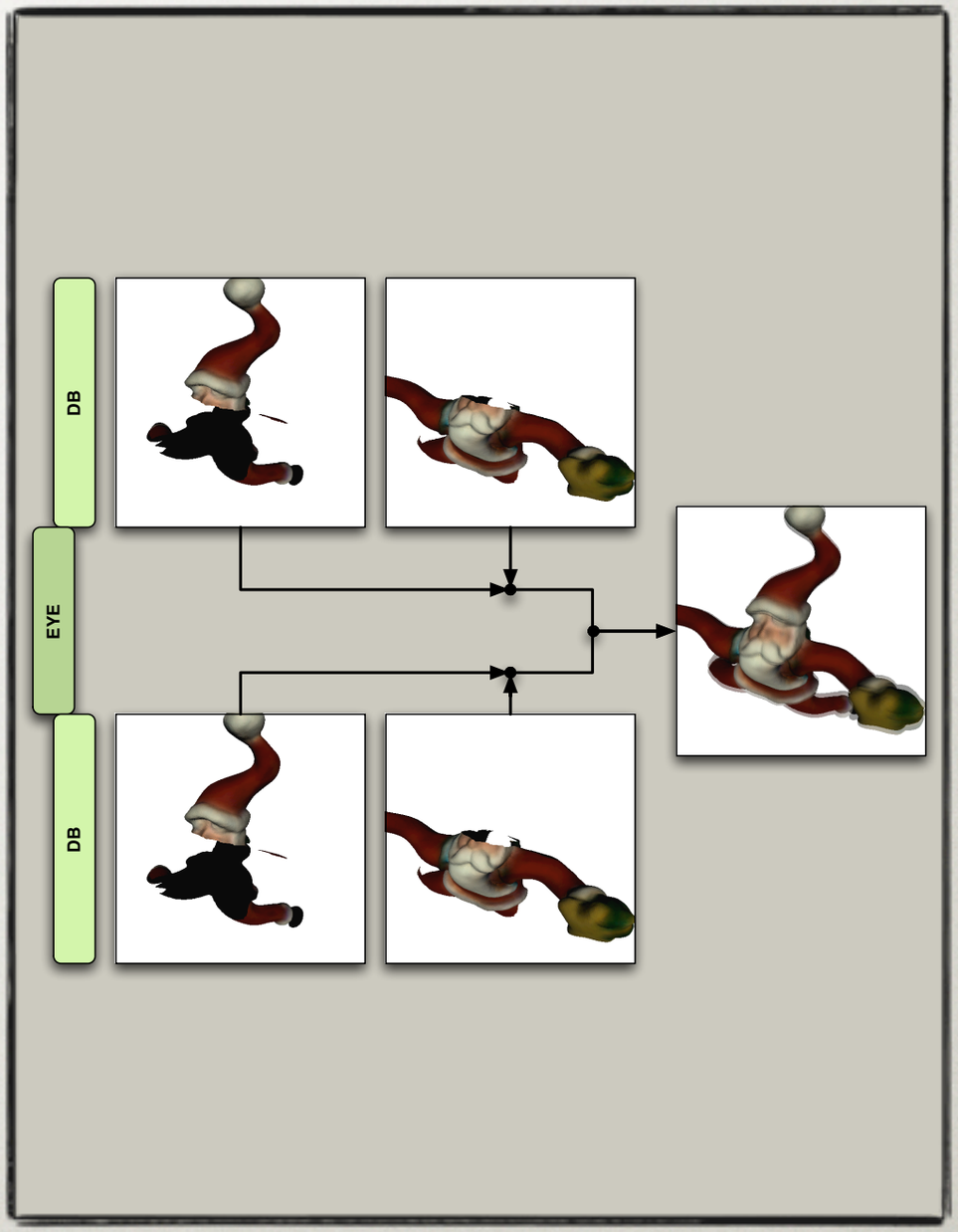
Runtime Scalability
Eye Compound

- Stereo Rendering
- Very good Loadbalancing
- Limited by Number of Eye Views



Runtime Scalability Multilevel Compounds

- Compound Tree allows any Combination of Modes
- Combine different Modes to address different Bottlenecks



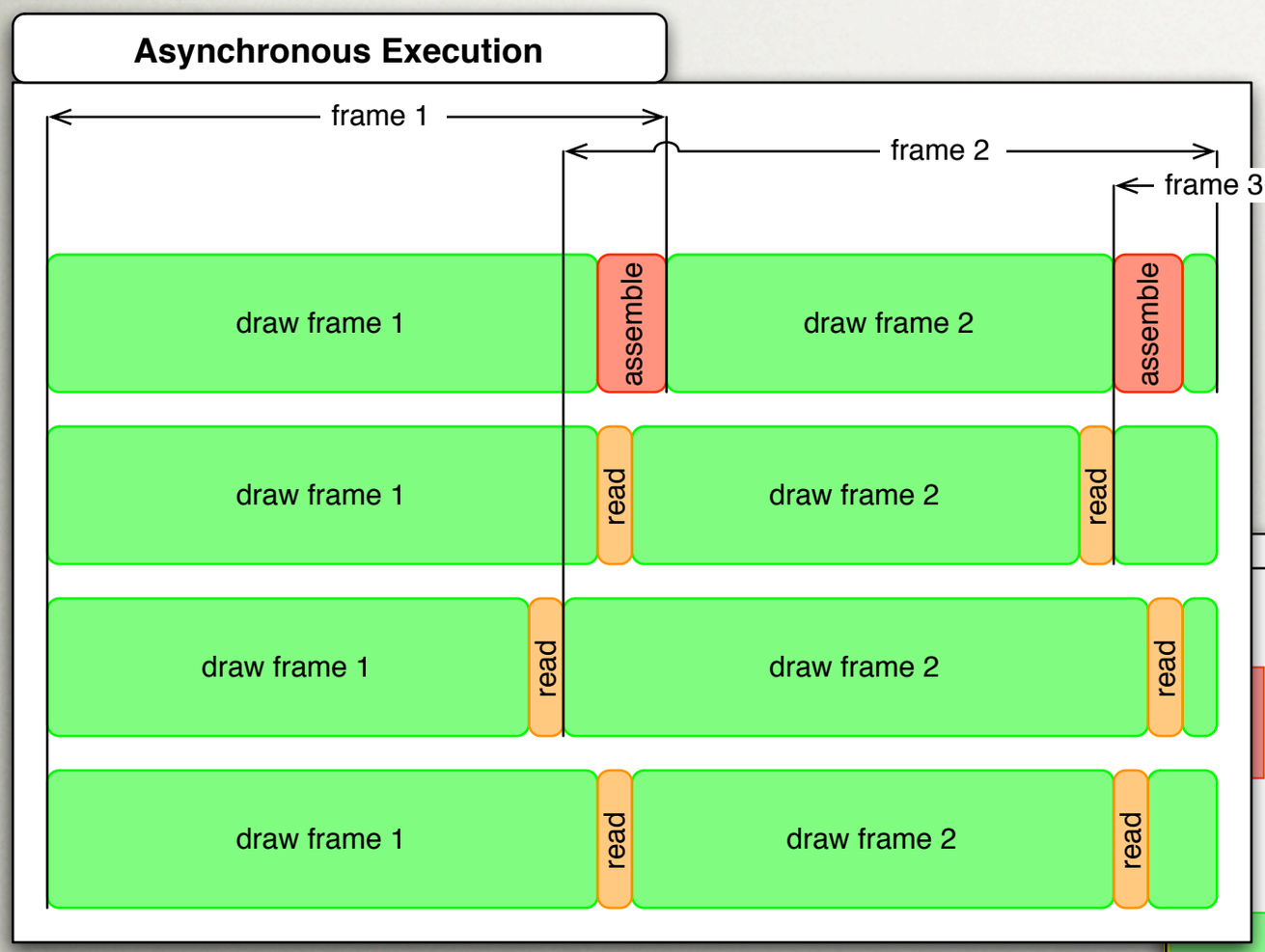
Decomposition Modes Conclusion

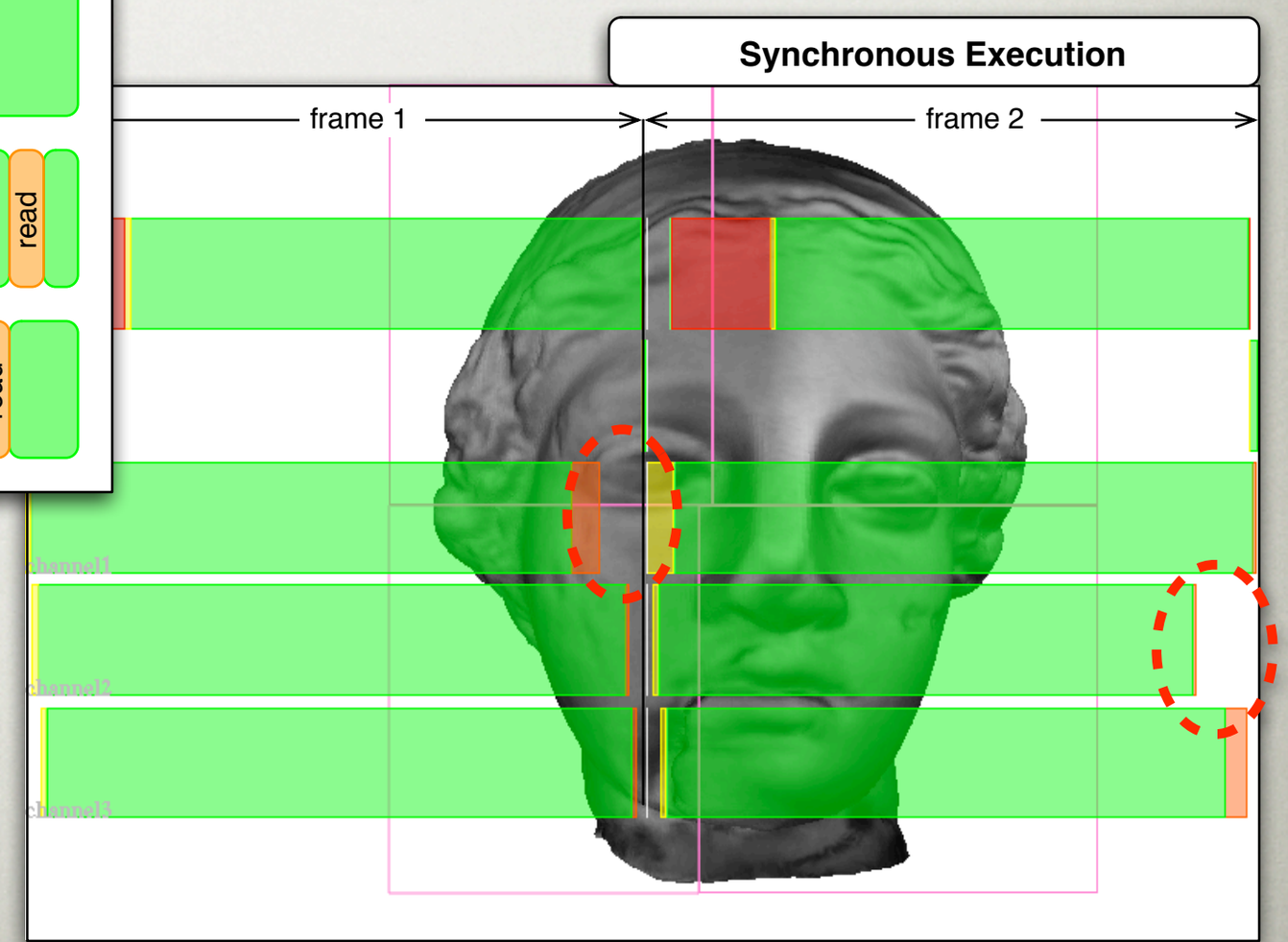
	2D	DB	DPlex	Eye
Fillrate	++	+	++	++
Geometry Processing	0/+	++	++	++
Bandwidth Usage	0/+	++	++	++
GPU Memory Usage	0/+	++	0	0
Loadbalancing	0	+	++	++
Latency	++	++	-	++
IO Requirements	O(1)	O(n)	O(1)	O(n)

Asynchronous Execution

- Source channels start rendering the next frame early
- Hides imbalance in task decomposition
- Only visible channels belonging to the same view need synchronization
- Greatly improves scalability on bigger pipe counts

Asynchronous Execution

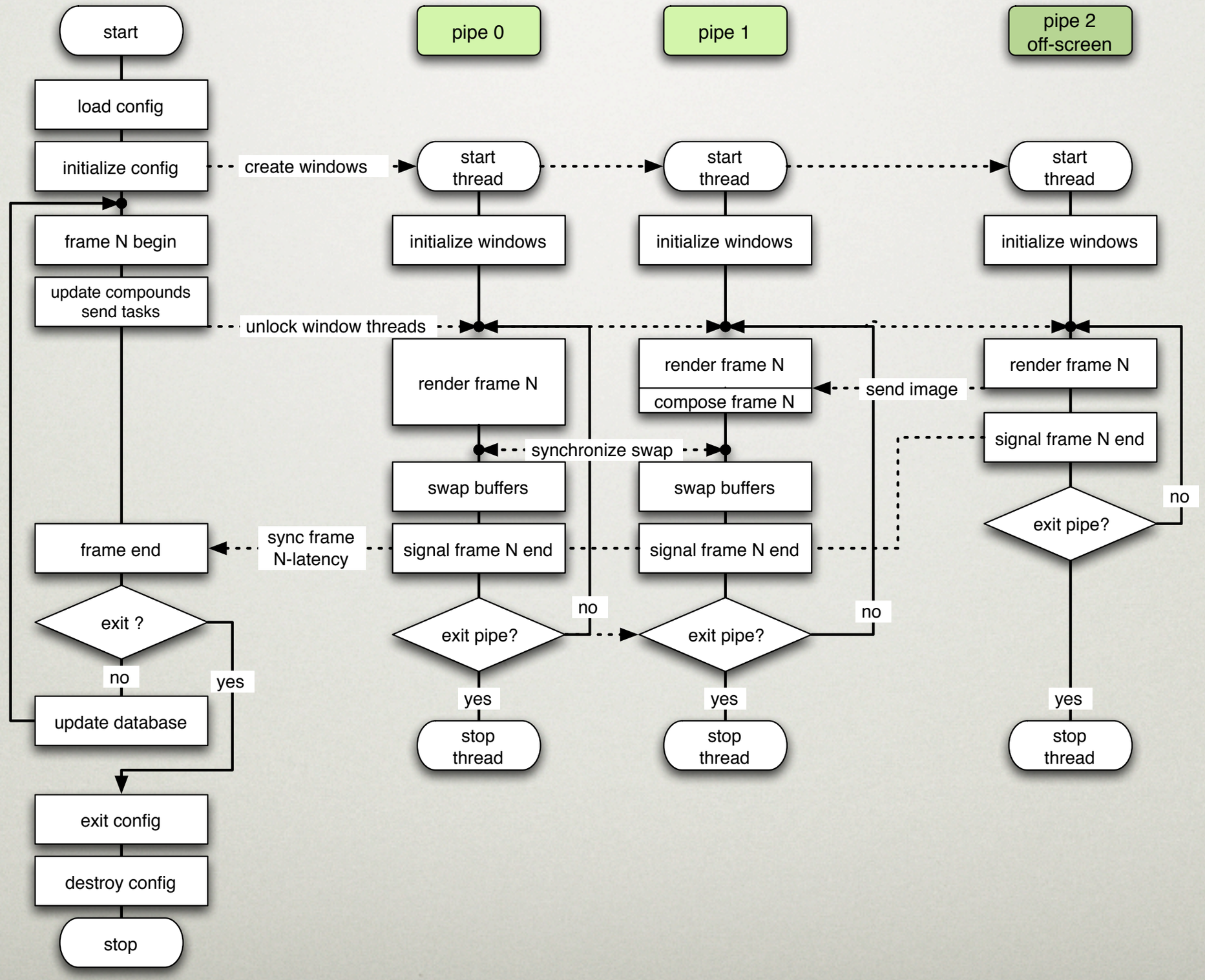




Remote Visualization

- Leverage Knowledge of the Application
  - Frames are often available in Main Memory
  - Additional Frame-Transport Optimisations
  - Remote Workstation is ‘Destination Channel’
- Loadbalancing of multiple Applications on one Central Cluster

Programming Model

- Callback-Driven
  - Init, Exit, Draw and Compound Callbacks
  - Default Callbacks for common Operations
- Flexible Event Handling
  - Continuous Update
  - Event-Driven

Execution Model