Workshop on Parallel Visualization and Graphics

Production Cluster Visualization: Experiences and Challenges

Randall Frank

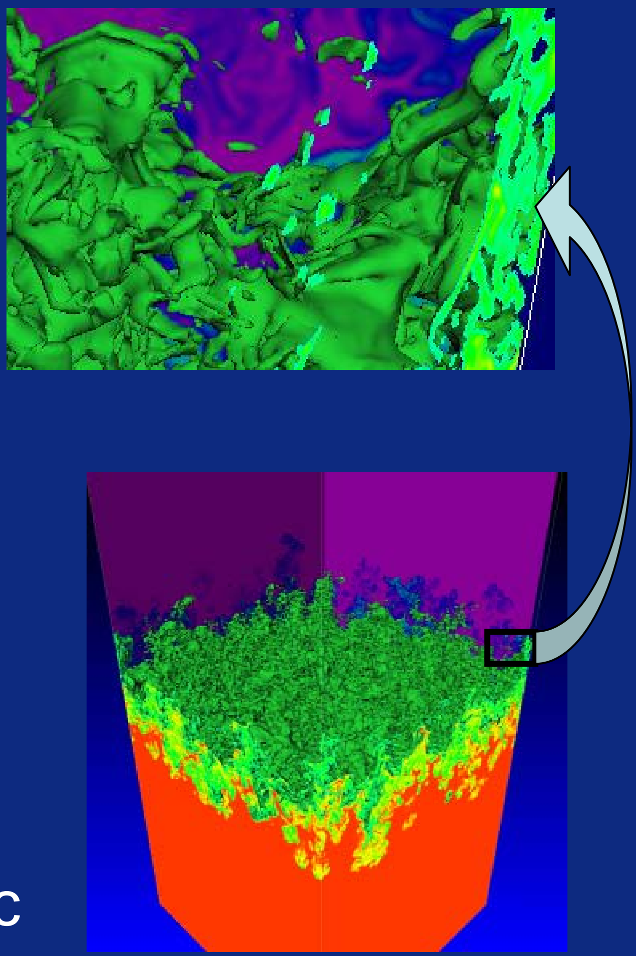
VIEWS Visualization Project Lead

Lawrence Livermore National Laboratory

UCRL-PRES-200237

Why COTS Distributed Visualization Clusters?

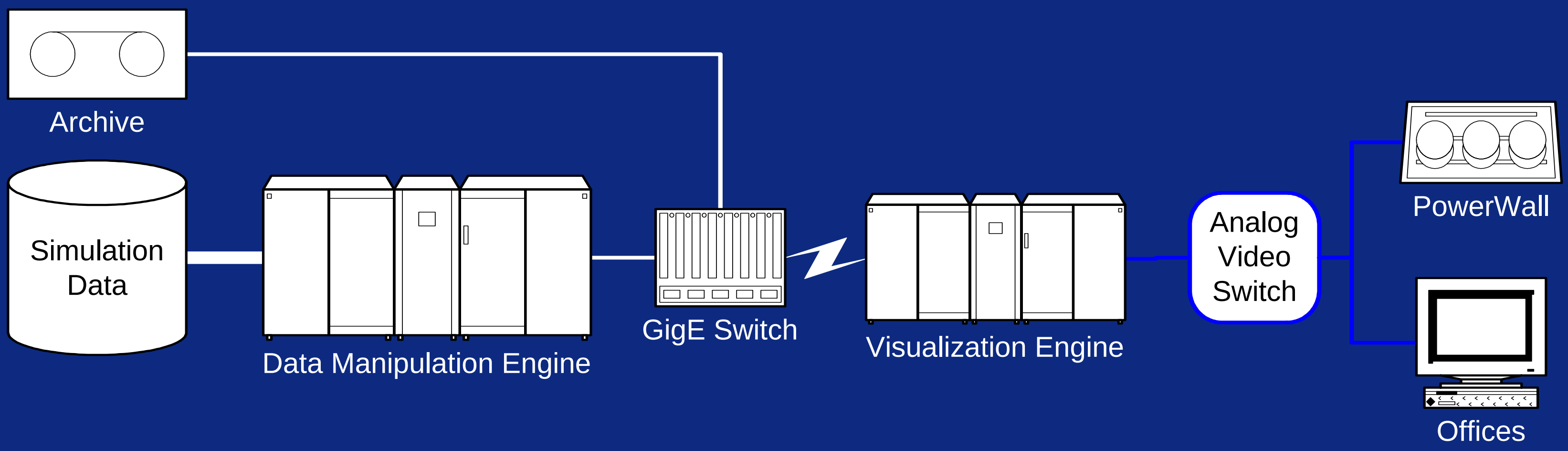
- The realities of extreme dataset sizes (10TB+)
  - Stored with the compute platform
  - Cannot afford to copy the data
  - Co-resident visualization
- Track compute platform trends
  - Distributed infrastructure
  - Commodity hardware trends
  - Cost-effective solutions
- Migration of graphics leadership
  - The PC (Gamers) Desktops
  - Display technologies
    - HDR, resolution, stereo, tiled, etc



Production Visualization Requirements

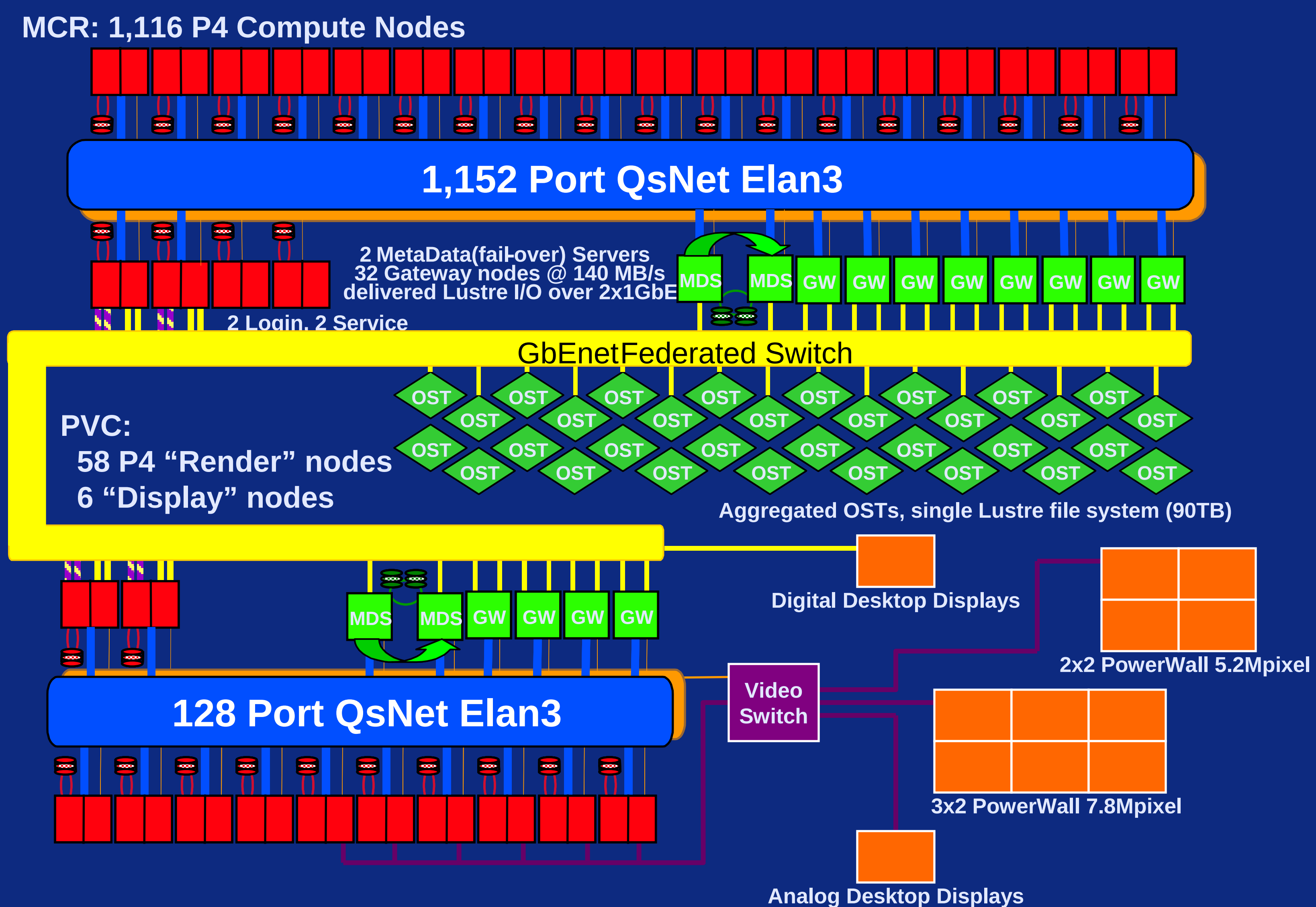
- Unique, aggresive I/O requirements
  - Access patterns/performance
- Generation of graphical primitives
  - Graphics computation: primitive extraction/computation
  - Dataset decomposition (e.g. slabs vs chunks)
- Rendering of primitives
  - Aggregation of multiple rendering engines
- Video displays
  - Routing of digital or video tiles to displays (over distance)
- Interactivity (not a render-farm!)
  - Real-time imagery
  - Interaction devices, human in the loop (latency, prediction issues)
- Scheduling
  - Systems and people

Visualization Environment Architecture



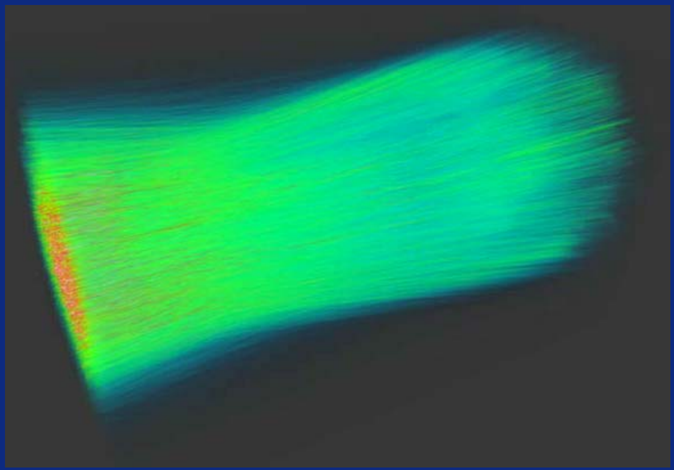
- Raw data on platform disks/archive systems
- Data manipulation engine (direct access to raw data)
- Networking (data/primitives/images)
- Visualization/rendering engine
- Video & remotely rendered image delivery over distance
- Displays (office/PowerWalls)

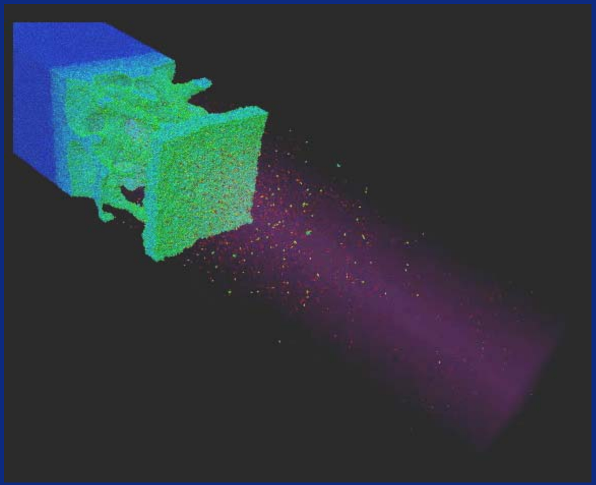
Deployed Environment: MCR & PVC

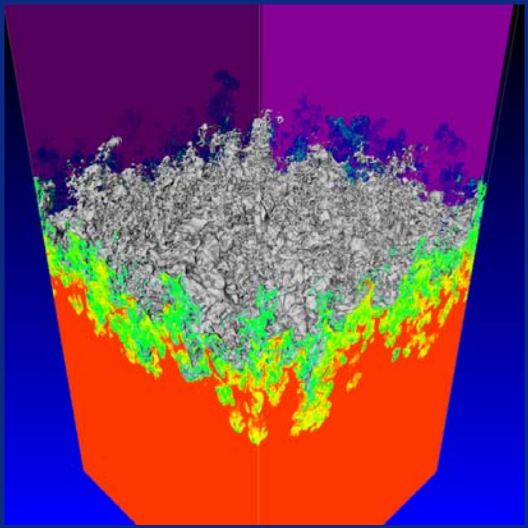


MCR & PVC Success

- MCR+PVC, a prolific tandem…
- Effectively handling multi-TB datasets, many over 10TB
  - Examples: 1152^3 7 variables 1000 timesteps, billion atom
  - Custom data formats: wavelet compression, spacefilling curves
  - Excellent aggregate raw I/O rates (GBs/sec)
- Generating animations: 1/10 the time, 1/5 the resources
- Driving ~8Mpixel PowerWall (3x2)
  - Supporting interaction in a problem solving environment (PSE)

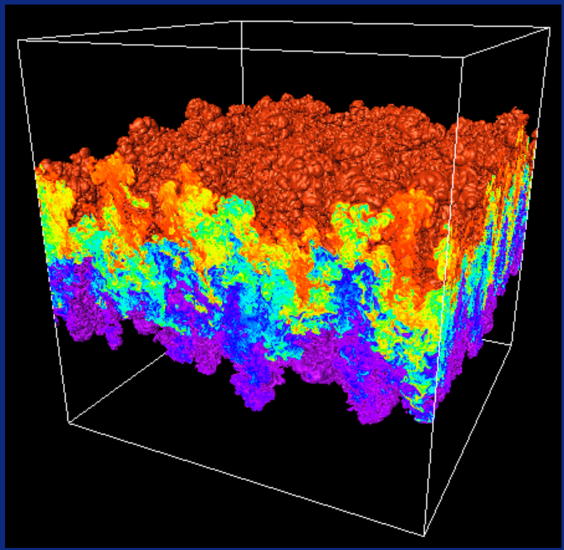






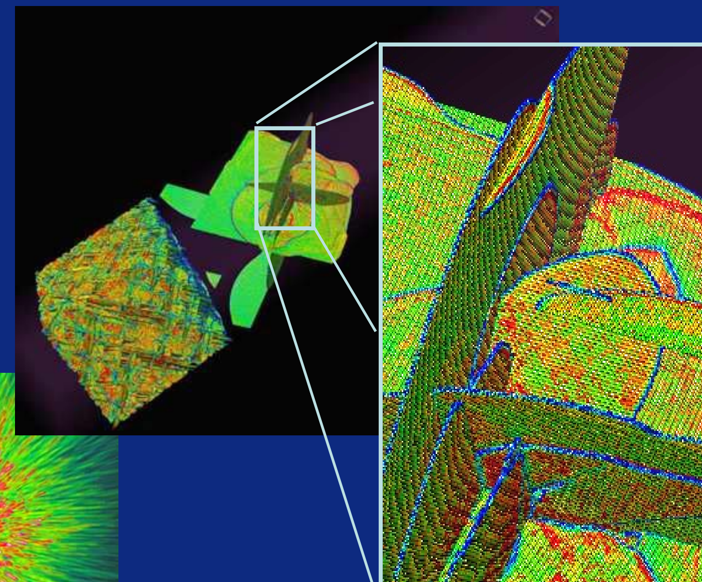
Challenges Remain…

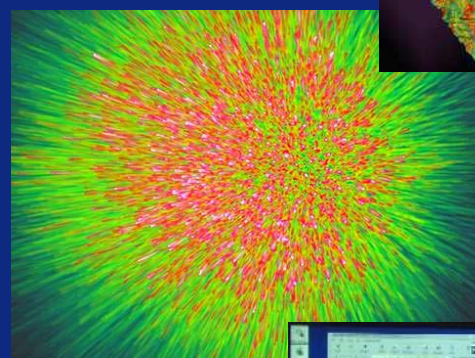
- Fragile system components
- Data access
  - Cluster-wide parallel I/O systems can be fragile
  - Must optimize for viz access patterns
- The impedance mismatch problem
  - Smaller number of nodes for viz
    - Out of core processing
  - Improper data decomposition
- Scheduling complexity
  - Co-scheduling of multiple clusters
  - Combinations of parallel clients, servers, services and displays



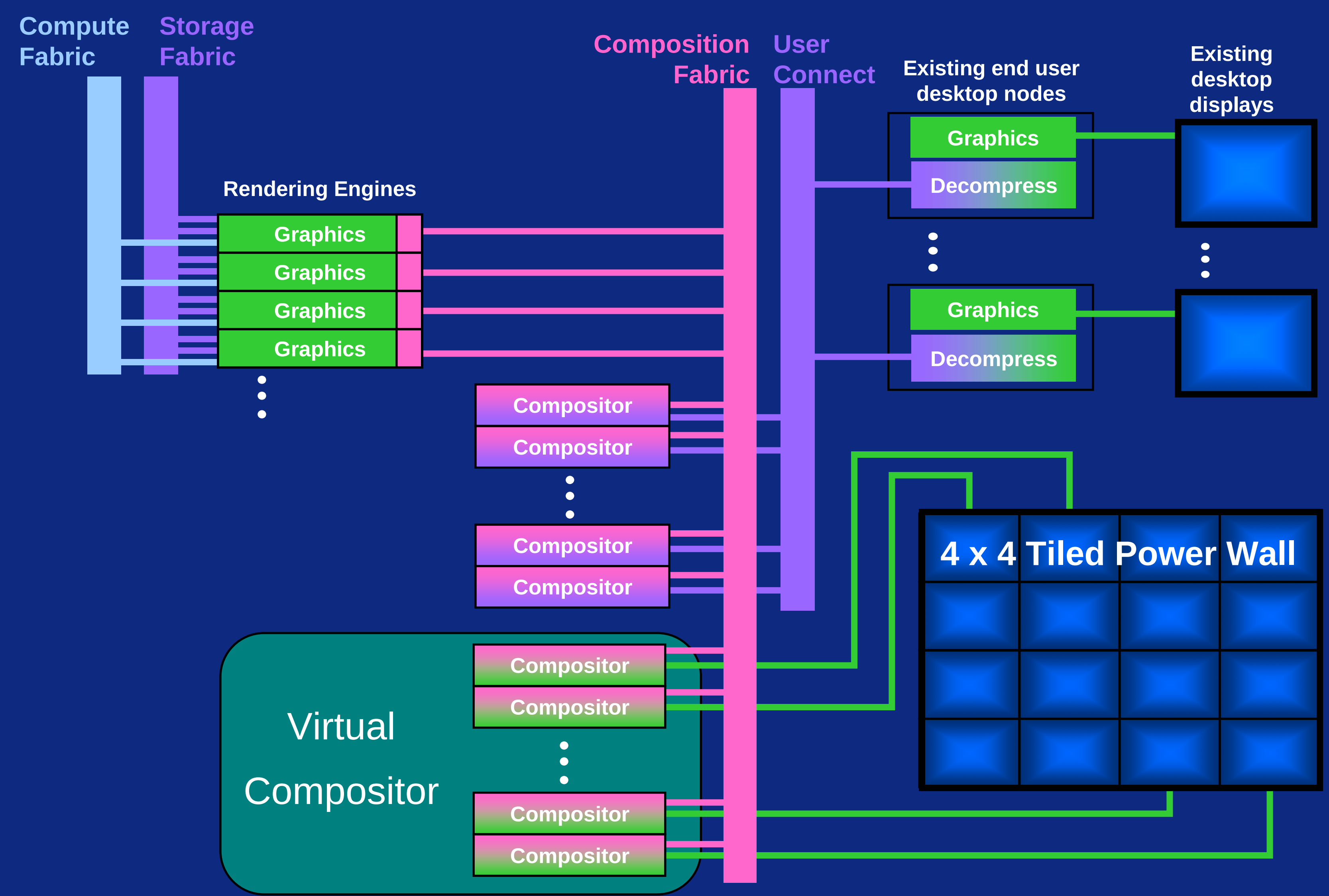
Data Challenges: Large & Complex

- Increased Complexity
  - Increased fidelity
    - Spatial, temporal, higher-order
  - Complex & unique representations
- PSE Integration
  - Multiple data sources
  - Additional context
- Interpretation failure
  - Difficult visual interpretation
    - Artificial polygon limits (90M?)
  - Scalability
    - Failing, global algorithms
    - Fidelity issues



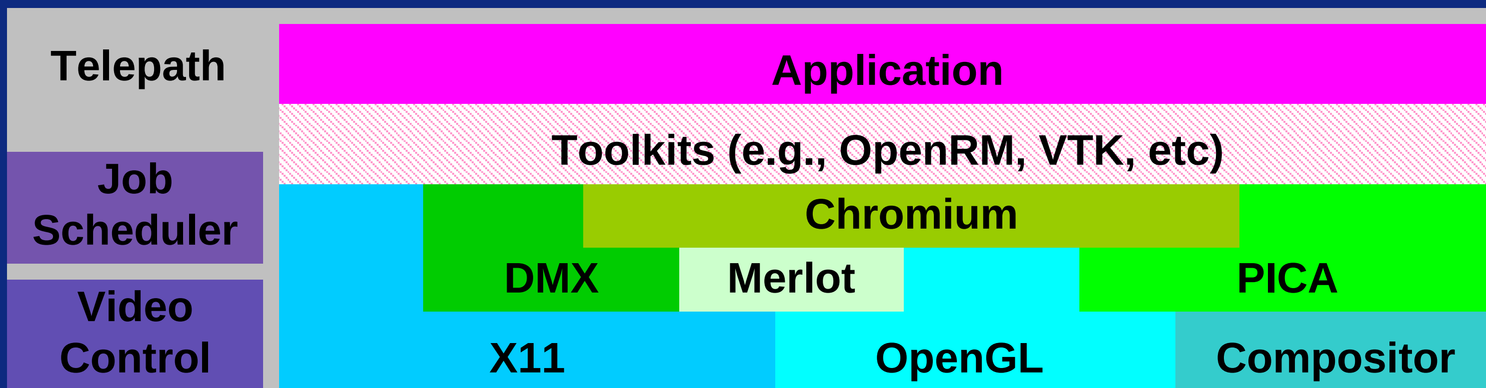


Idealized Visualization Environment



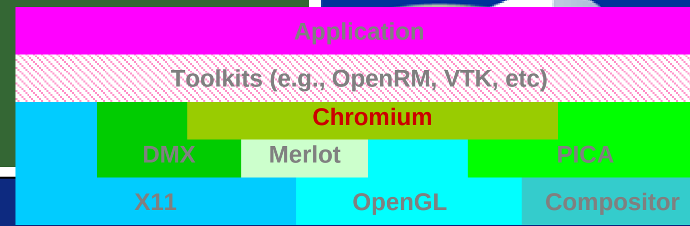
Scalable Visualization Software Stack

- Goal: Provide integrated, distributed parallel services for viz apps. Encourage new apps, increase portability & device transparency.

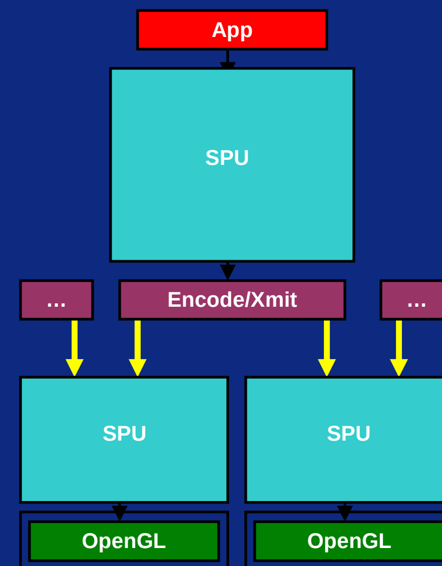


  - Applications: VisIt, ParaView, EnSight, Blockbuster, etc.
  - Toolkits – Parallel visualization algorithms, scene graphs, etc.
  - DMX – Distributed X11 windowing and input devices
  - Chromium – Parallel OpenGL rendering model
  - PICA – Parallel image compositing API
  - Merlot – Digital image delivery infrastructure
  - Telepath – Visualization “session” control and scheduling
  - Core “vendor” services - X11/OpenGL/compositors/NICs/Job control

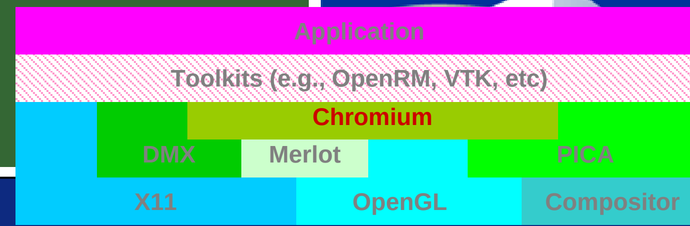
Distributed GL: Chromium



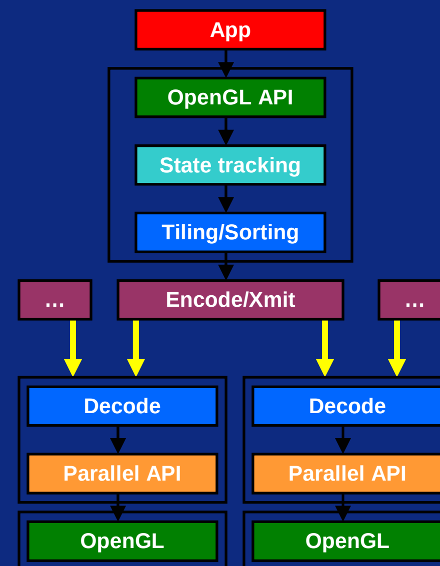
- Distributed OpenGL rendering pipeline. Provides a parallel OpenGL interface for an N to M rendering infrastructure based on a graphics stream processing model.
- The Stream Processing Unit (SPU)
  - “Filter” view OpenGL
  - SPU interface is the OpenGL API
    - Render, modify, absorb…
  - Allows direct OpenGL rendering
  - Supports SPU inheritance
  - Application “translucent”
- Development:
  - chromium.sourceforge.net
  - RedHat/Tungsten Graphics ASCI PathForward
  - Stanford, University of Virginia
  - Stereo, Fragment/Vertex pgms, CRUT, dynamic caching



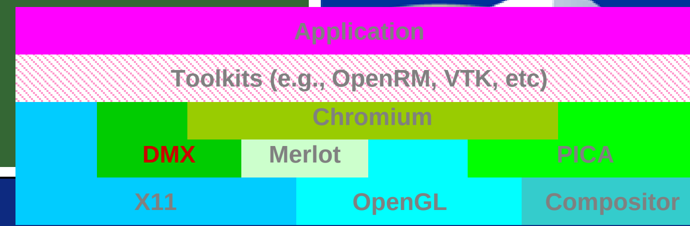
Distributed GL: Chromium



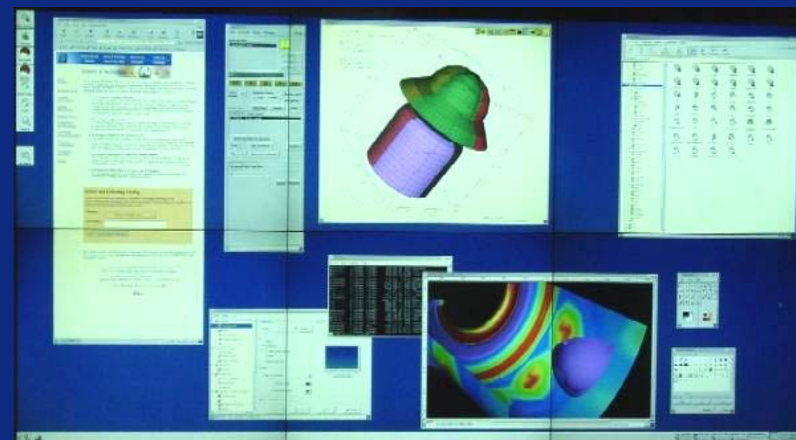
- Distributed OpenGL rendering pipeline. Provides a parallel OpenGL interface for an N to M rendering infrastructure based on a graphics stream processing model.
- The Stream Processing Unit (SPU)
  - “Filter” view OpenGL
  - SPU interface is the OpenGL API
    - Render, modify, absorb…
  - Allows direct OpenGL rendering
  - Supports SPU inheritance
  - Application “translucent”
- Development:
  - chromium.sourceforge.net
  - RedHat/Tungsten Graphics ASCI PathForward
  - Stanford, University of Virginia
  - Stereo, Fragment/Vertex pgms, CRUT, dynamic caching



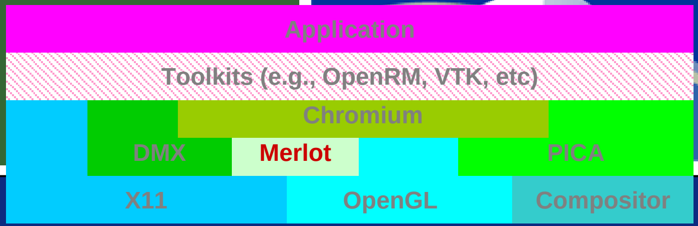
Parallel X11 Server: DMX



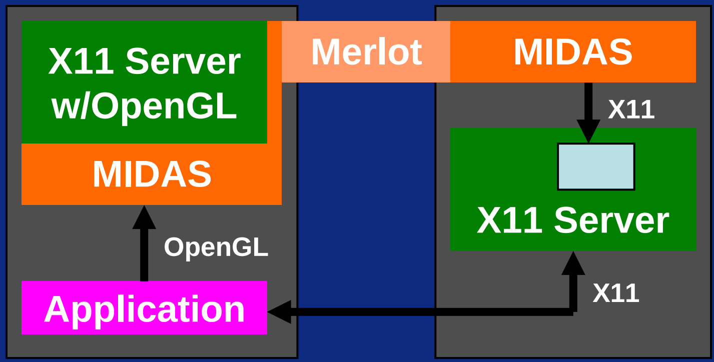
- Distributed multi-headed X server: DMX
- Aggregates X11 servers
  - "Server of servers" for X11
  - Single X server interface
- Accelerated graphics
  - 2D via accelerated X server
    - Common extensions as well
  - Back-side APIs for direct, local X11 server access
  - OpenGL via ProxyGL/GLX (from SGI) or via Chromium SPU
- Development:
  - dmx.sourceforge.net
  - RedHat ASCI PathForward contract
  - Integrated with XFree86



Remote Delivery: Merlot

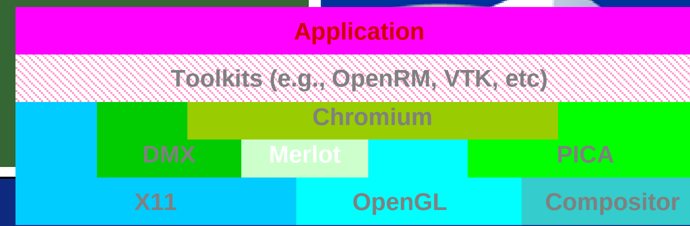


- Merlot is a framework for digital image delivery
  - Transport layer abstraction, Codec interfaces, Device transparency
- MIDAS: Merlot Image Delivery Application Service
  - Indirect OpenGL rendering services for X11 environment
  - Indirect window management
  - Image stream transport

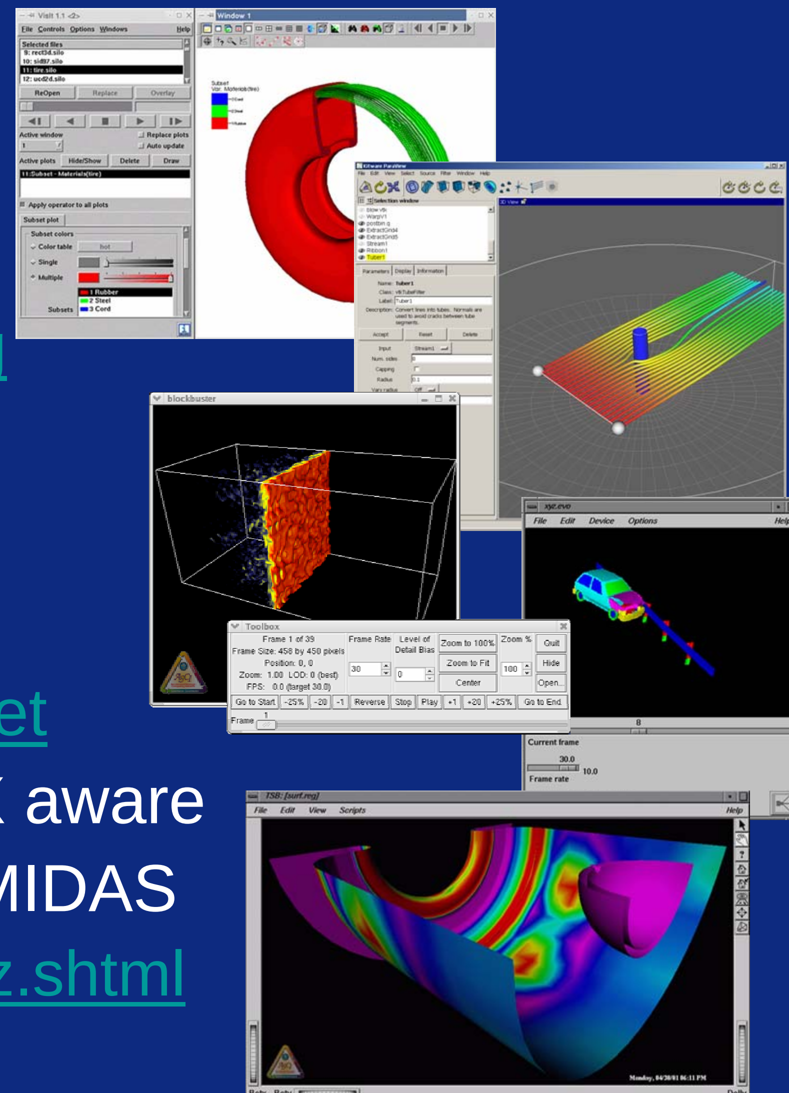


- Development:
  - Alpha released as OpenSource
  - More apps and experimental hardware support

Applications



- Full-featured visualization
  - VisIt: www.llnl.gov/visit
    - VTK, client-server model
  - ParaView: www.paraview.org
    - Parallel VTK viz tool
- Specialty applications
  - Blockbuster
    - blockbuster.sourceforge.net
    - Scalable animations, DMX aware
  - TeraScale Browser/Xmovie/MIDAS
    - www.llnl.gov/icc/sdd/img/viz.shtml

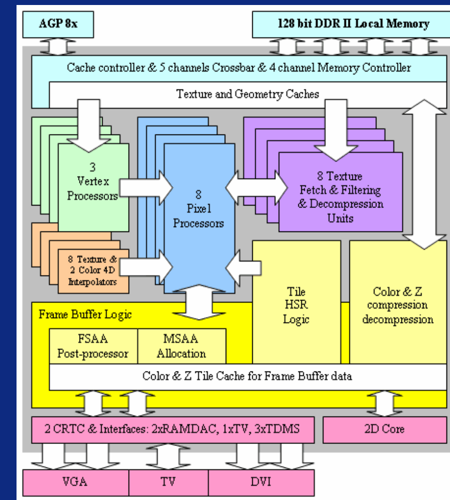


The Road Ahead

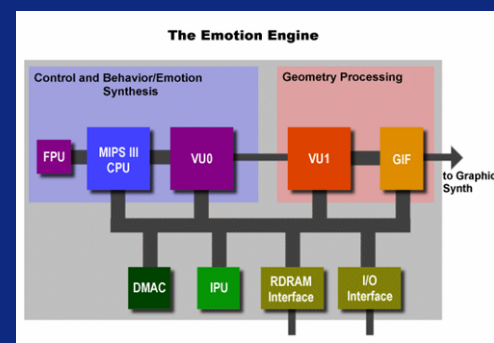
- Improved data understanding and exploitation
  - Basic interactivity is pretty much a done deal
    - Plenty of workable, scalable approaches
  - Picking up to temporal (or other dimensional) challenge
  - Topological based representations
  - How far do we follow the terabyte to gigapixel path?
- Imminent progress
  - Bandwidth increases everywhere: Buses, interconnects, etc
  - Dedicated compositing hardware
  - Second generation digital video technologies: 10gigE, IB, etc
- On the near horizon…
  - Managing software longevity, software & hardware complexity
  - High dynamic range: data to rendering to display
  - The extreme FLOP approach

The NV30 and the Sony Playstation 3

- The nVidia NV30 Architecture
  - 256MB+ RAM, 96 32bit IEEE FP units @ 500Mhz
  - “Assembly language” for custom operations
  - Streaming internal infrastructure
- The PlayStation3 (patent application)
  - Core component is a cell
    - 1 “PowerPC” CPU + 8 APUs (“vectorial” cpus)
    - 4GHz, 128K RAM, 256GFLOP/cell
    - Building block for multimedia framework
  - Multiple cells
    - Four cell architecture (1TFLOP)
    - Central 64MB memory
    - Switched 1024 bit bus, optical links?
- Effective streaming data access?
- New rendering abstractions (are polygons dead?)



nVidia NV30



Sony PS2 “Emotion” Engine

The Streaming Programming model

- Streaming exposes concurrency and latency at the system level as part of the programming target
- Data moves through the system: exposed concurrency
  - Avoid global communication: prefer implicit models (e.g. Cr)
- Memory model: exposed latency/bandwidth
  - Scalable, must support very small footprints
  - Distributed, implicit flow between each operation
- A working model:
  - Computational elements + caching and bandwidth constraints
  - External “oracle” for system characterization and realization
- Goals:
  - Optimally trade off computation for critical bandwidth
  - Leverage traditionally “hidden” programmable elements

Next Generation Streaming Cluster…

- Modified visualization algorithms
  - Cache oblivious: local, at the expense of computation
  - “Non-graphical” algorithms, moving away from polygon primitives
  - Digital, high dynamic range imagery from generation to display
- Computation and memory caches everywhere
  - NICs, Drive controllers, Switches, TFLOP GPUs + PCI express
  - Utilizing them may require a disruptive programming shift
    - Exposed data staging and streaming issues
- What are the appropriate abstractions?
  - New languages with higher levels of abstractions?
    - Run-time “realization”, dynamic compilation and scheduling
  - Glue languages: e.g. “shader” languages?
  - Graphics or other API extensions/encapsulation?
  - How do we preserve our largest investments, people & software?

Auspices:

- This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

- This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48 and was partially supported by Sandia National Laboratories. Sandia is a multi-program laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.