Silicon Graphics, Inc.

Silicon Graphics Prism™ : A Platform for Scalable Graphics

Presented by:

Alpana Kaulgud

Engineering Director, Visual Systems

Bruno Stefanizzi

Applications Engineering

sgi®

Overview of Talk

- Goals for Scalable Graphics
- Scalable Architecture for Silicon Graphics Prism
- Case Study
- Call to Action and Future Directions

Goals

Traditional Computational Problems (CFD, Crash, Energy, Crypto, etc.)

- Determine problem set size
- Size compute server to solve problem in needed timeframe

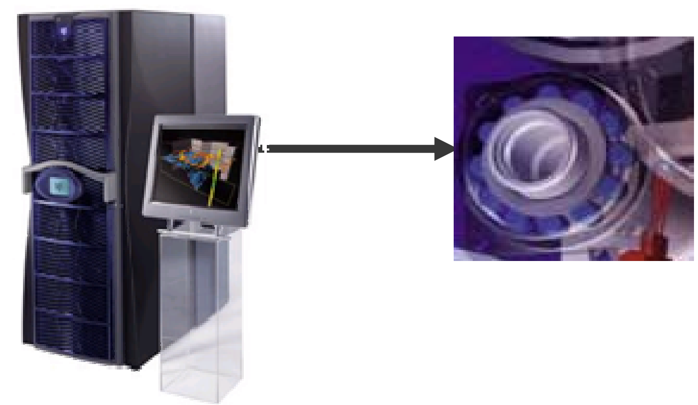
Small Problems: 1 – 16 CPUs
Bigger Problem: 16 – 64 CPUs
Large Problem: 64 – 1024 CPUs
Scientific Challenge: 1024 CPUs or more

Applications scale to use all computational resources: CPU, memory, I/O to reduce time to solution

Traditional Visualization Problems (Media, CAD, Energy, Biomedical, etc.)

- Determine problem set size
- Reduce problem until it fits on a single GPU

Small Problems: 1 GPU
Bigger Problem: 1 GPU
Large Problem: 1 GPU
Scientific Challenge: up to 16 GPUs



Images courtesy of Pratt and Whitney Canada and Landmark Graphics

Goals

Use the appropriate resource for each algorithm in the workflow to reduce “time to solution”

CPUs for computation/visualization

GPUs for visualization/computation

FPGAs for algorithm acceleration

Scalability Dimensions

-Display

-Data

-Render (Geometry/Fill)

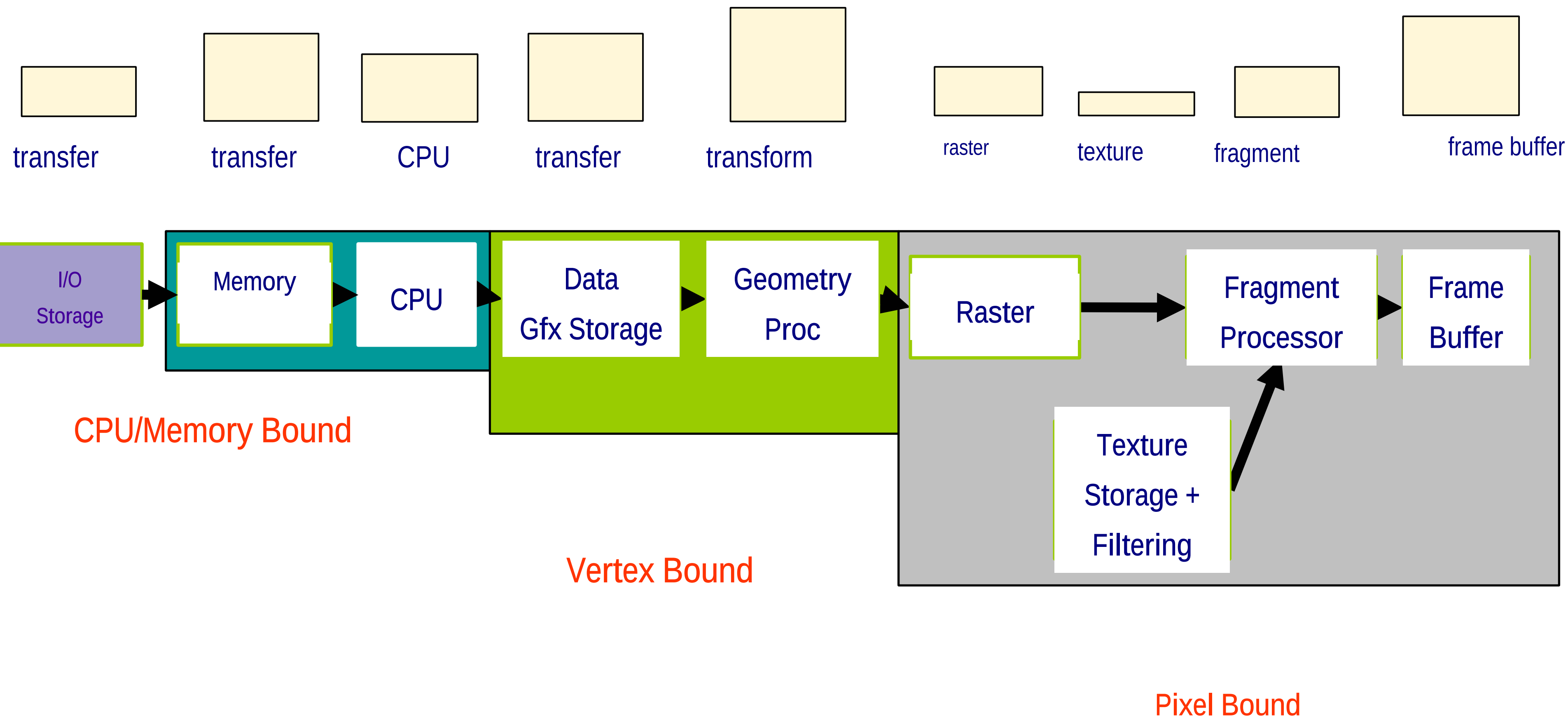
-Number of User/Input Devices

Single System Image

–Ease of Use

Images courtesy of Pratt and Whitney Canada and Landmark Graphics

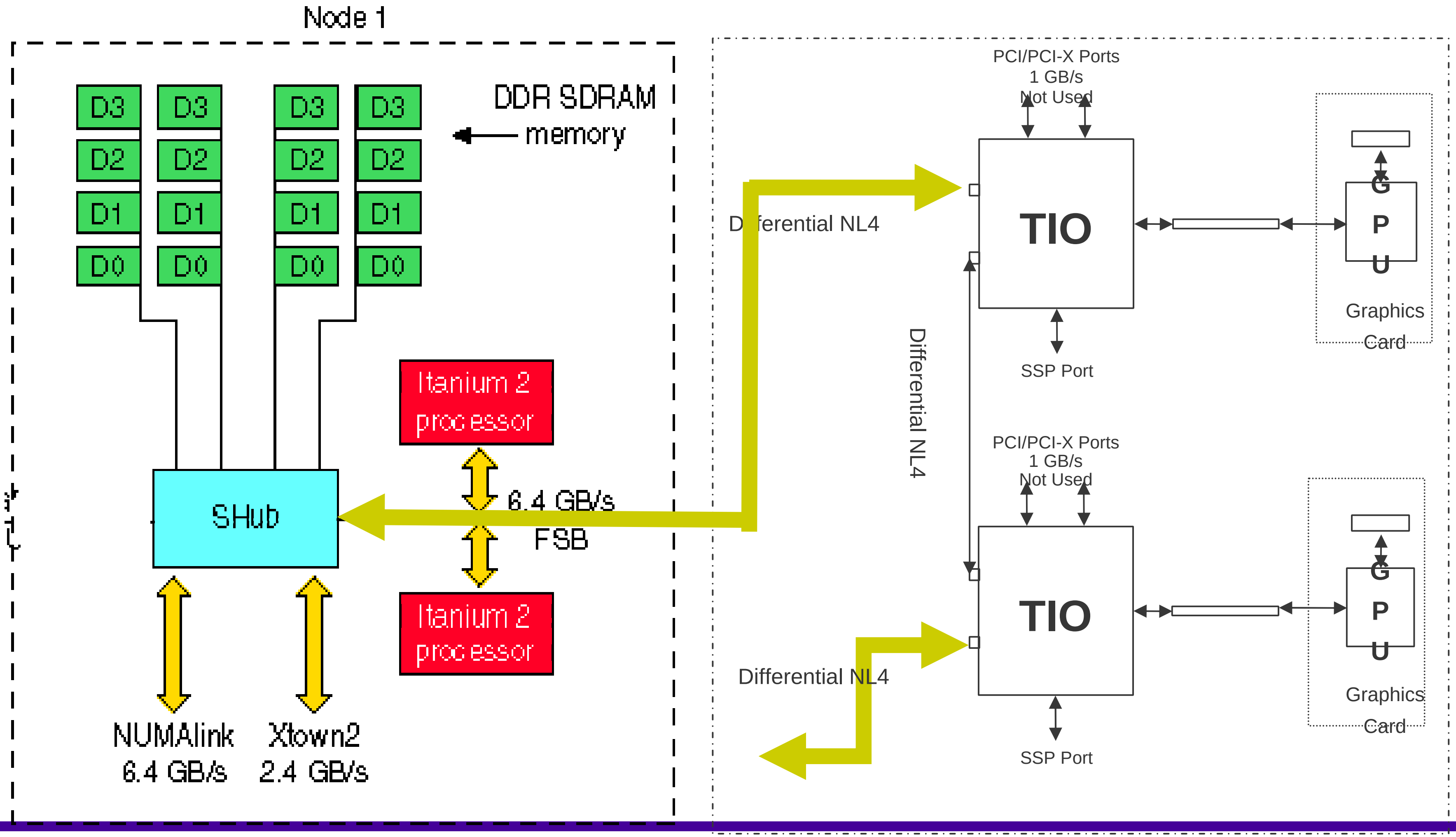
Lets Follow the Data – where are the bottlenecks ?



Scalability General Principles

- **Localize access**
  - Defined by network and topology
  - NUMA principles apply well
- **Pipeline and Parallelize**
- **Minimize locking and synchronization points**
  - **Finer granularity locks**

Components (Render Fast)



Visualization System for Linux®
Software (render smart)

- OpenGL Performer™
- OpenGL Volumizer™
- OpenGL Multipipe™ SDK
- OpenGL Multipipe™
- OpenGL Vizserver™ and Visual Area Networking (VAN)

Invasiveness

OpenGL Performer

OpenGL Volumizer

OpenGL Multipipe SDK

OpenGL Multipipe

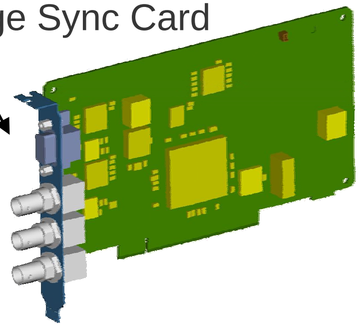
Scalability

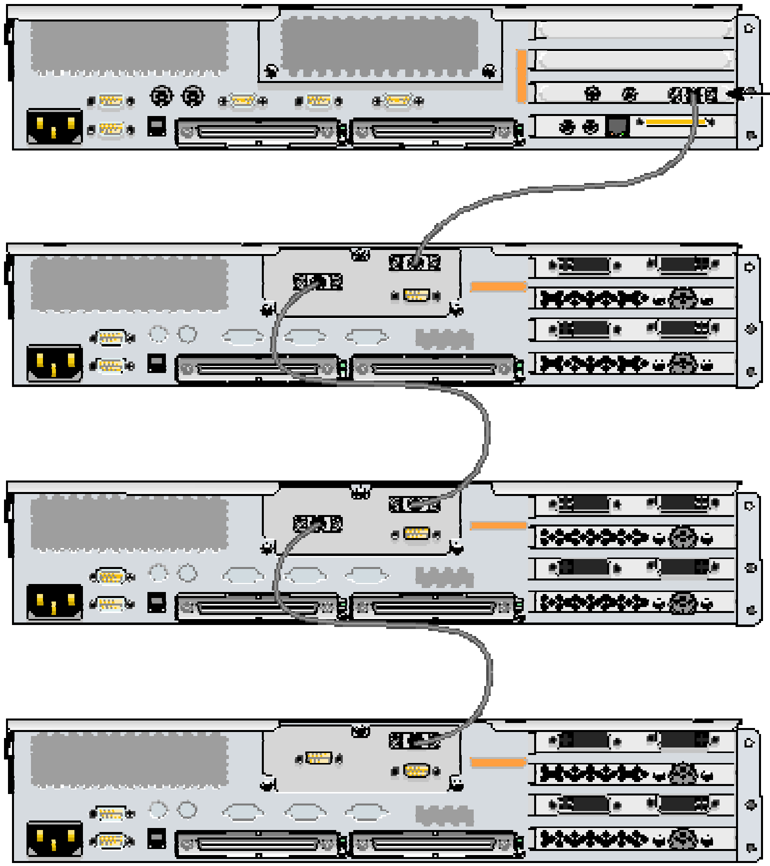
CT data courtesy of University of N. Carolina, image courtesy of EPL Productions

Image Synchronization : Minimize Synchronization Points

- Silicon Graphics Prism offers true backend synchronization through Image Sync – key to scalable platform – architecture does not impose application level rendezvous points

Image Sync Card





ImageSync Features:

•True Framelock capability (genlock with compositor)

•True Swapready capability

•Can be used to lock to internal and external swap and video sync signals.

Solving the Memory Bottleneck

Memory Addressibility - more address bits

- Intel Itanium® II 50-bits: 128000GB
- AMD 64 Opteron® 40-bits: 128GB
- Intel Xeon® 36-bits: 8GB

Memory Bandwidth & Memory Contention

Memory & process placement (lessons learned from HPC)

- Scalability inhibitors
  - False sharing
  - Non local data references
  - Memory contention
- Scalability Enabler Tools
  - Careful code and memory organization
  - Must Run (lock processes to nodes) and default memory placement – First Touch
  - Round Robin placement

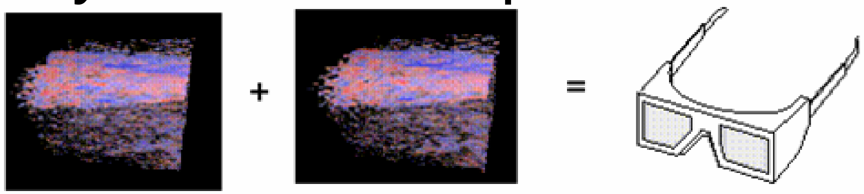
Solving the Rendering Bottleneck

1. Screen-based decomposition

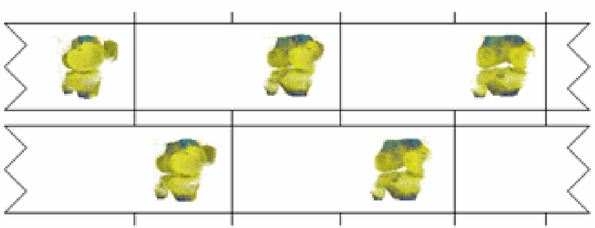
<u>Even more powerful in combination</u>

All modes can be used separately or combined in any number of ways

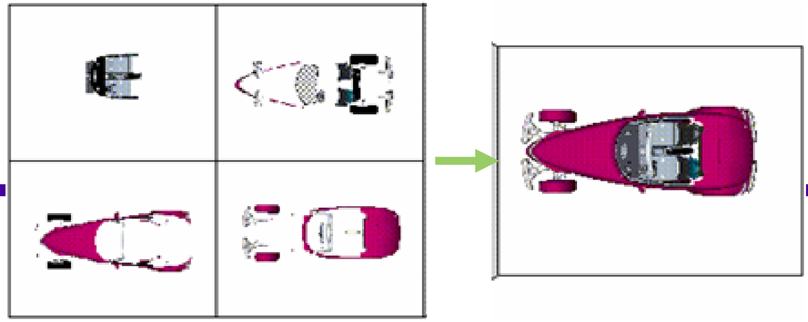
2. Eye-based decomposition

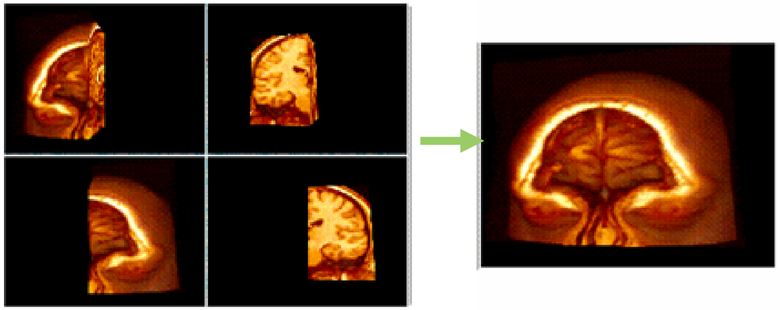


3. Time-based decomposition



4. Data-based decomposition





Data courtesy of DaimlerChrysler, Images courtesy of MAK

Visible Human public data set

Solving the Rendering Bottleneck

Silicon Graphics Prism is capable of all of these modes and more – hybrid modes

- Fixed composition in hardware or more flexible software composition schemes
- Capable of adaptive composition schemes
- Capable of hybrid composition schemes
- Bisection Bandwidth is an important consideration

Case Study

Challenge : Make the visualization of a large model **interactive** using scalability into an application

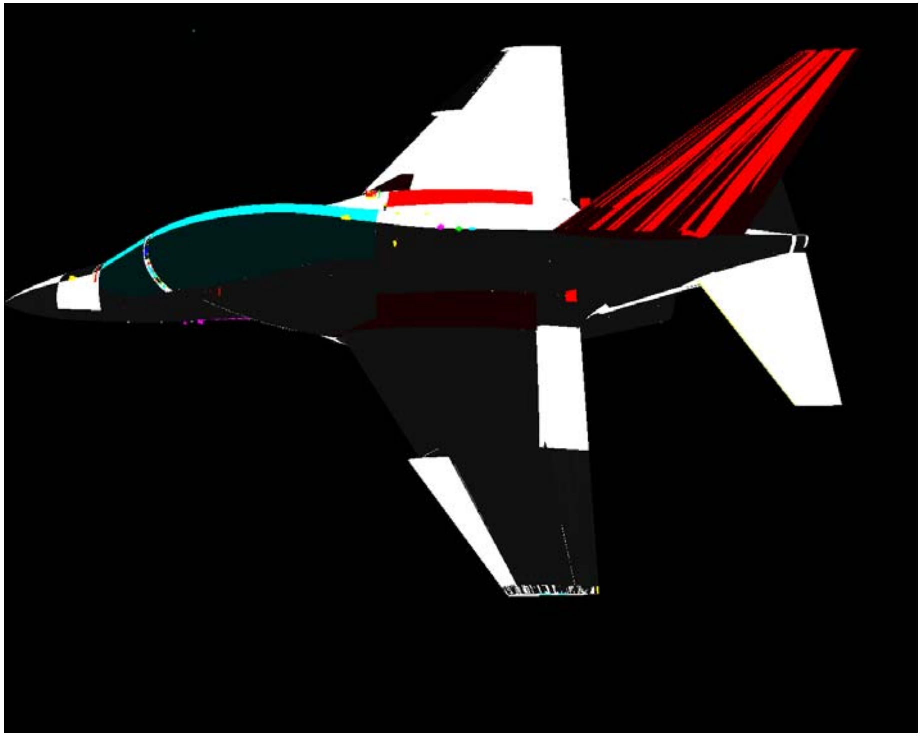
- Model Aermacchi M346*
  - 30Millions non optimized triangles
  - No interactive performance with **< 1Hz** on 1 GPU
  - Around 25000 individual parts
  - No reduction of the problem size
- Technologies used
  - SGI Prism NUMA Multi CPU/GPU
  - OpenSceneGraph
  - OpenGL Multipipe SDK
  - OpenGL Performer

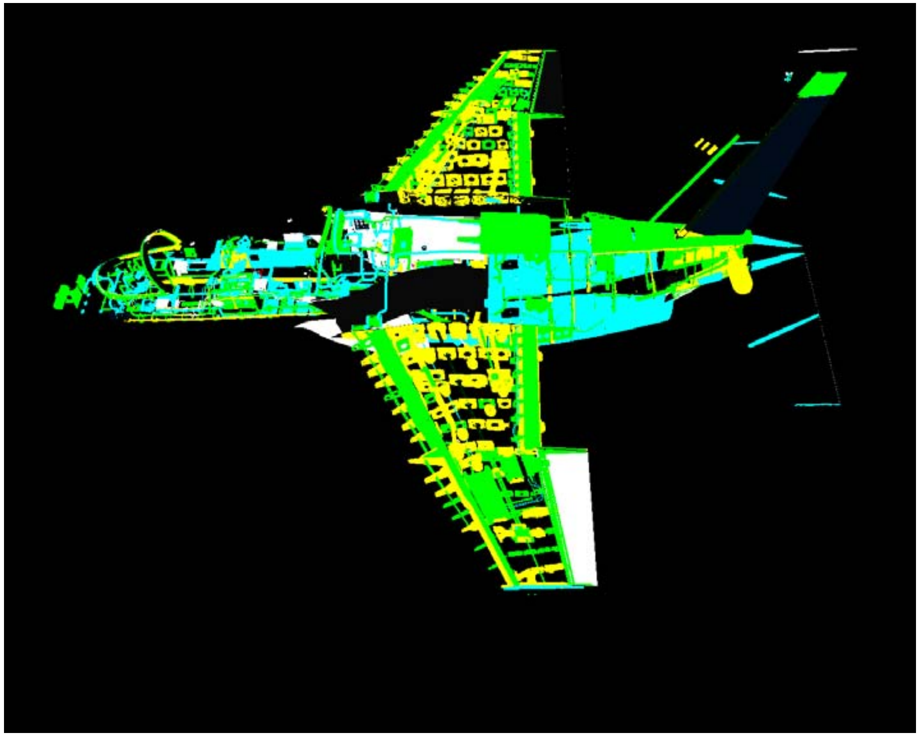
*Courtesy of Aermacchi

Case Study

Model Aermacchi M346*

*Courtesy of Aermacchi





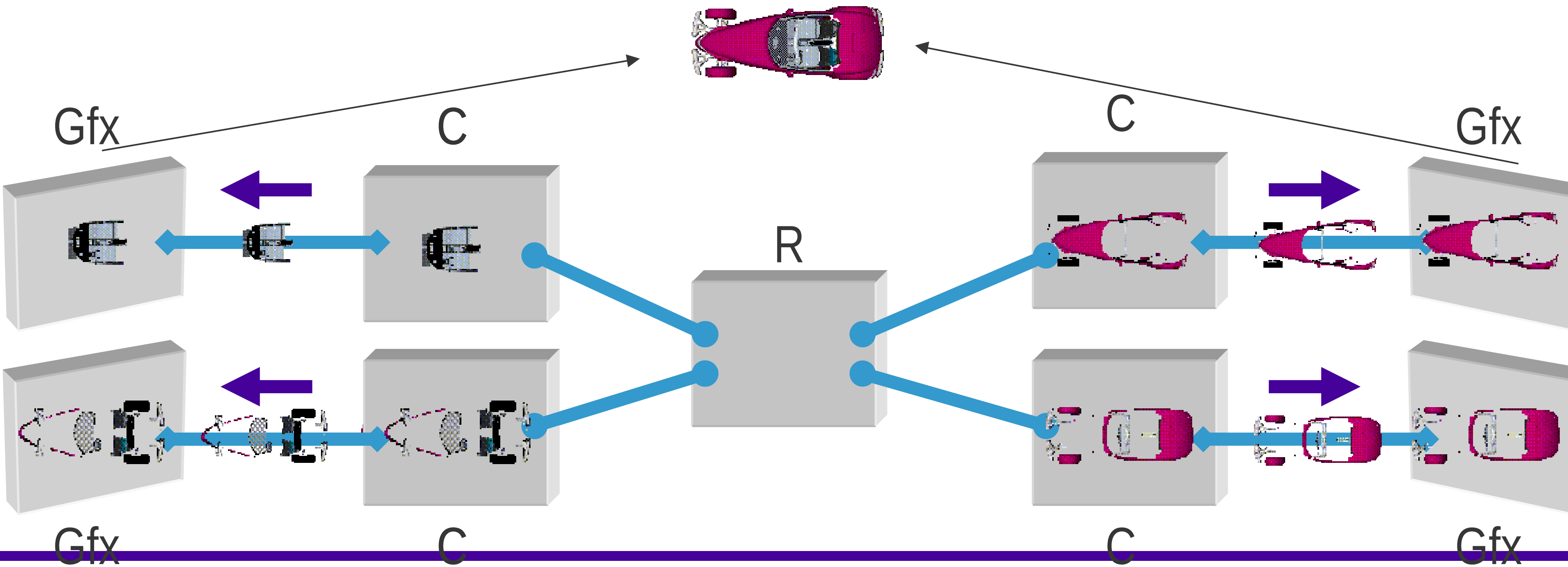
Solving the GPU bottleneck

Using Database decomposition to scale the rendering

- Scale in graphic memory to achieve ‘super scaling’
- Scale in CPU to GPU communication
- But compositing is expensive

C : cpu brick

R : router brick



Solving the GPU bottleneck

Optimizing the compositing phase with a large number of GPUs

- Basic serial GPU compositing
- Stripped GPU compositing for parallel GPU
- CPU compositing

The Read and Draw pixels (color and z) associated for p GPUs

$$C = R_C + R_z + (p-1)(D_c + D_z) \xrightarrow{p \to \infty} \infty$$

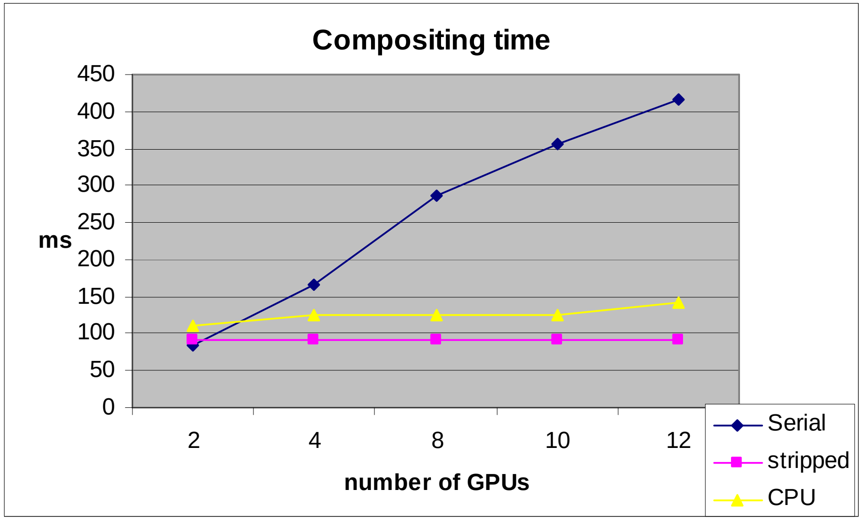
$$C = \frac{(R_c + R_z)(p-1)}{p} + \frac{(D_c + D_z)(p-1)}{p} + \frac{(R_c + R_z)}{p} + \frac{(D_c + D_z)(p-1)}{p} \xrightarrow{p \to \infty} R_C + R_z + 2(D_c + D_z)$$

$$C = R_C + R_z + CPU + D_c + D_z \xrightarrow{p \to \infty} Cte$$

Solving the GPU bottleneck

The compositing phase

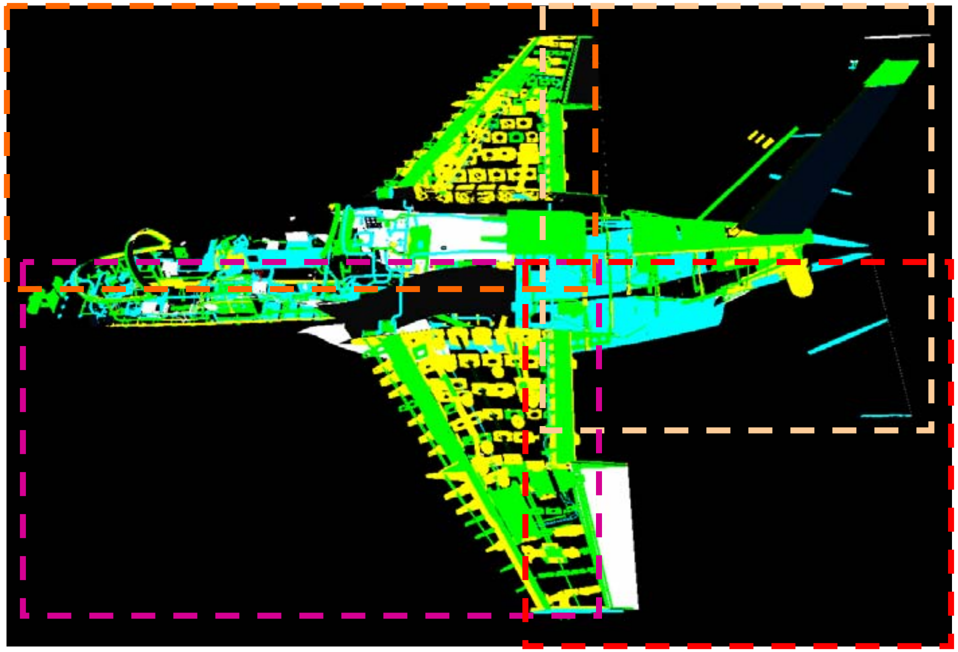
- Basic serial GPU was the bottleneck
- Stripped GPU reduces the bottleneck to almost constant
  - but read/draw smaller is not efficient
- CPU is constant



Solving the GPU bottleneck

Optimizing the compositing phase

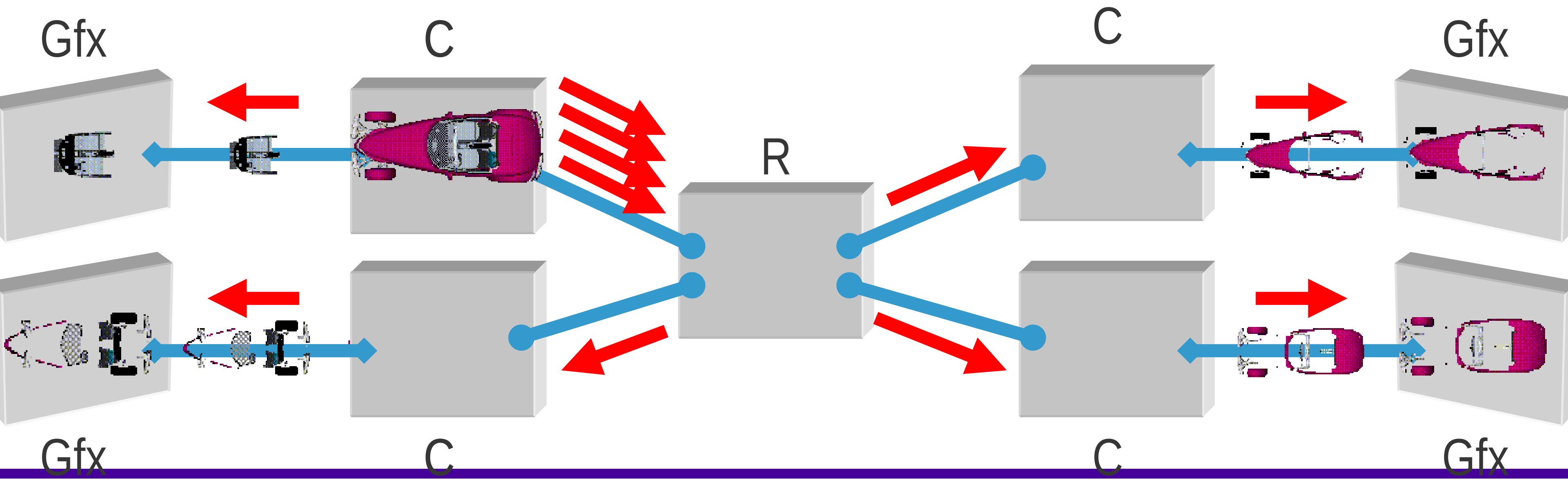
- Reducing the Read and Draw pixels areas
  - Octree to spatialize the model



Solving the Memory bottleneck

In large data visualization, Memory is the bottleneck

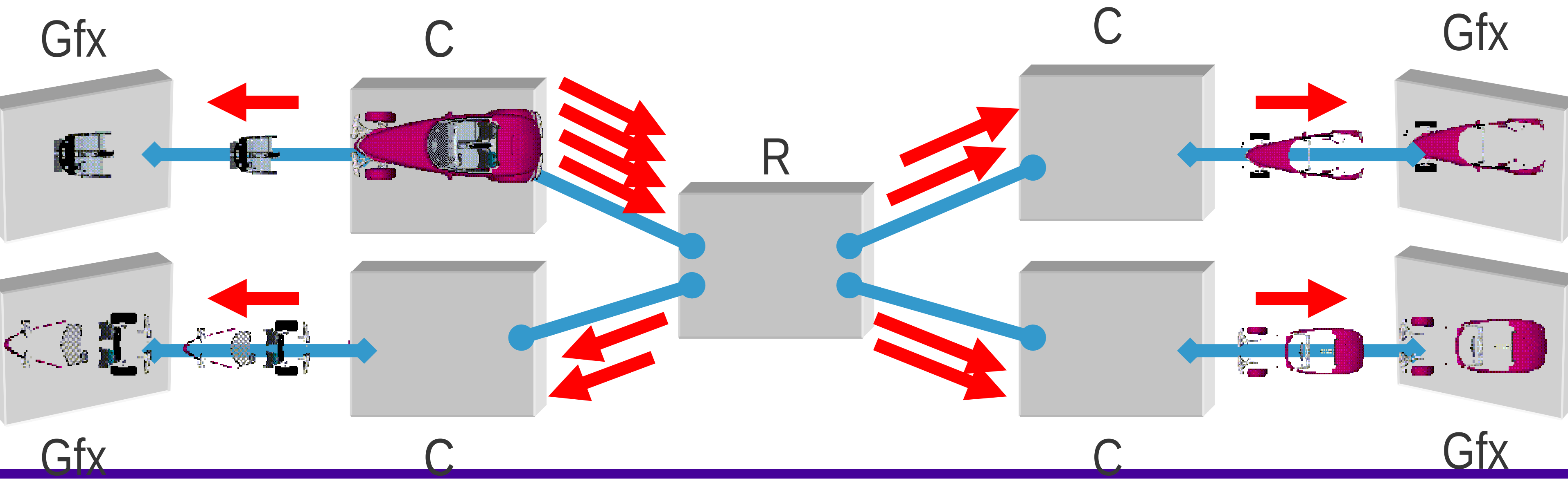
- Traversing and culling the data is expensive
- Especially if the data is all located at the same place!
- Memory placement is important as well as understanding the system topology



Solving the Memory bottleneck

Traversing and culling the data avoiding the memory bottleneck

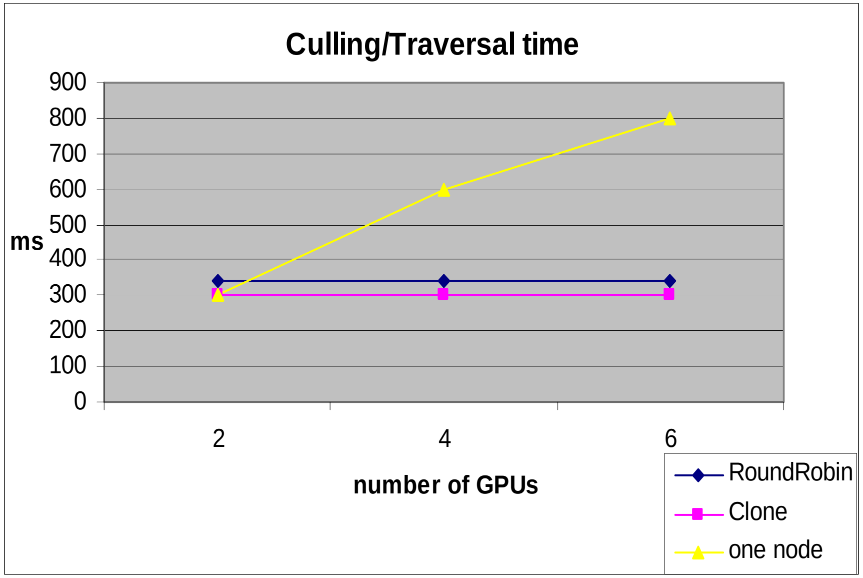
- Parallelize the traversal/culling and the draw
  - More CPU are busy
  - Make things more memory intensive



Solving the Memory bottleneck

Memory placement without changing the data

- Duplicate the database on each node
  - Difficult to maintain for an application doing editing
  - Memory consuming
- Domain decomposition
- System level Round robin with numa tools of Prism

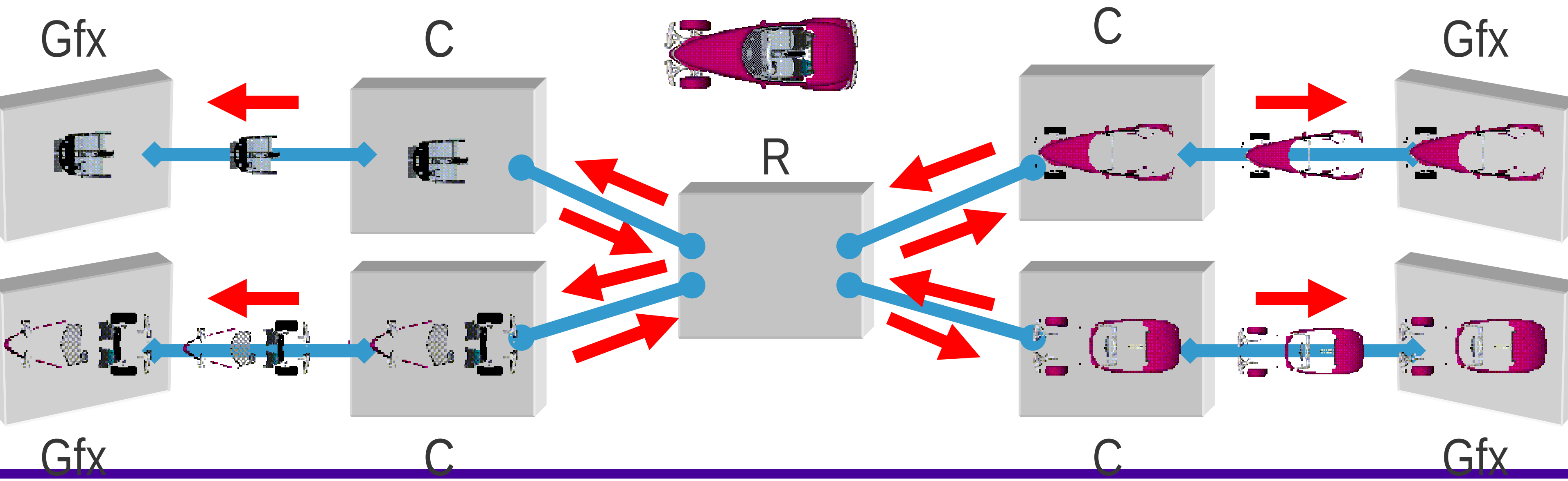


Solving the CPU bottleneck

In large data visualization, Memory is the bottleneck

Balancing the data with NUMA tools in order to get

- Less memory contention
- More bandwidth

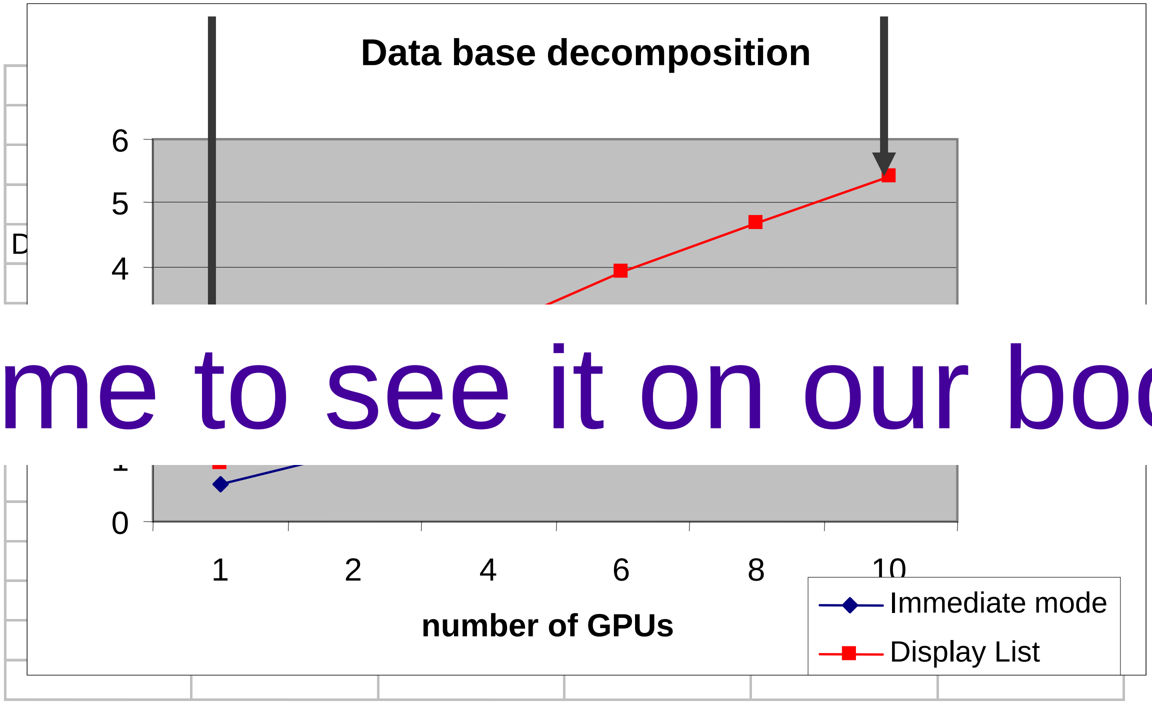


Case Study

Scalability results

Static !!

Interactive!!



Come to see it on our booth!

Call to Action

- Open Standards – continue to support and promote
- Build latency tolerant components, i.e. deeper pipelining
- Virtualization of resources

Future Work

- Real-time options
- Integration with digital media
- Multi-core, Multi-GPU, Multi-everything
  – More “render smart”
  – Hybrid schemes

SILICON GRAPHICS | The Source of Innovation and Discovery™