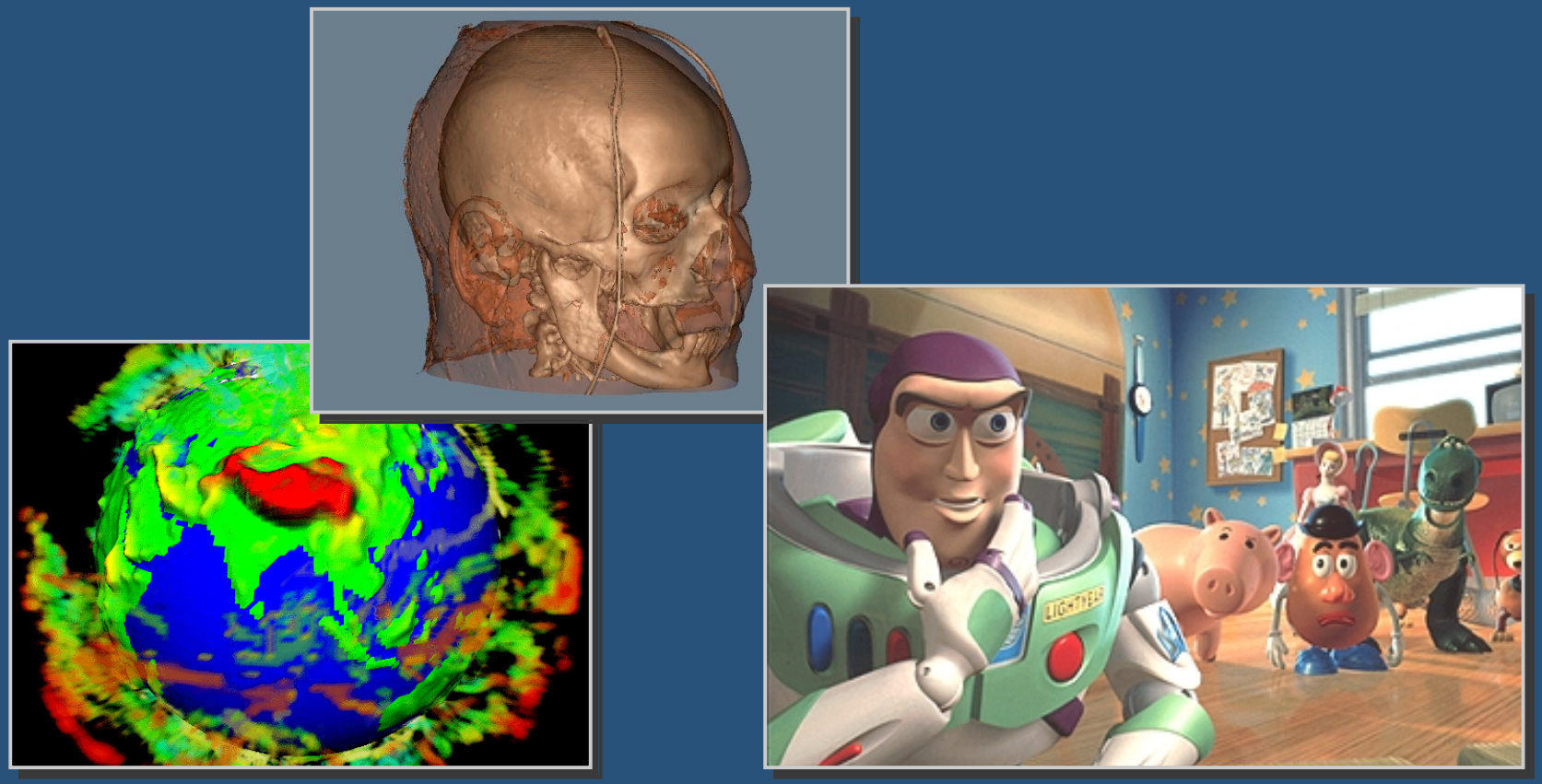
Hybrid Sort-First and Sort-Last rendering with a Cluster of PCs

Rudrajit Samanta
Thomas Funkhouser
Kai Li and Jaswinder Pal Singh

Princeton University

Motivation : Parallel rendering

- Large data sets, details, and realism



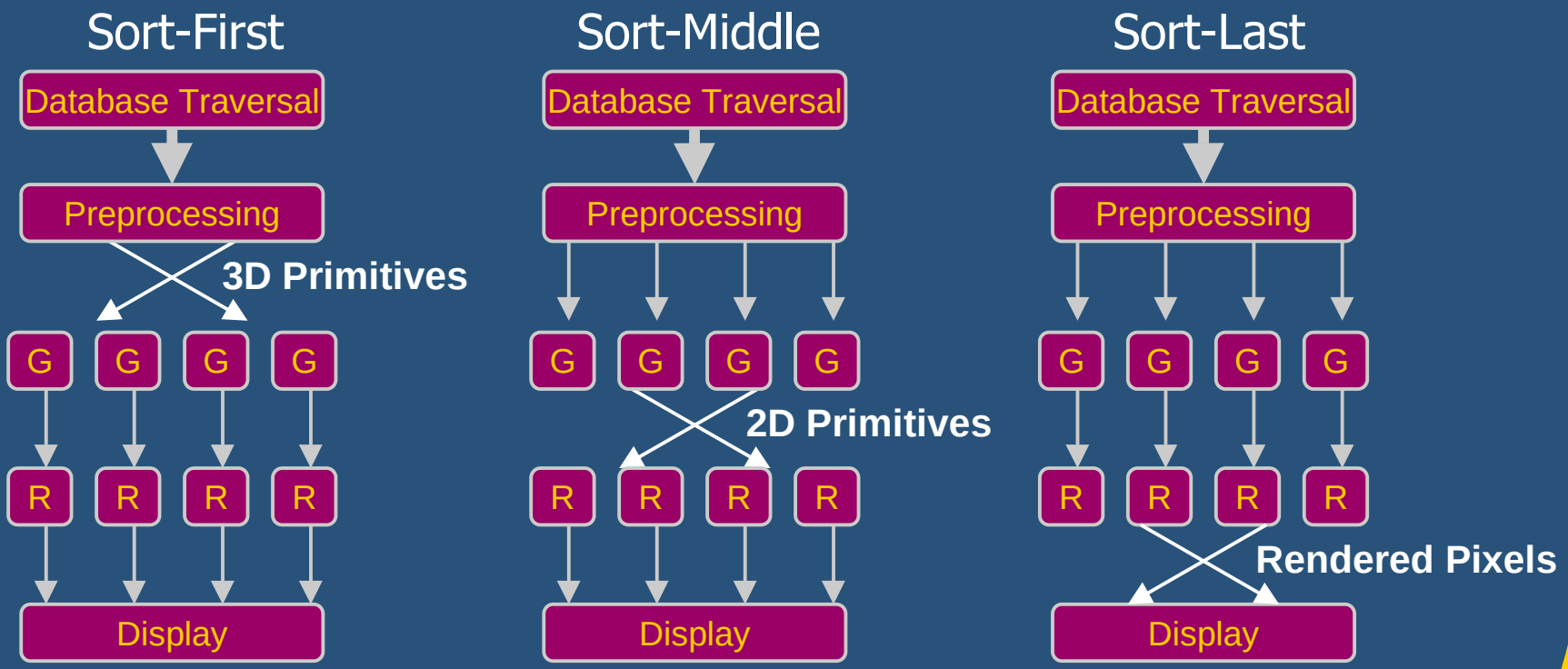
Motivation : PC Clusters

- Low cost
- Tracks technology curve
- Modular and flexible
- Scalable capacity

Outline

- Motivation
- System Architecture
- Algorithms
- Simulation Results
- Conclusion and Future Work

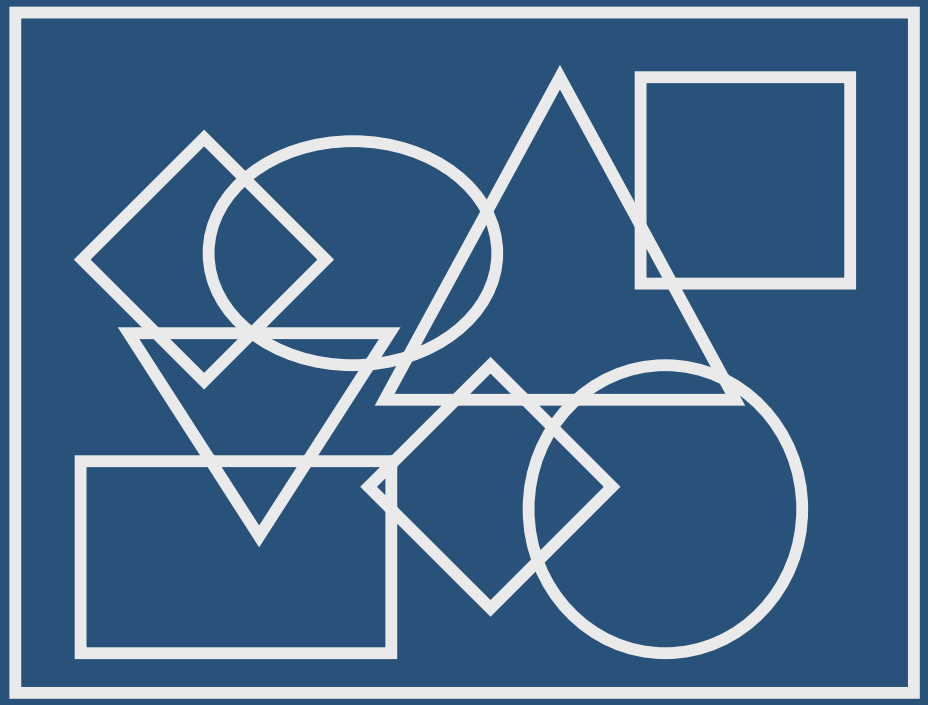
Architecture : Classification



[Molnar et al. `94]

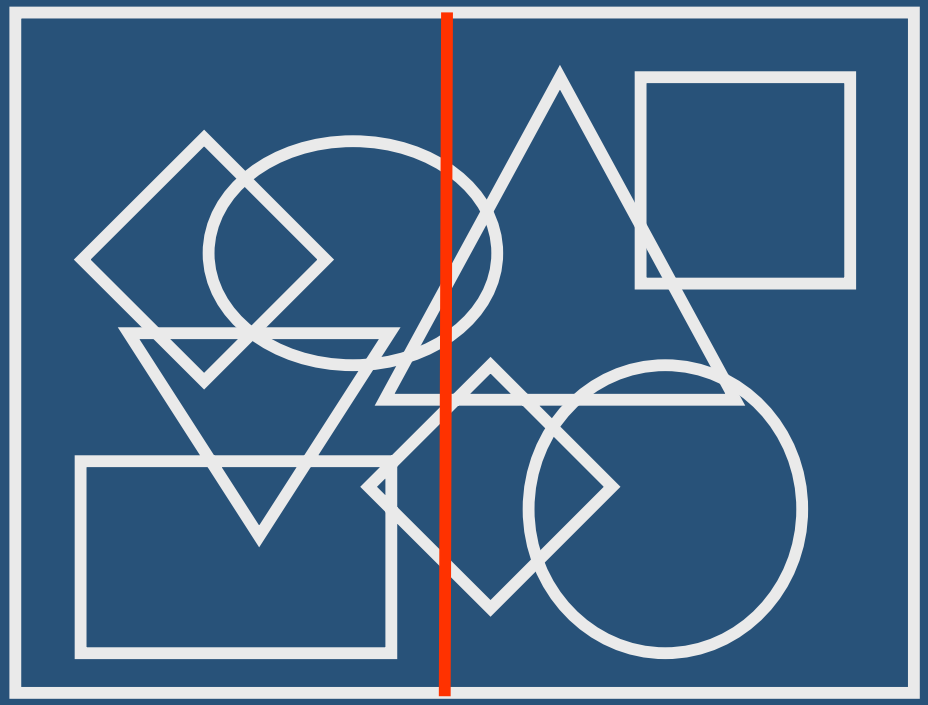
Architecture : Sort-First

- Partition pixels into non-overlapping tiles
- Render overlapping primitives redundantly



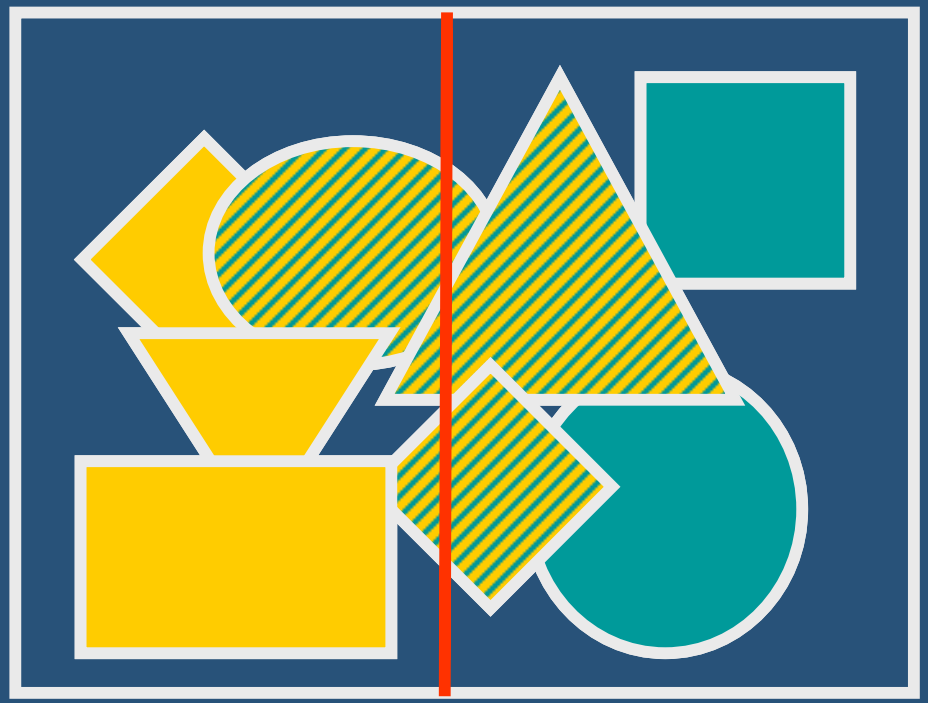
Architecture : Sort-First

- Partition pixels into non-overlapping tiles
- Render overlapping primitives redundantly



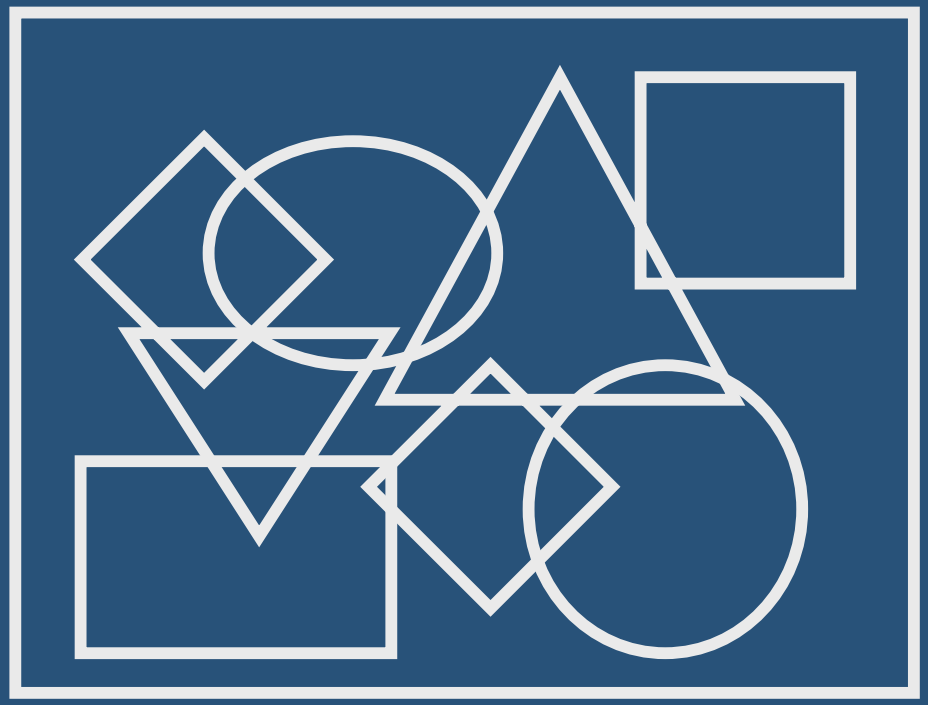
Architecture : Sort-First

- Partition pixels into non-overlapping tiles
- Render overlapping primitives redundantly



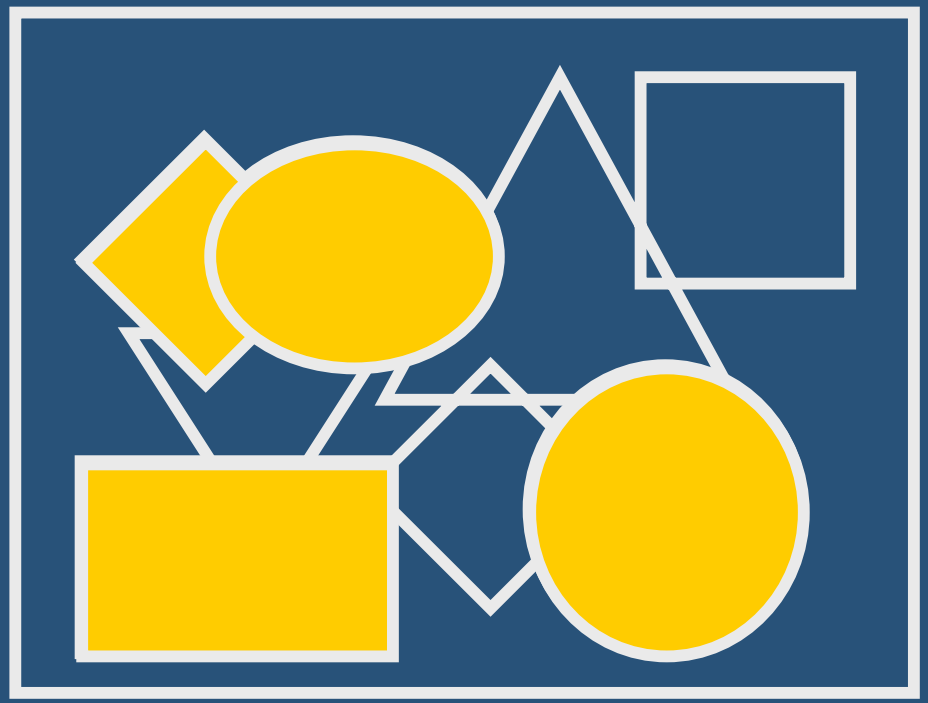
Architecture : Sort-Last

- Partition 3D primitives (e.g., round-robin)
- Z-composite overlapping screen areas



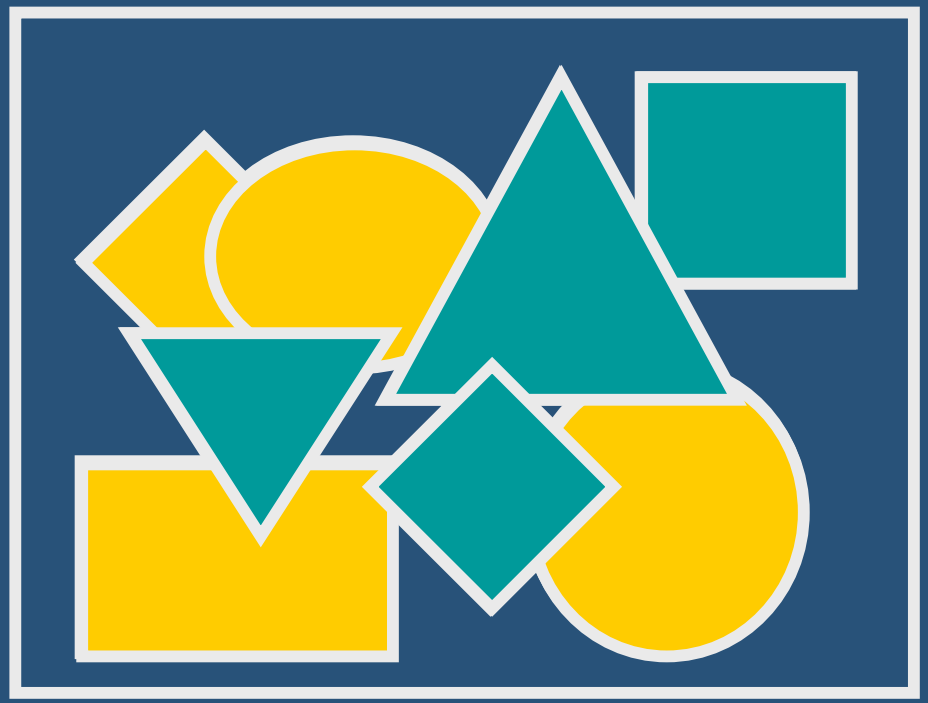
Architecture : Sort-Last

- Partition 3D primitives (e.g., round-robin)
- Z-composite overlapping screen areas



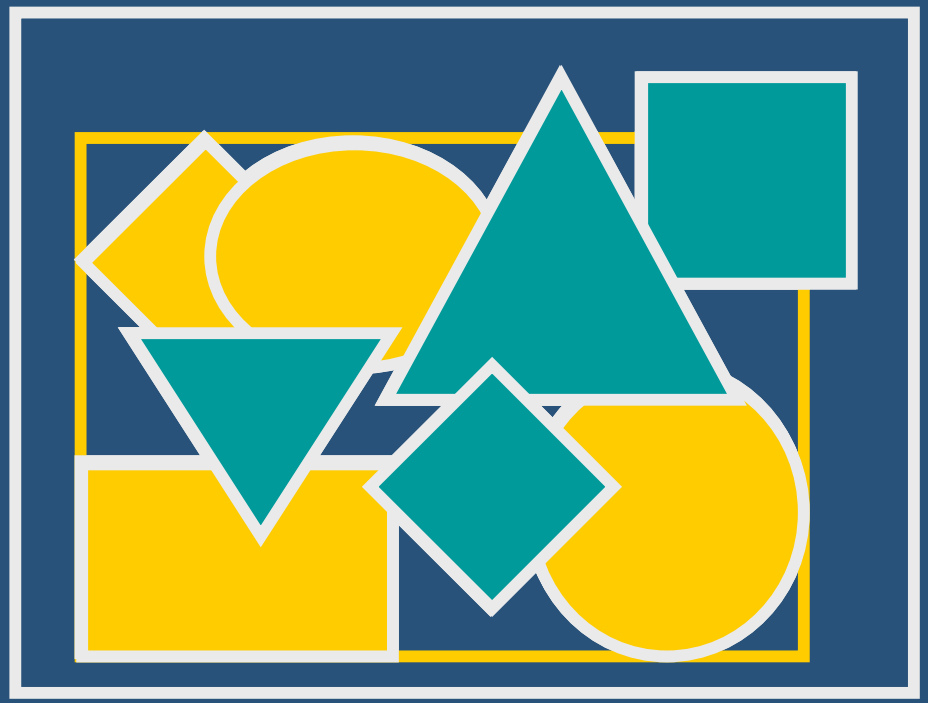
Architecture : Sort-Last

- Partition 3D primitives (e.g., round-robin)
- Z-composite overlapping screen areas



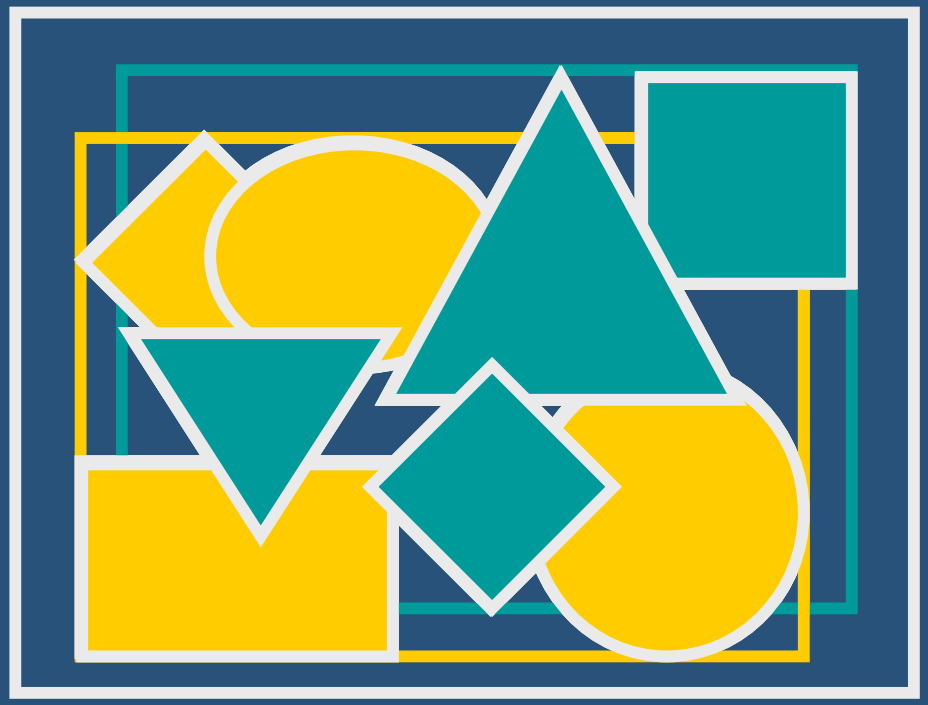
Architecture : Sort-Last

- Partition 3D primitives (e.g., round-robin)
- Z-composite overlapping screen areas



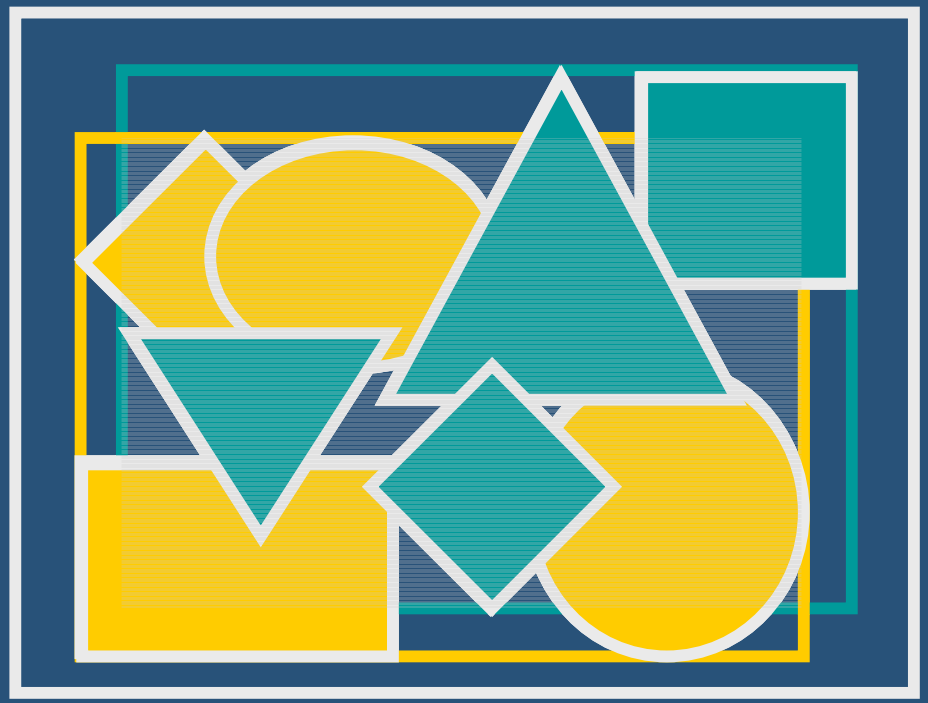
Architecture : Sort-Last

- Partition 3D primitives (e.g., round-robin)
- Z-composite overlapping screen areas



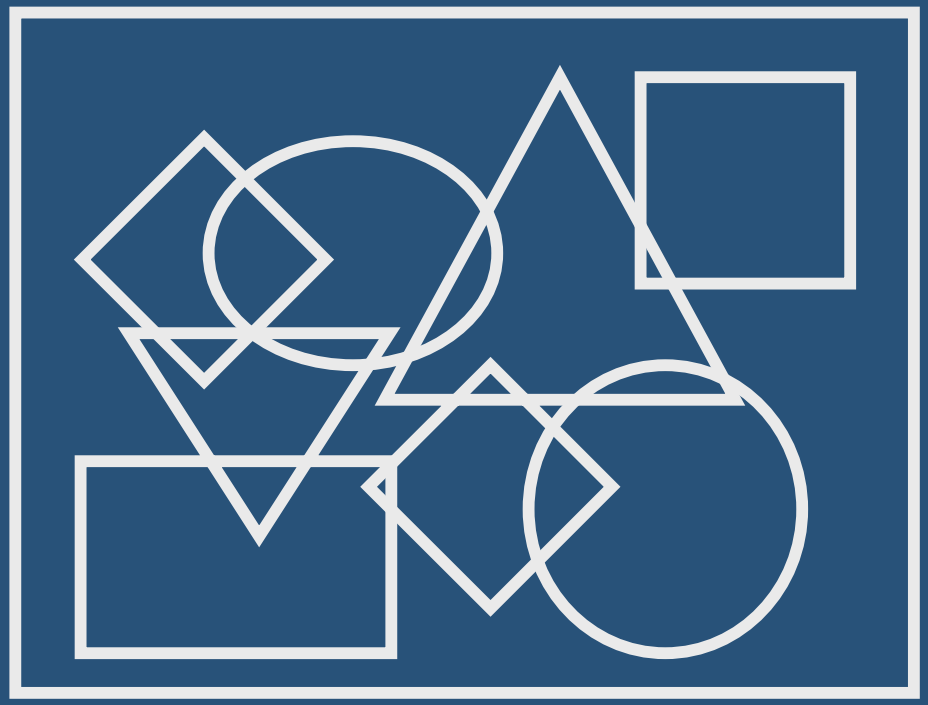
Architecture : Sort-Last

- Partition 3D primitives (e.g., round-robin)
- Z-composite overlapping screen areas



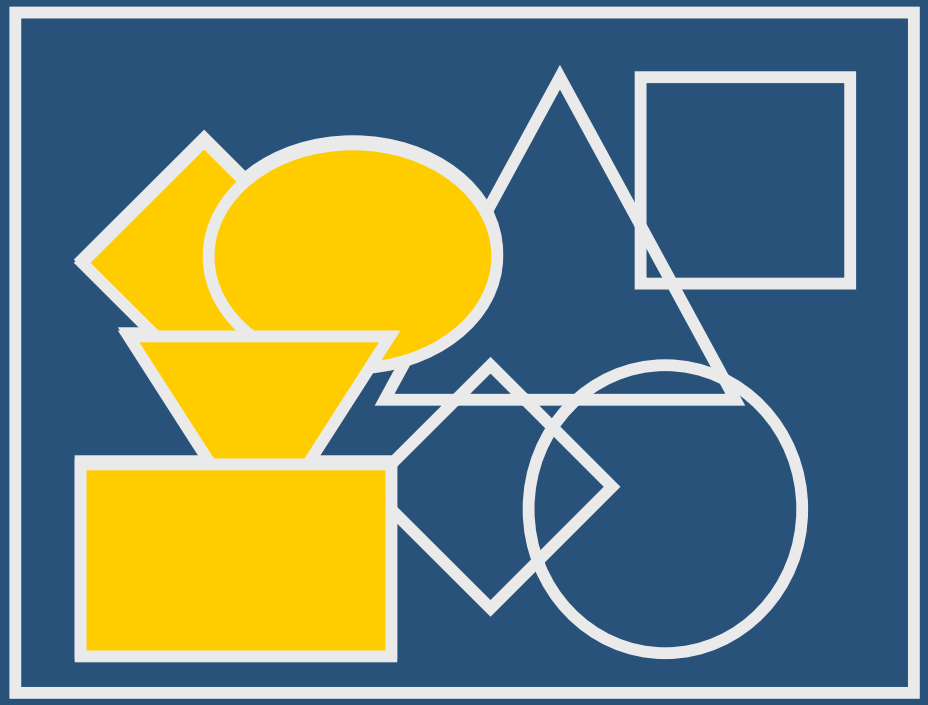
Our Approach : Hybrid

- Partition primitives and screen together
  - Dynamic, view-dependent partition
  - Cluster objects in screen space



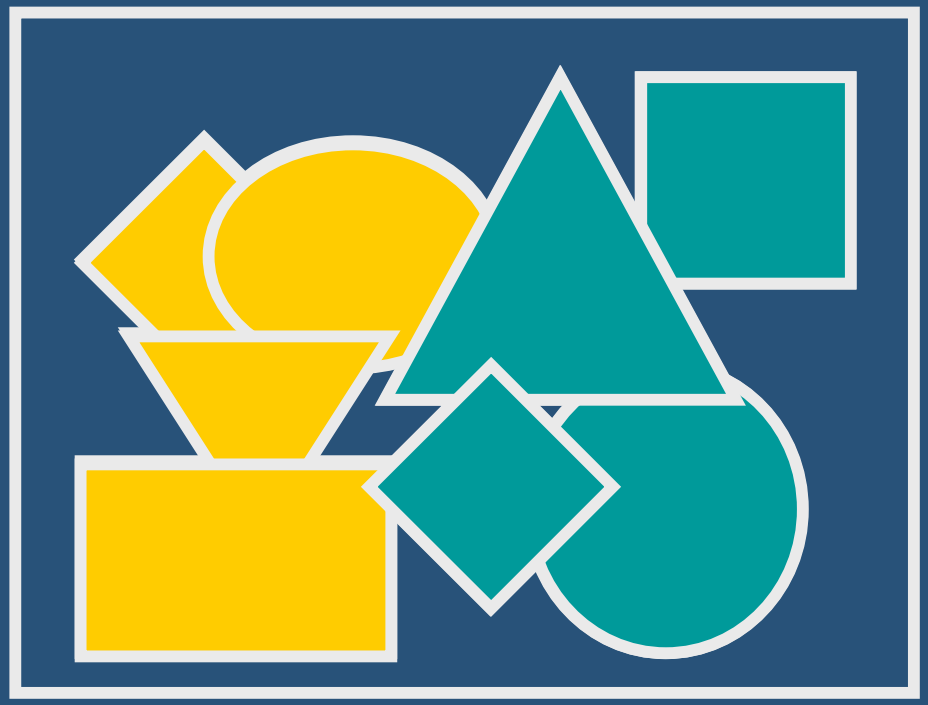
Our Approach : Hybrid

- Partition primitives and screen together
  - Dynamic, view-dependent partition
  - Cluster objects in screen space



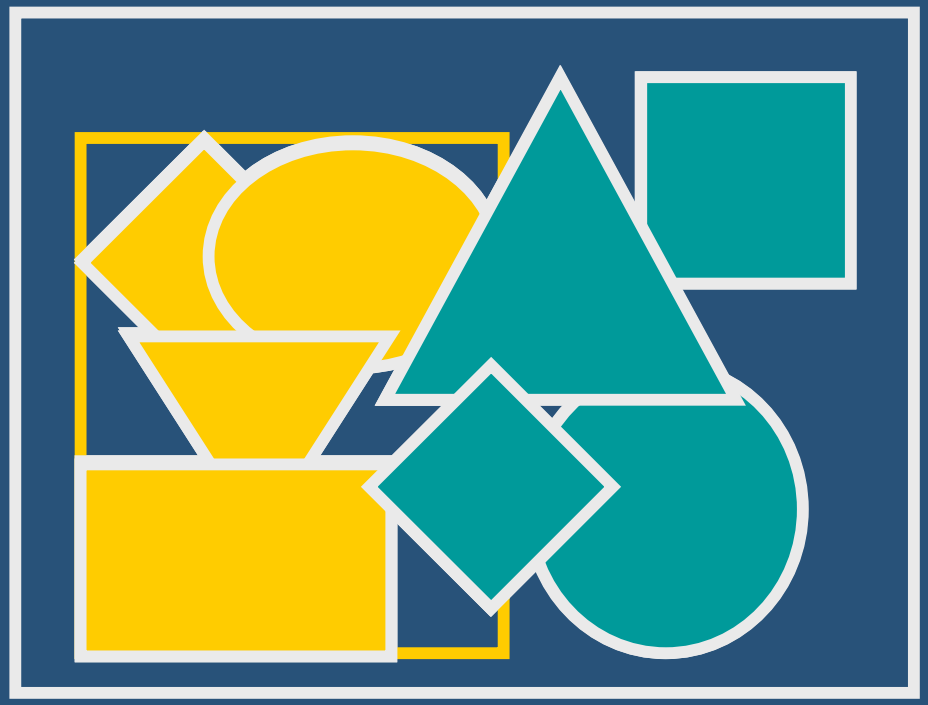
Our Approach : Hybrid

- Partition primitives and screen together
  - Dynamic, view-dependent partition
  - Cluster objects in screen space



Our Approach : Hybrid

- Partition primitives and screen together
  - Dynamic, view-dependent partition
  - Cluster objects in screen space



Our Approach : Hybrid

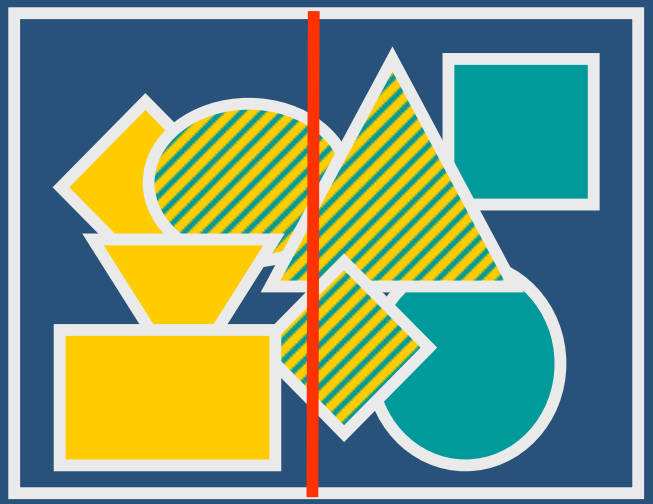
- Partition primitives and screen together
  - Dynamic, view-dependent partition
  - Cluster objects in screen space

Our Approach : Hybrid

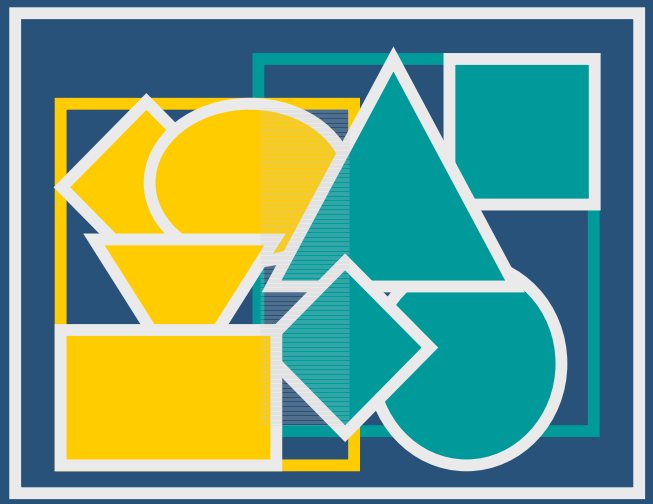
- Partition primitives and screen together
  - Dynamic, view-dependent partition
  - Cluster objects in screen space

Comparison to Sort-First

- Avoid redundant rendering
  - By depth sorting pixels in overlapping regions



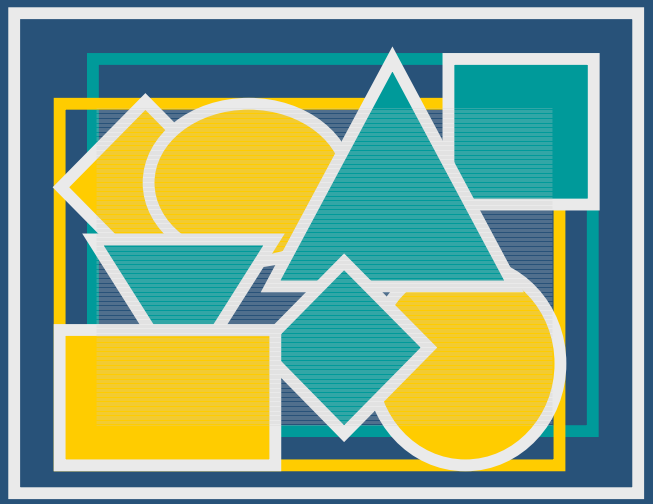
Sort-First



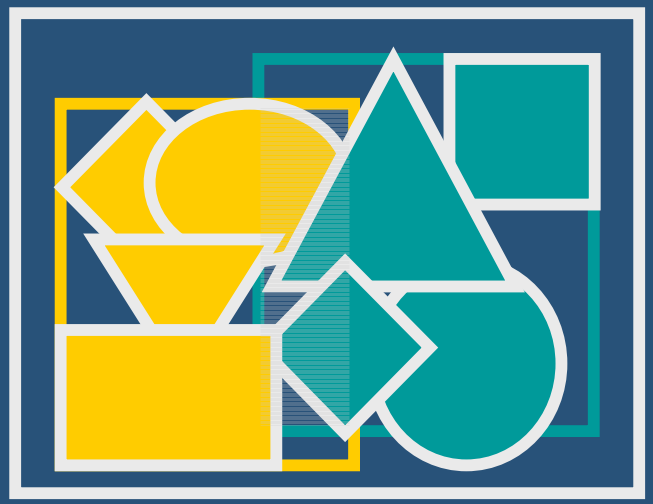
Hybrid

Comparison to Sort-Last

- Composite fewer pixels
  - By sorting objects based on screen projections



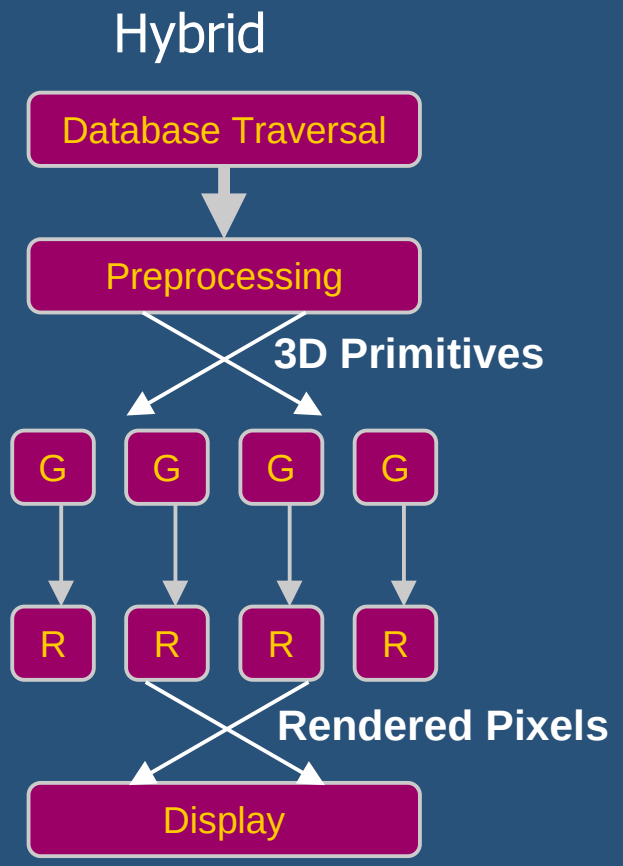
Sort-Last



Hybrid

Key Idea : Sort-Twice

- First sort by client
  - Sorts objects
- Second sort by servers
  - Depth sorts pixels

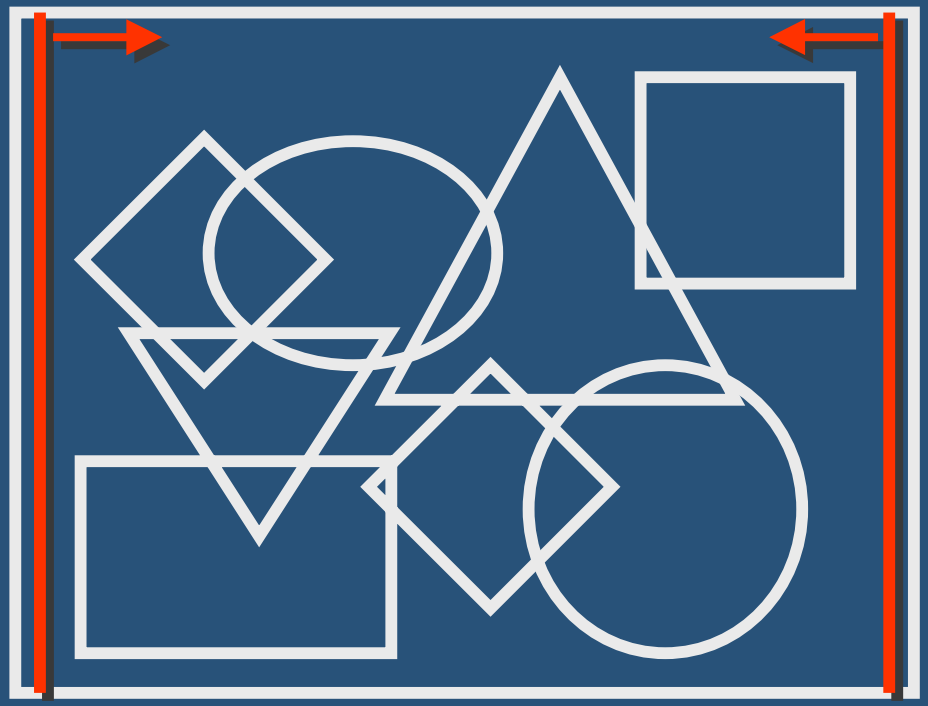


Outline

- Motivation
- System Architecture
- Algorithms
- Simulation Results
- Conclusion and Future Work

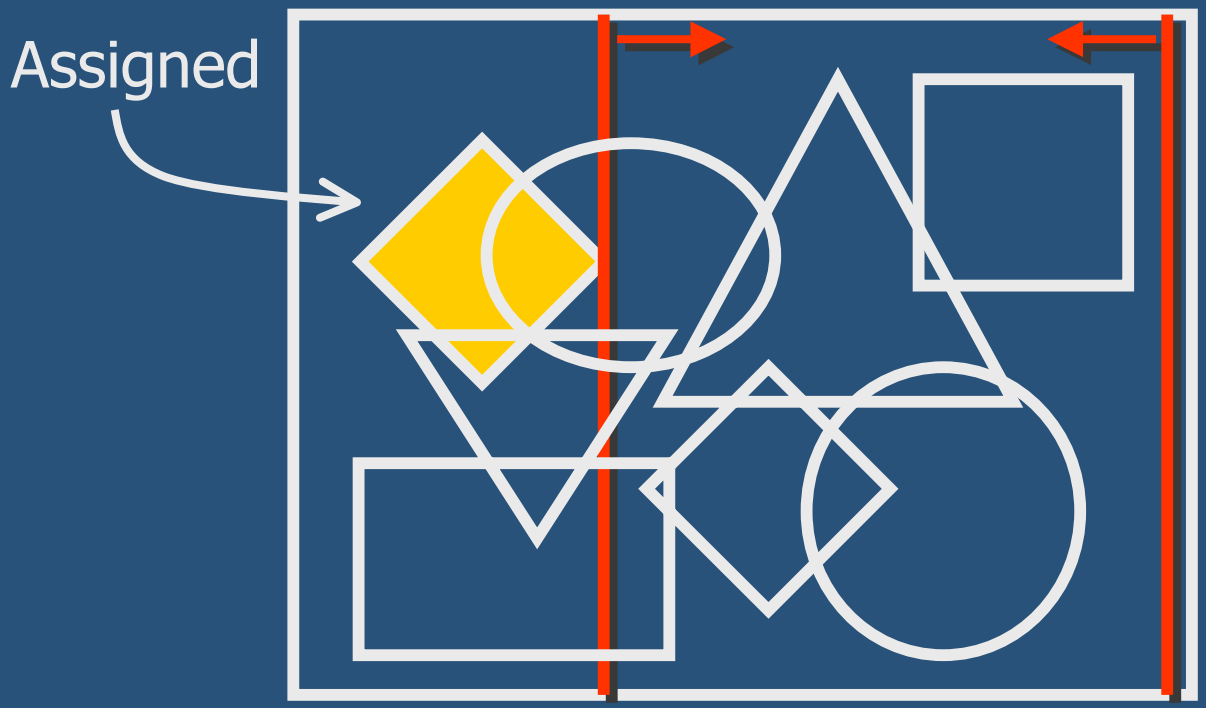
Algorithm : Execution

- Sweep-line algorithm
- Move line for group with least work



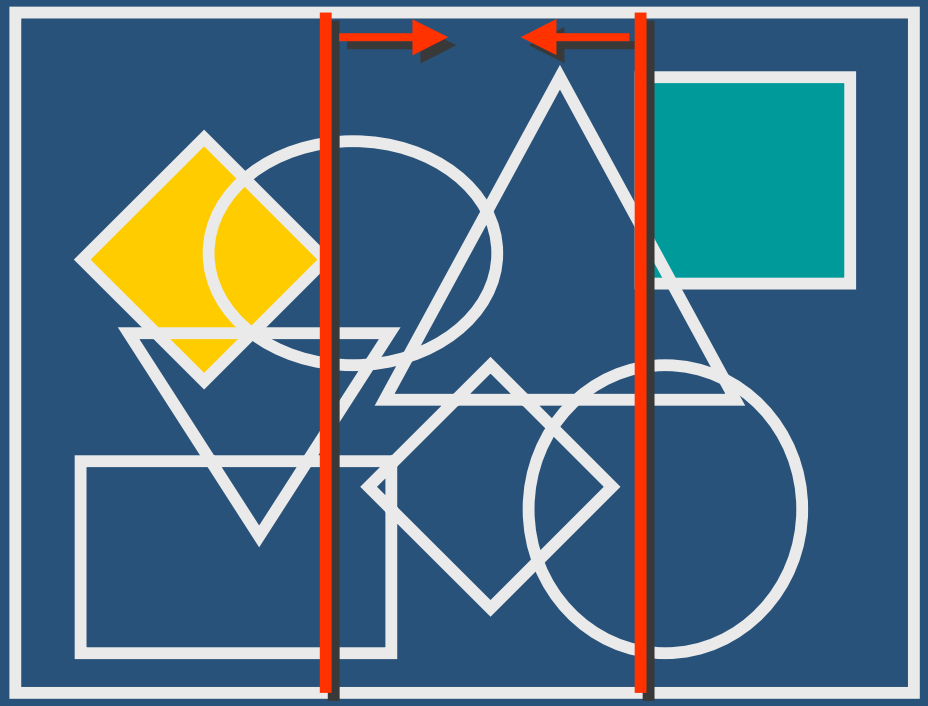
Algorithm : Execution

- Sweep-line algorithm
- Move line for group with least work



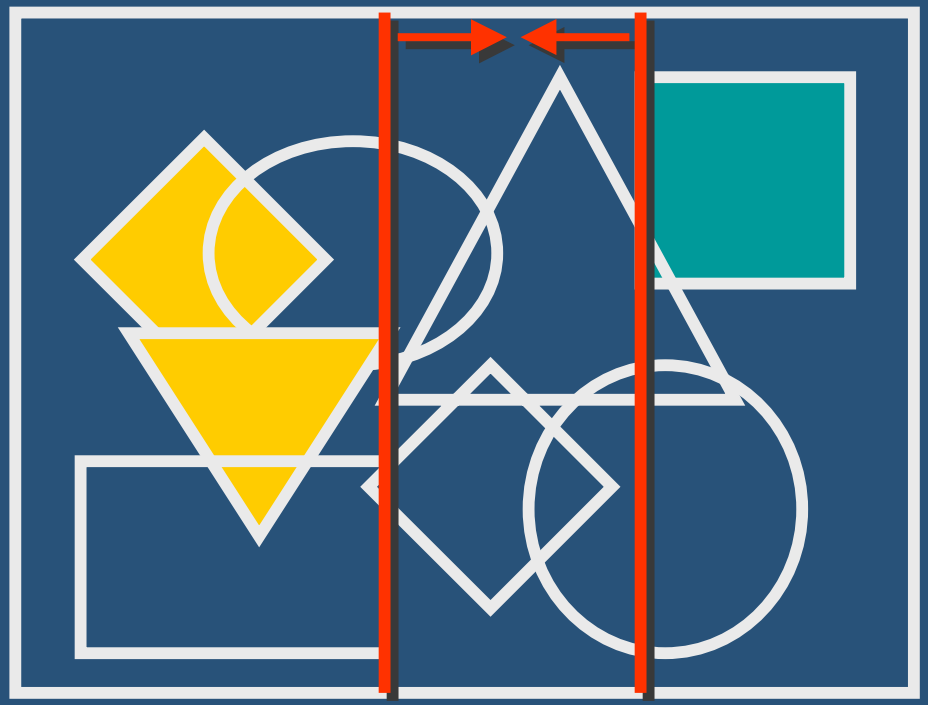
Algorithm : Execution

- Sweep-line algorithm
- Move line for group with least work



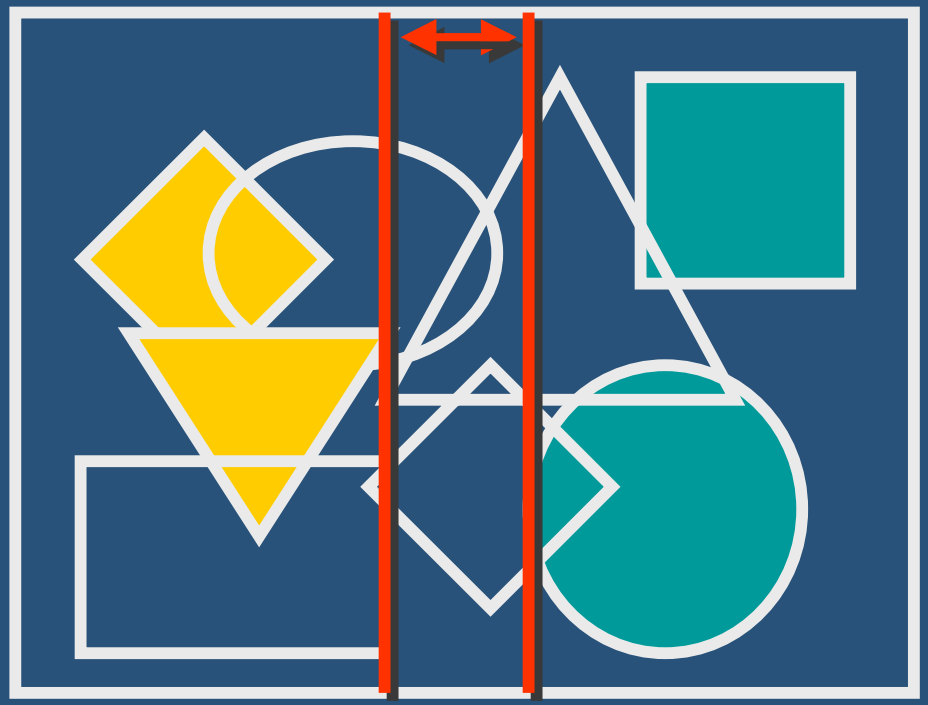
Algorithm : Execution

- Sweep-line algorithm
- Move line for group with least work



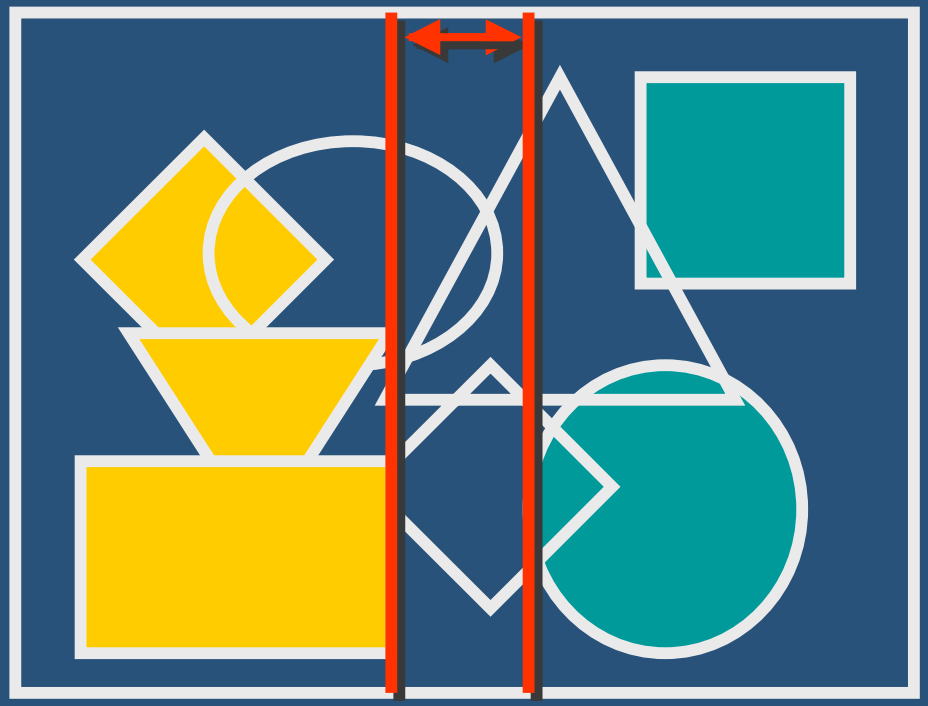
Algorithm : Execution

- Sweep-line algorithm
- Move line for group with least work



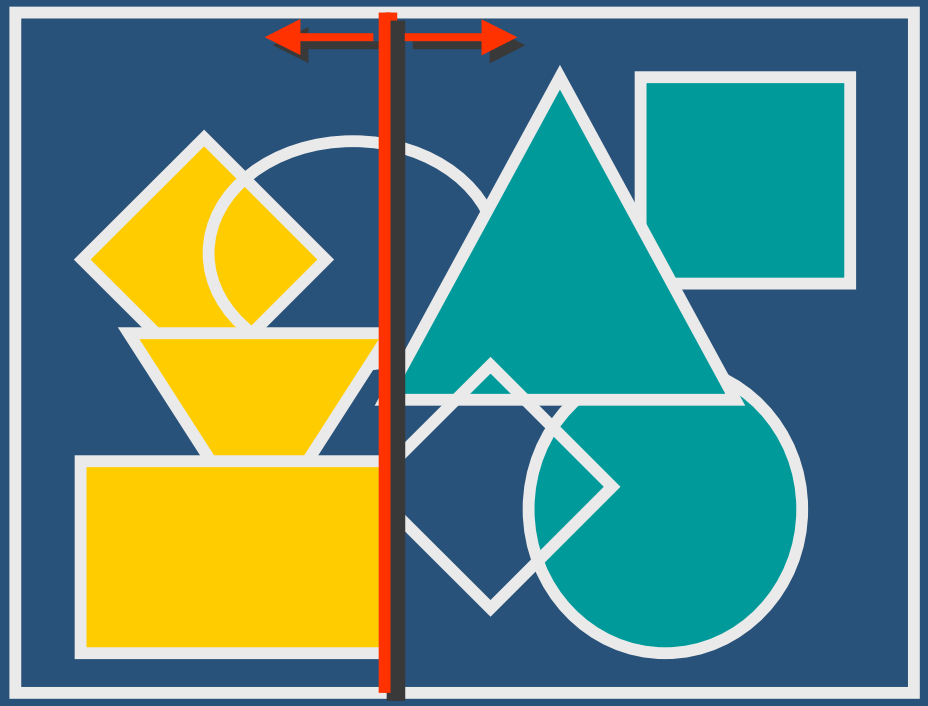
Algorithm : Execution

- Sweep-line algorithm
- Move line for group with least work



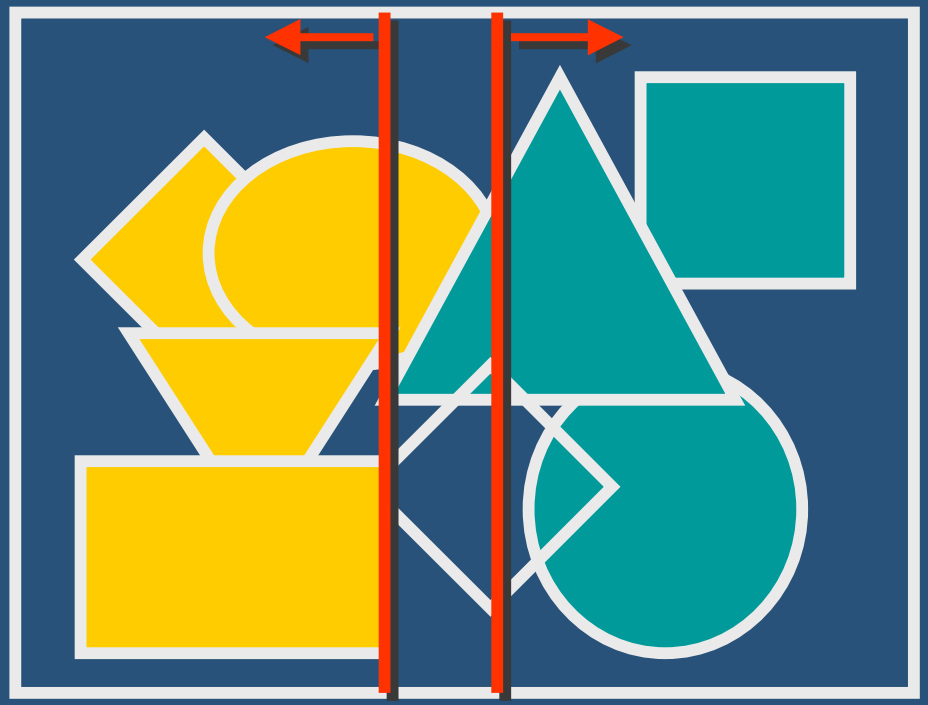
Algorithm : Execution

- Sweep-line algorithm
- Move line for group with least work



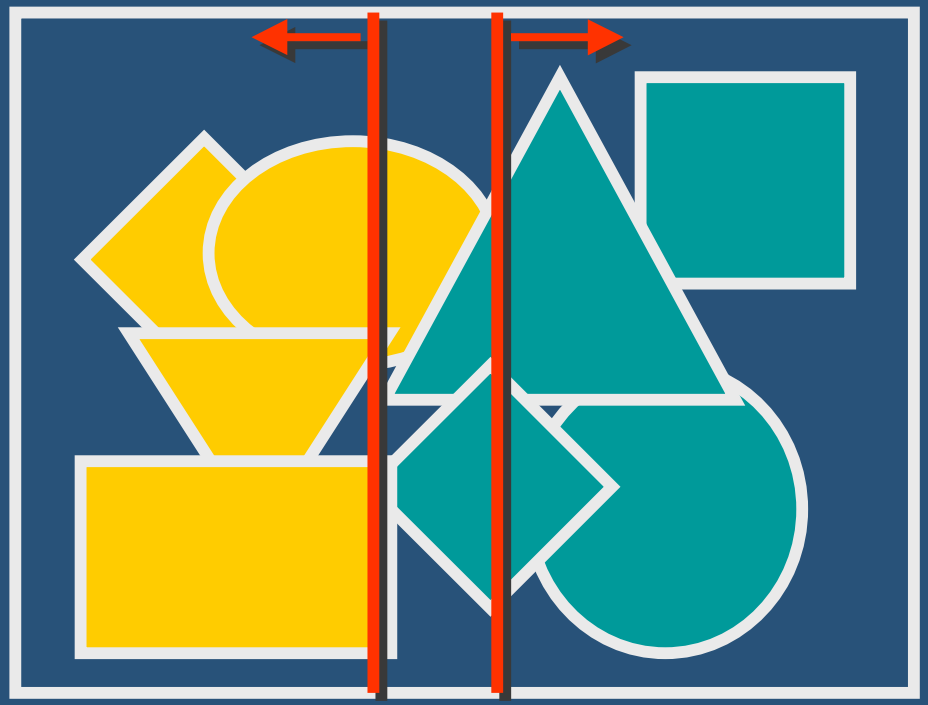
Algorithm : Execution

- Sweep-line algorithm
- Move line for group with least work



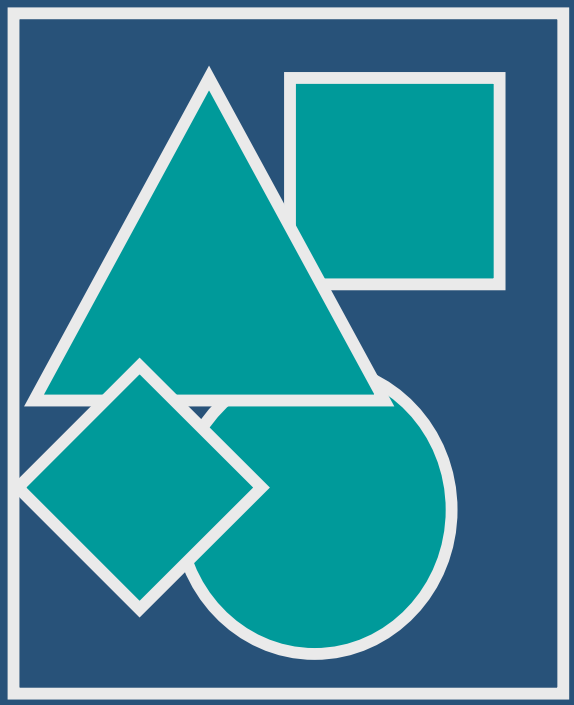
Algorithm : Execution

- Sweep-line algorithm
- Move line for group with least work



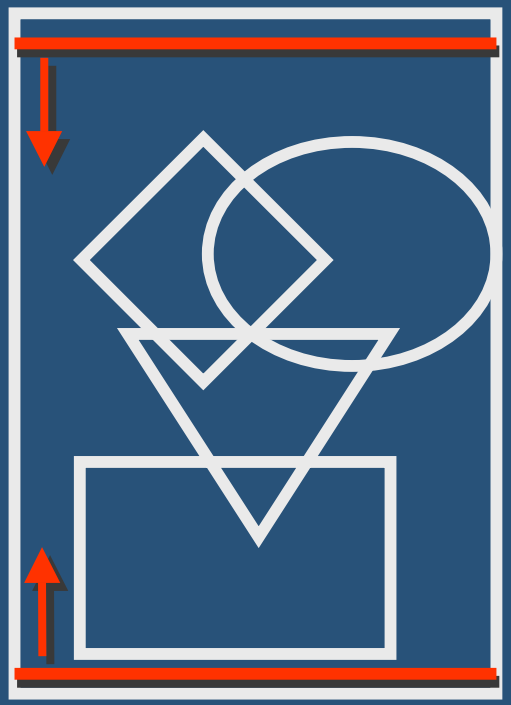
Algorithm : Execution

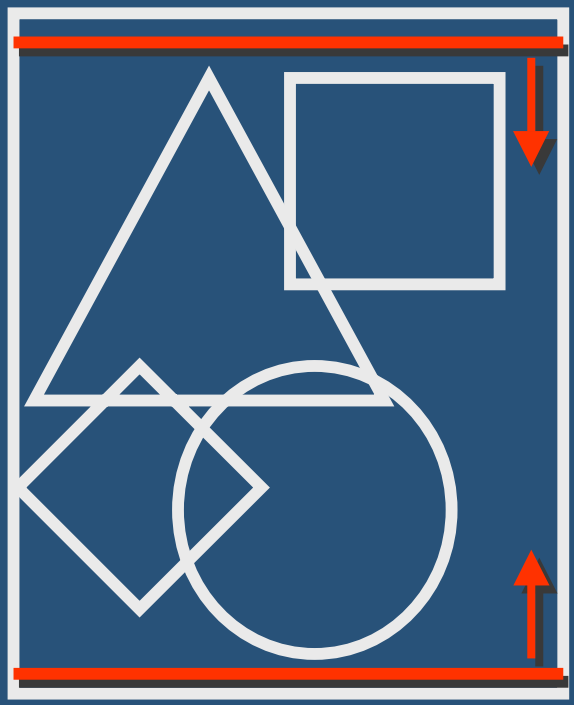
- Perform algorithm recursively on each tile



Algorithm : Execution

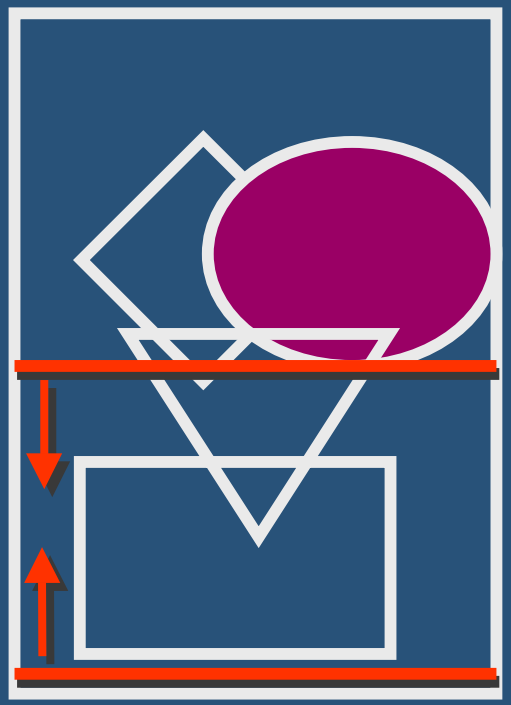
- Partition works in orthogonal direction

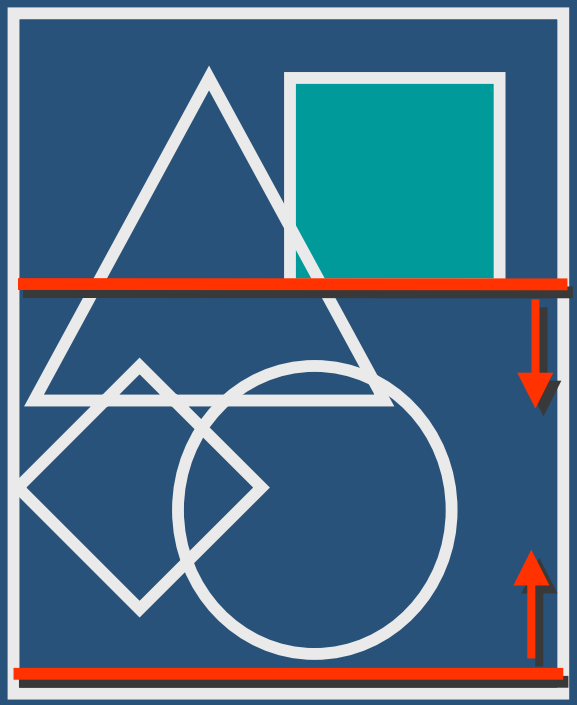




Algorithm : Execution

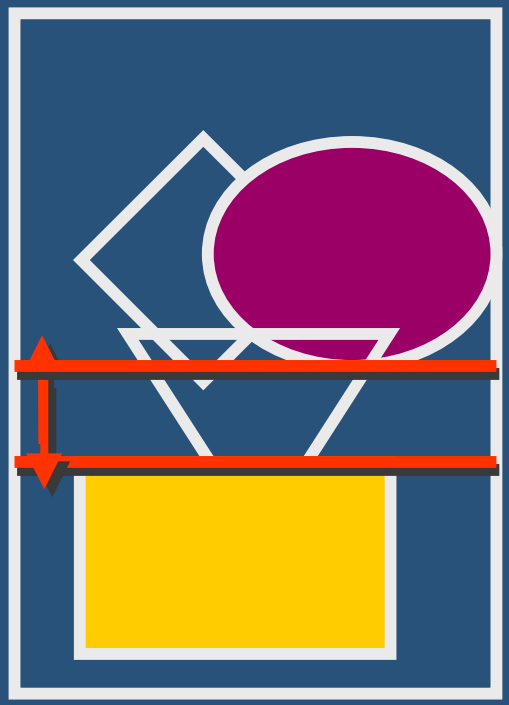
- Partition works in orthogonal direction

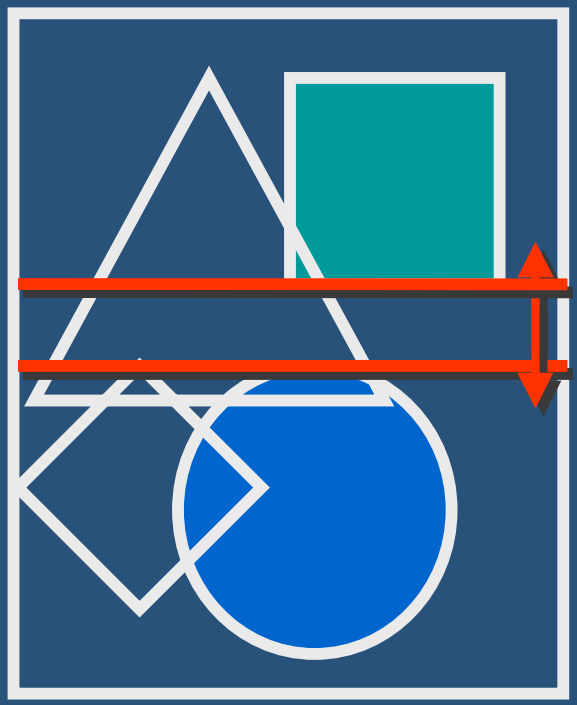




Algorithm : Execution

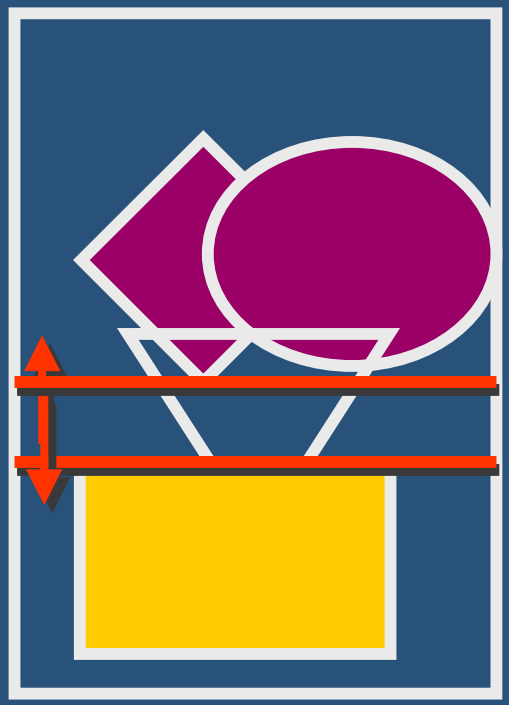
- Partition works in orthogonal direction

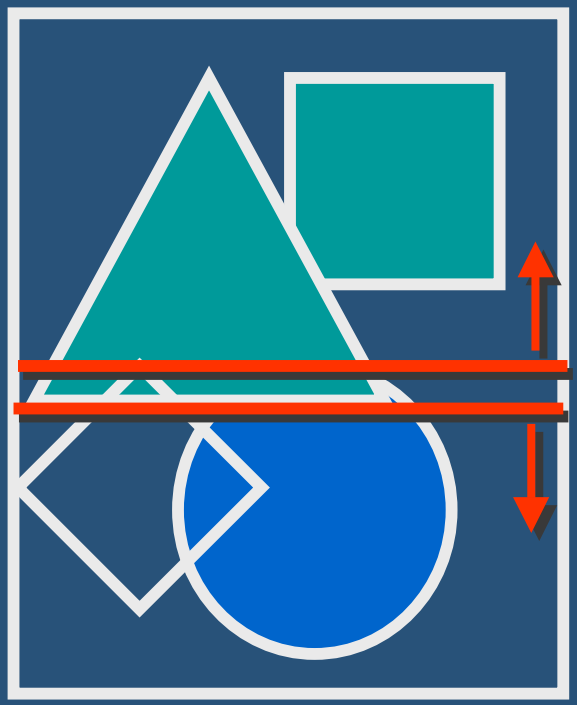




Algorithm : Execution

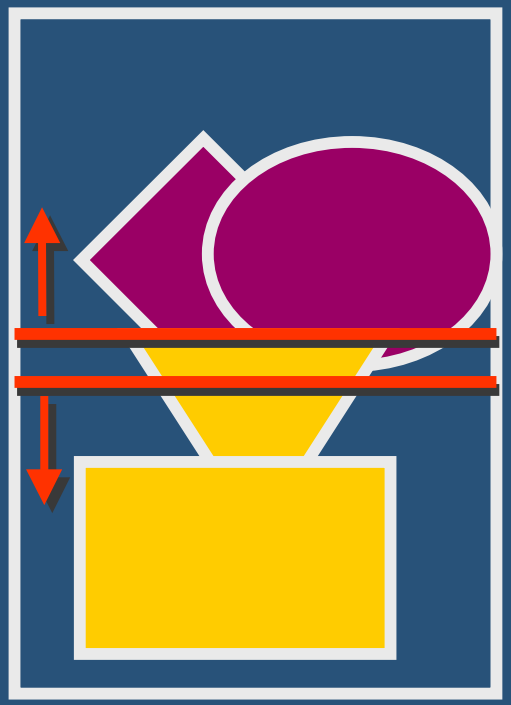
- Partition works in orthogonal direction

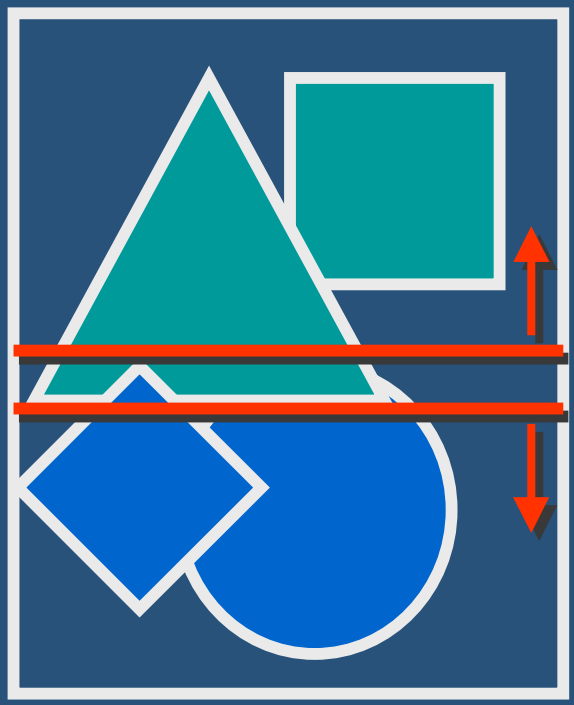




Algorithm : Execution

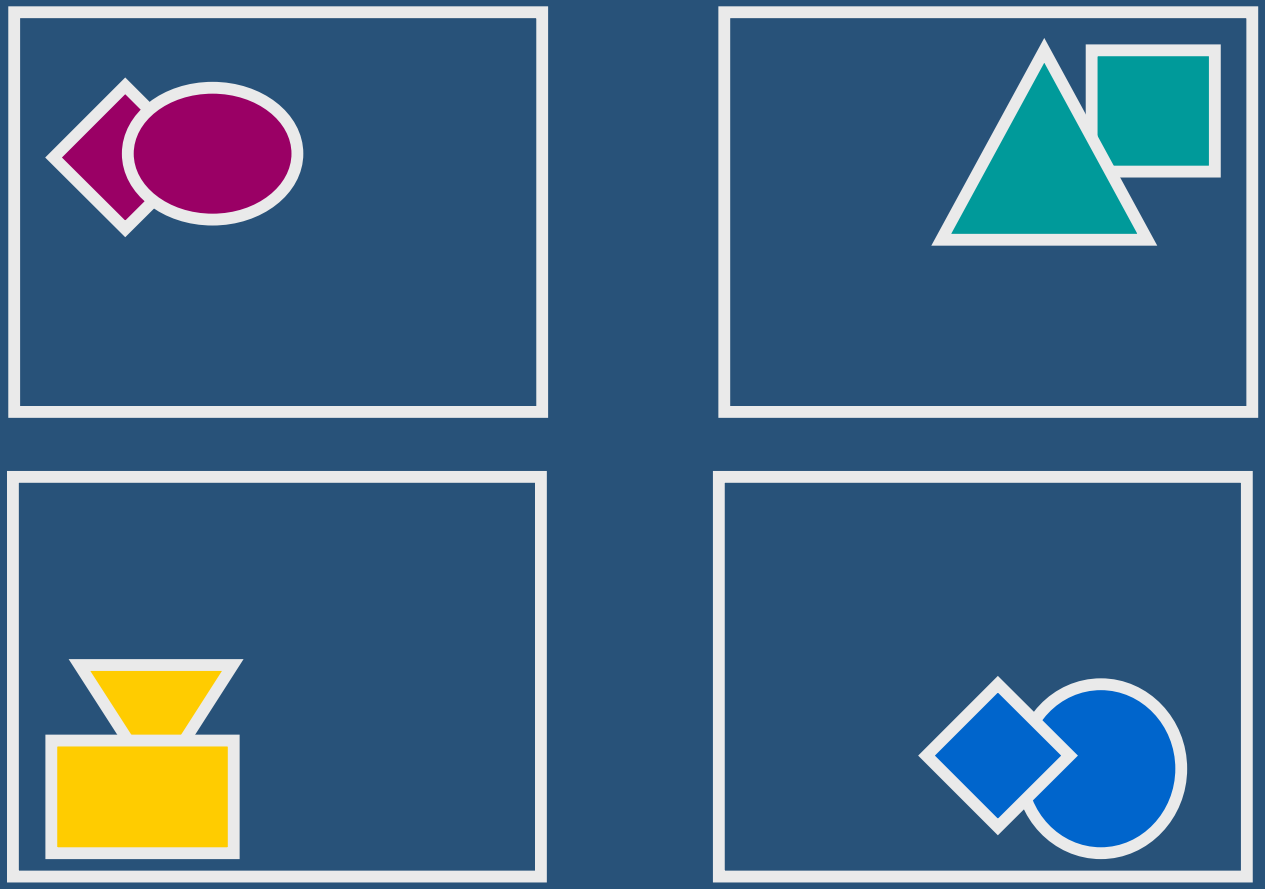
- Partition works in orthogonal direction





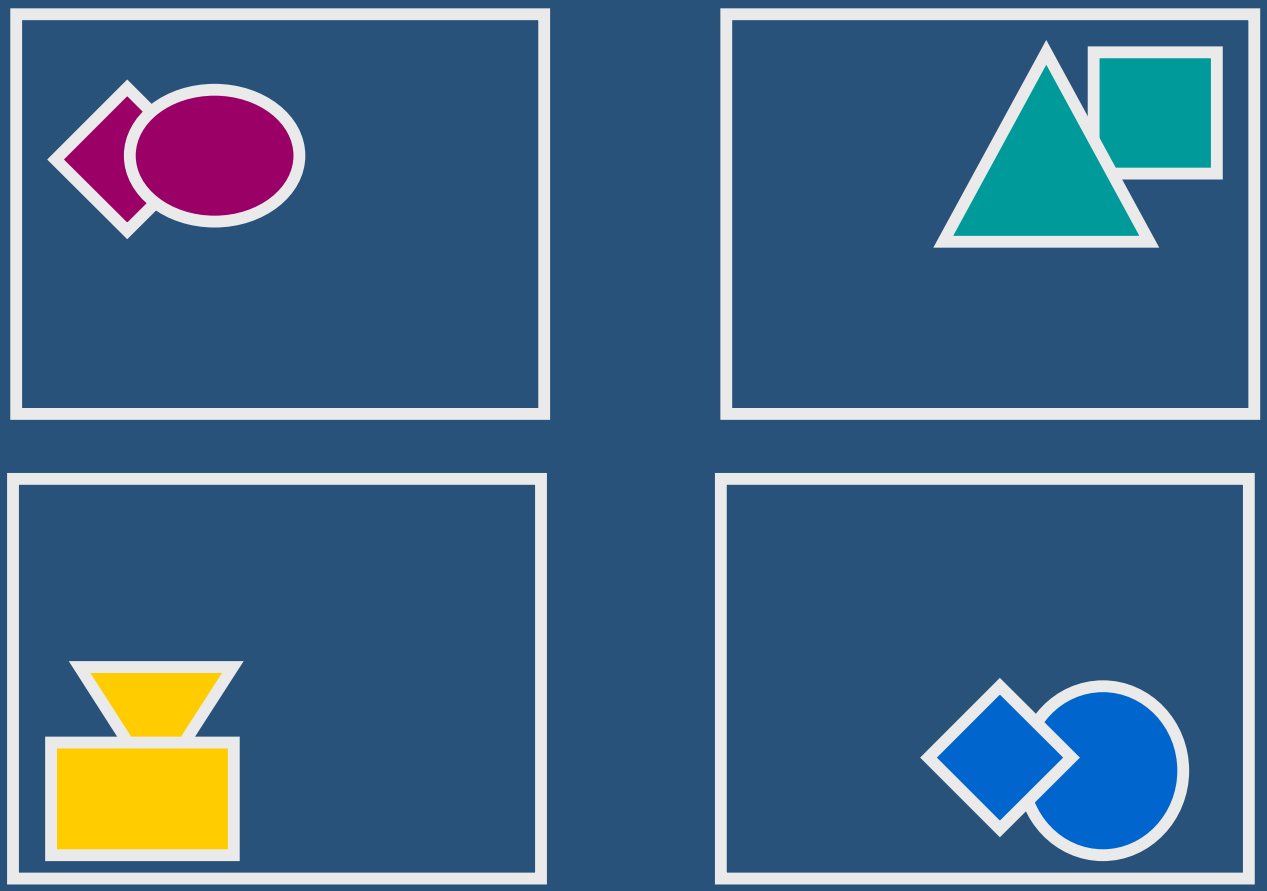
Algorithm : Execution

- Now servers render their respective tiles



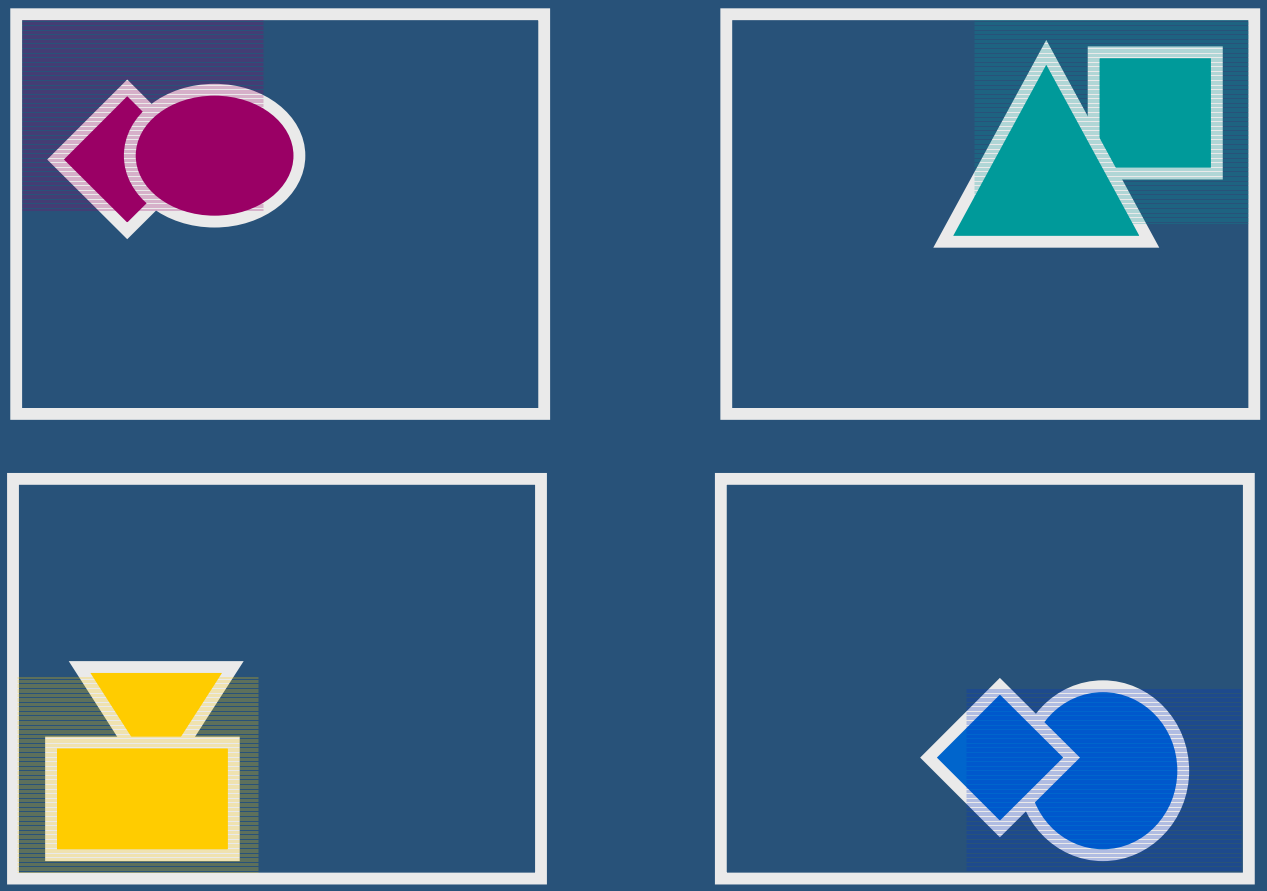
Algorithm : Execution

- Servers were assigned a composite region



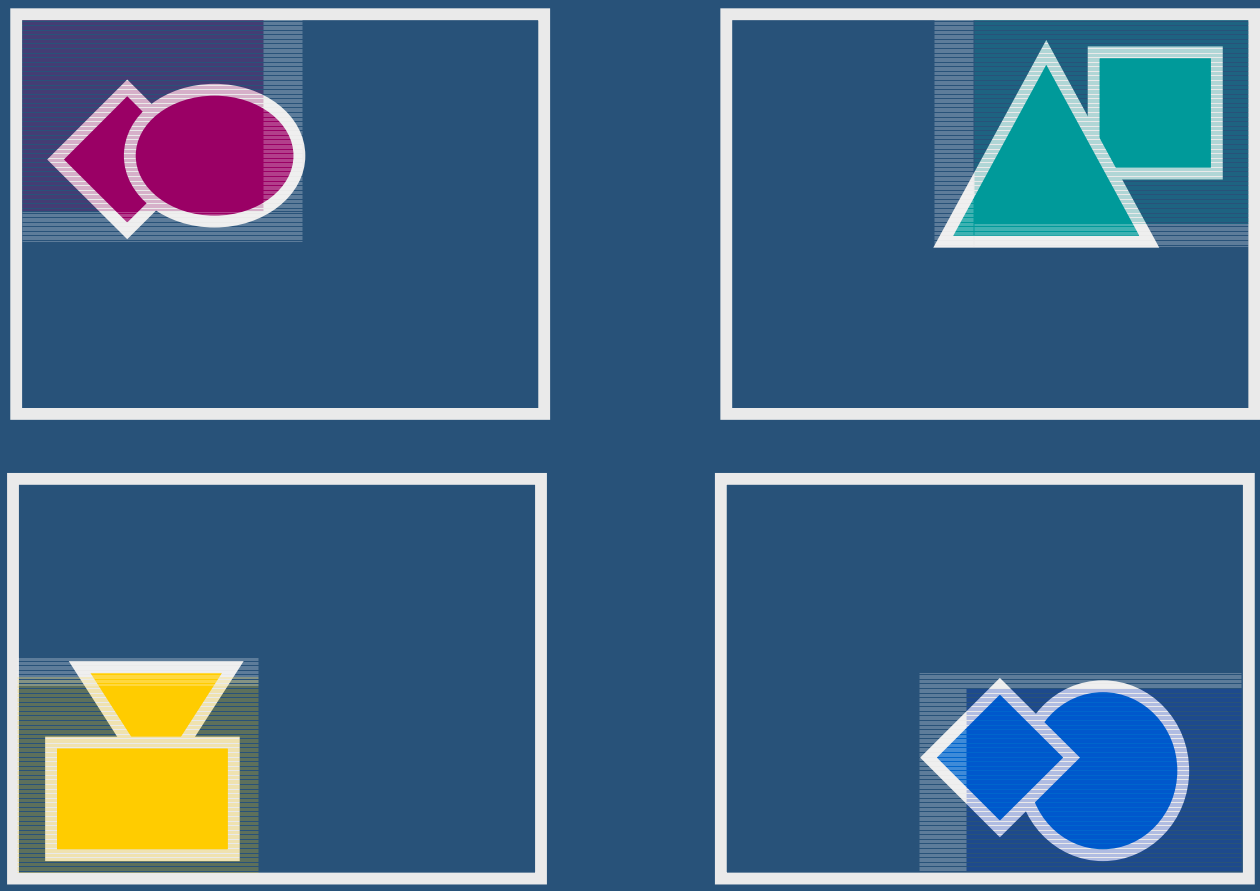
Algorithm : Execution

- Servers were assigned a composite region



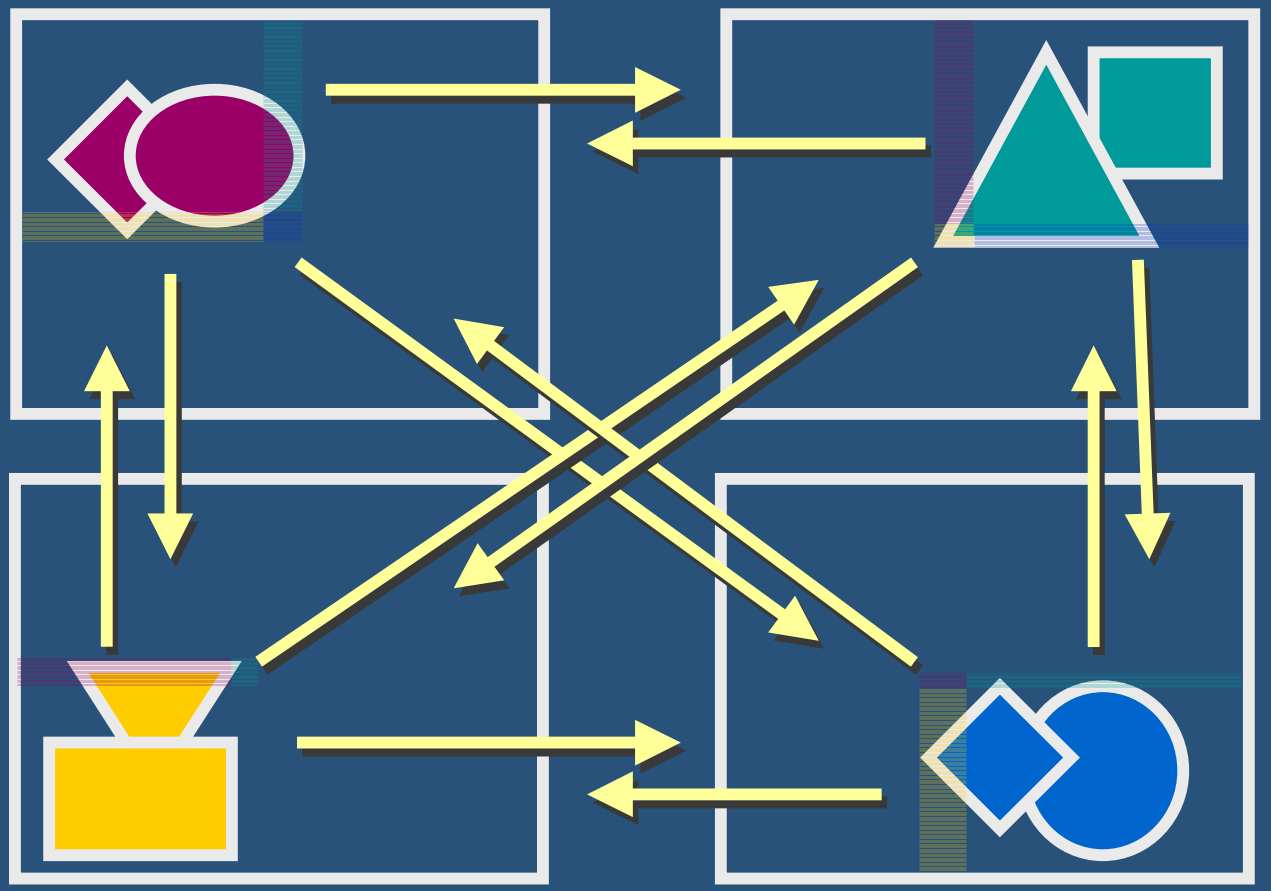
Algorithm : Execution

- Servers were assigned a composite region



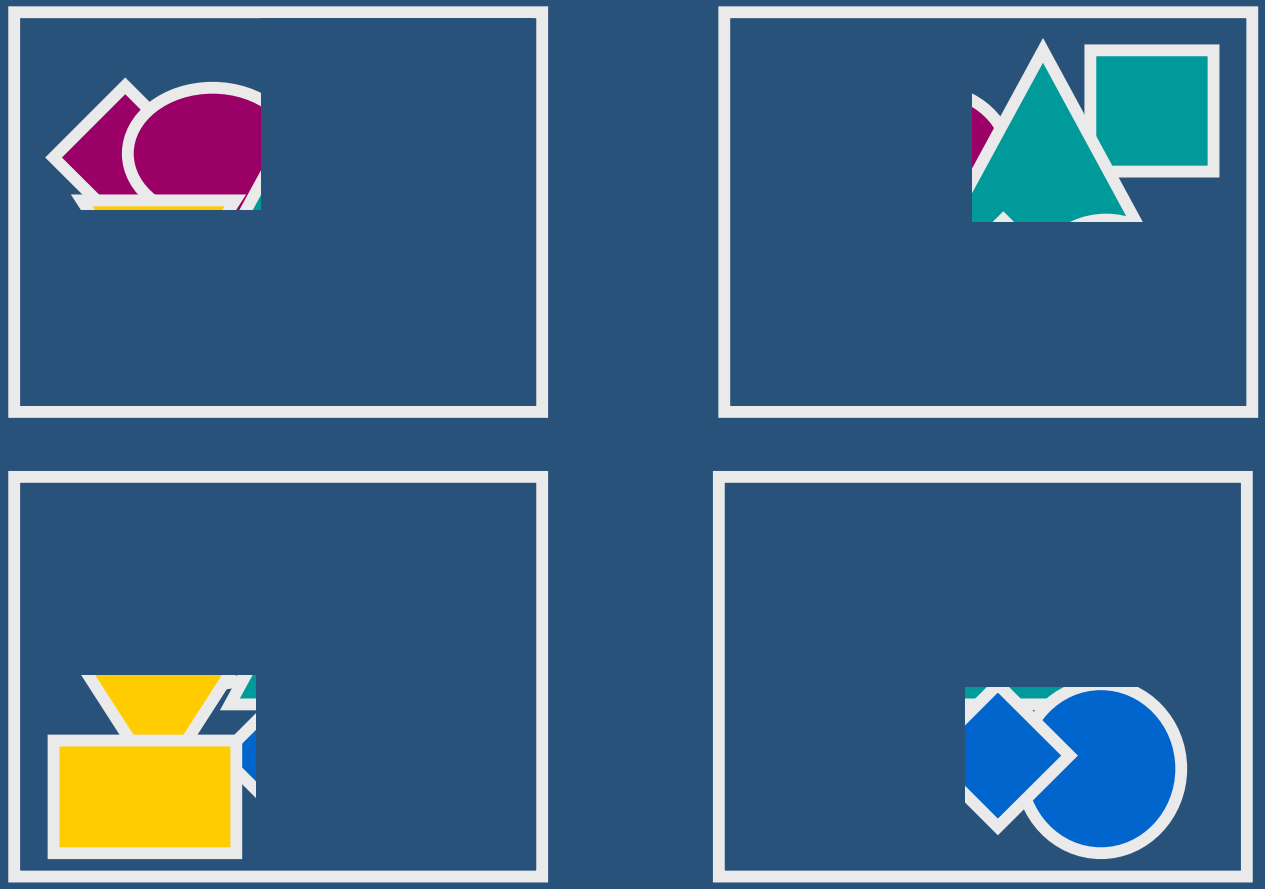
Algorithm : Execution

- Pixels sent to the assigned nodes



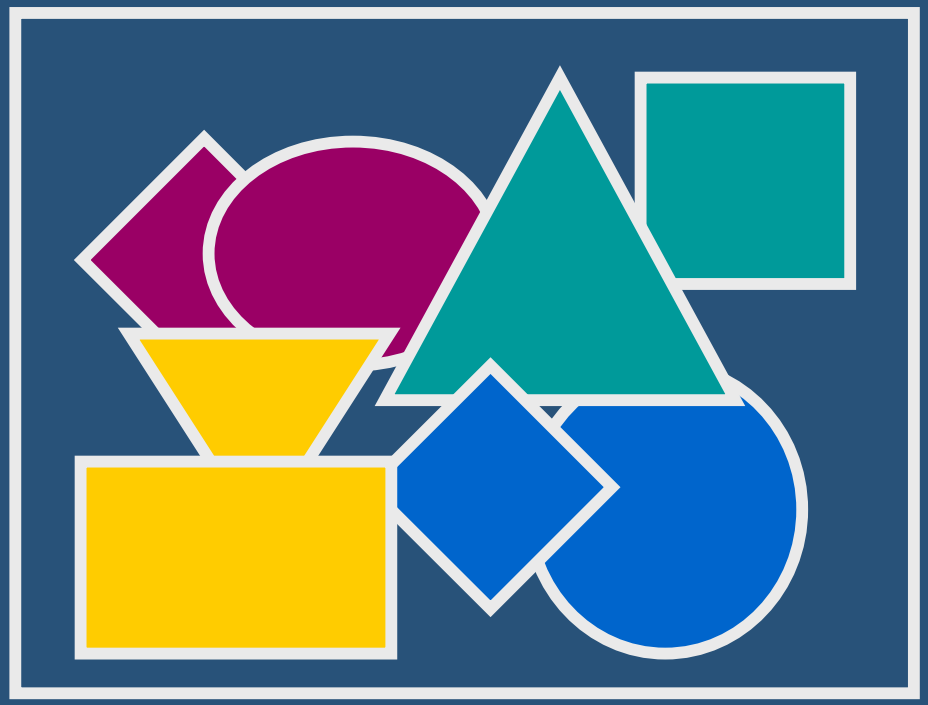
Algorithm : Execution

- The nodes composite the pixels received



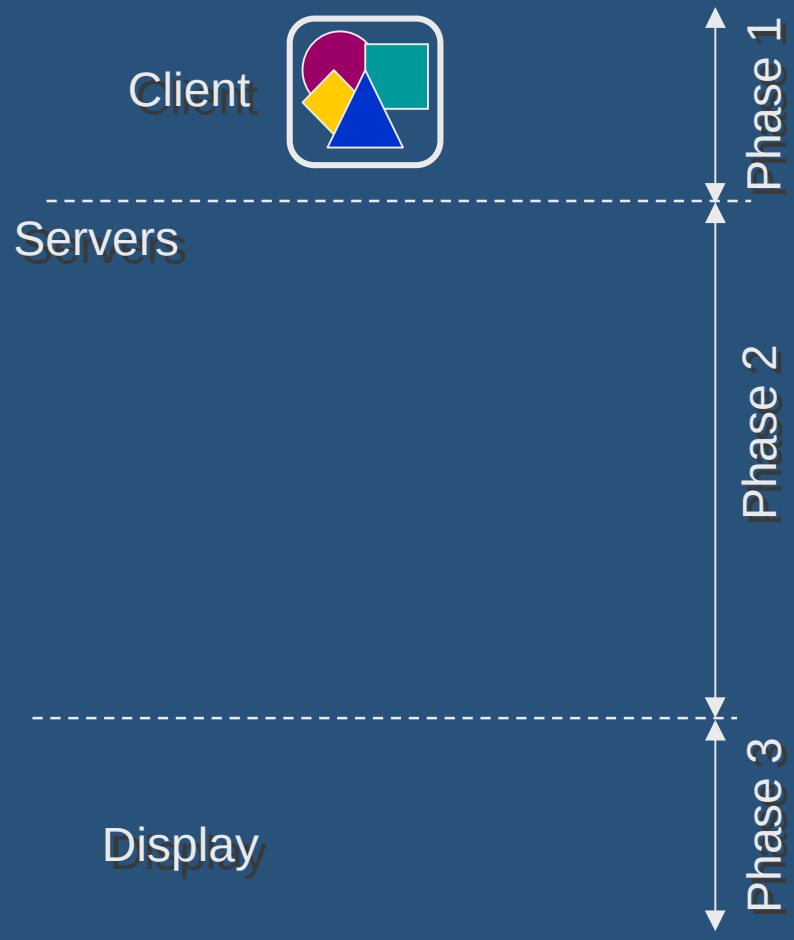
Algorithm : Execution

- Server nodes send tiles to the display



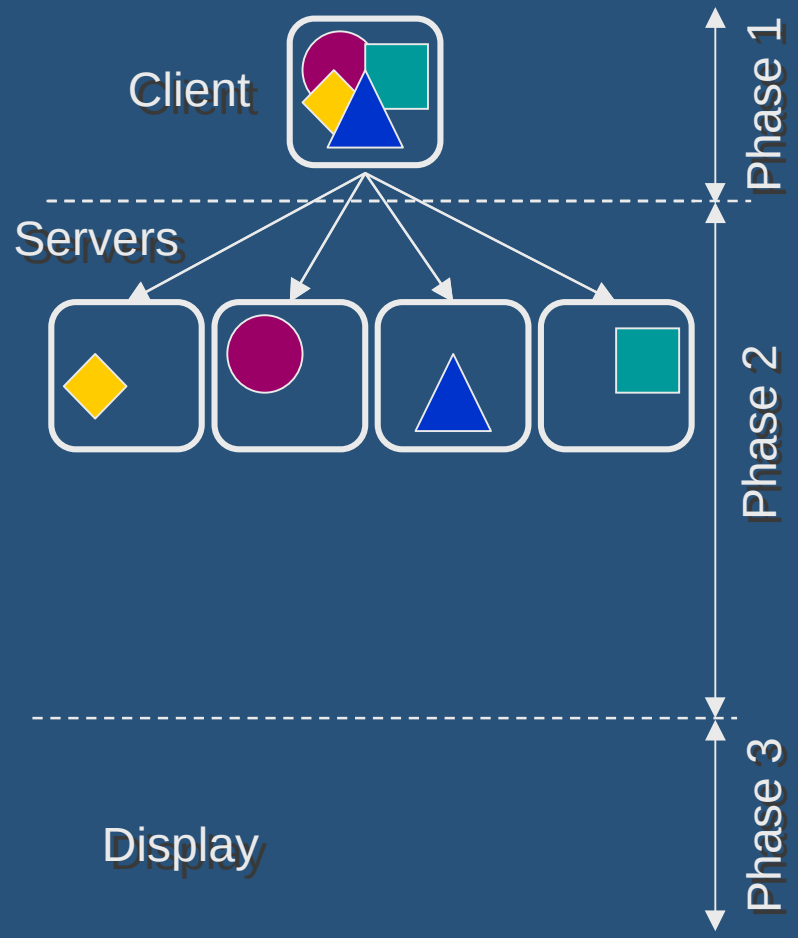
Algorithm : System Stages

- 3 Pipelined stages



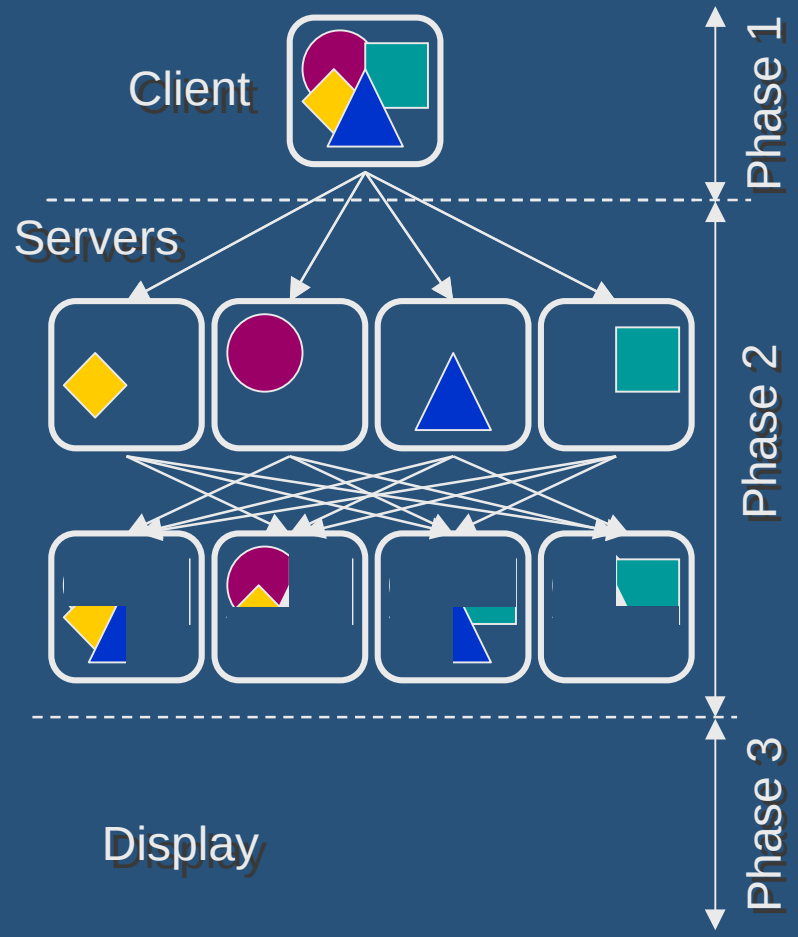
Algorithm : System Stages

- 3 Pipelined stages



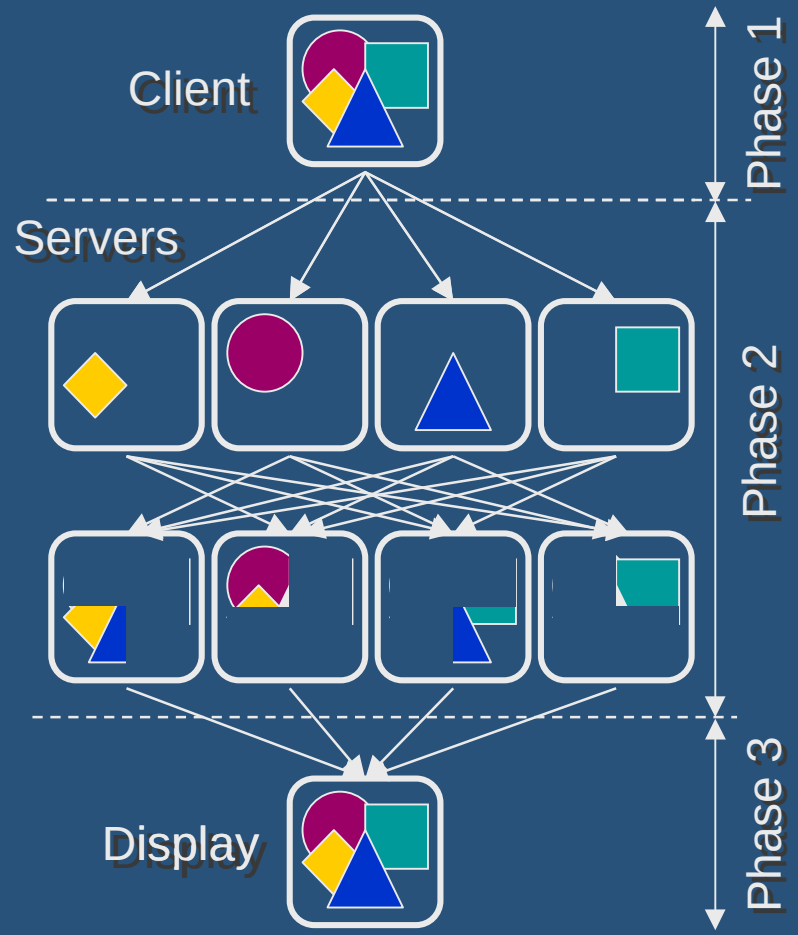
Algorithm : System Stages

- 3 Pipelined stages



Algorithm : System Stages

- 3 Pipelined stages

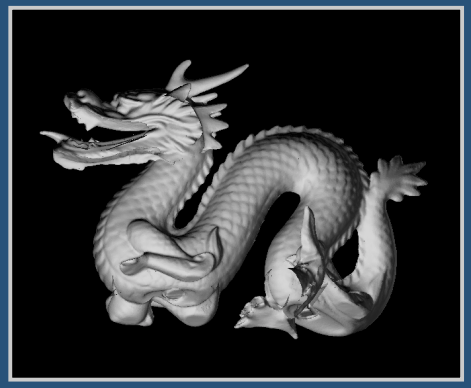


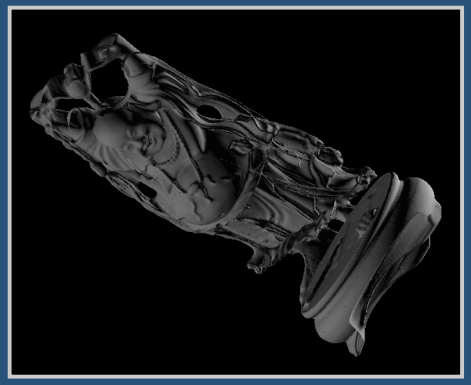
Outline

- Motivation
- System Architecture
- Algorithms
- Simulation Results
- Conclusion and Future Work

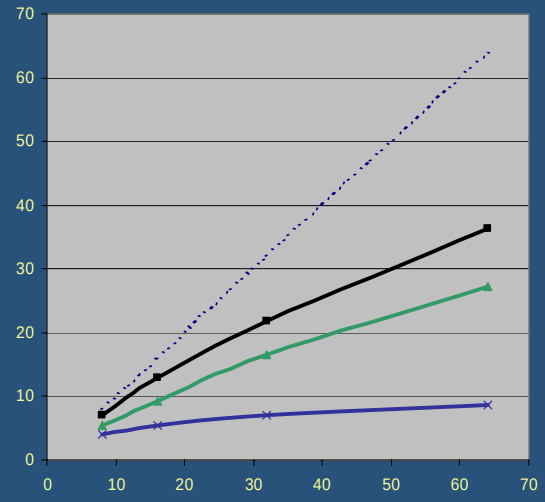
Results : Setup

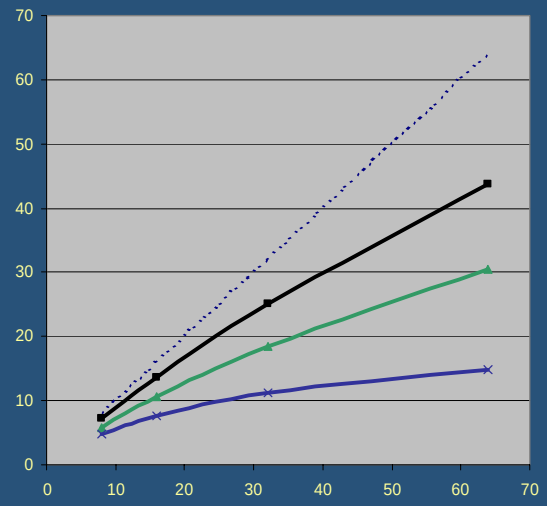
- Simulated hardware
  - Pentium III 500Mhz
  - GeForce accelerators
  - Myrinet network
- Screen resolution
  - 1280 x 960
- Algorithms
  - Sort-First
  - Hybrid
  - Sort-Last

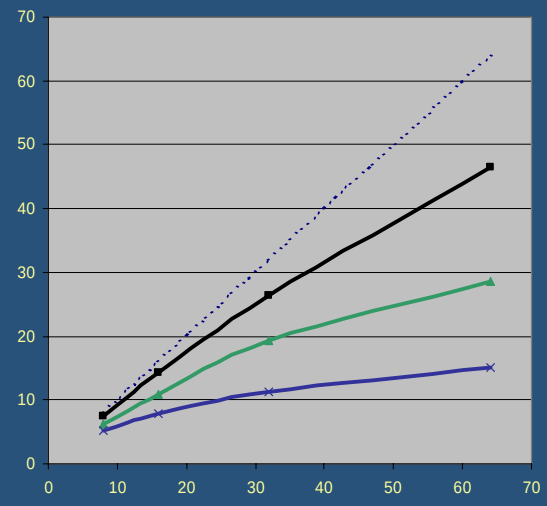


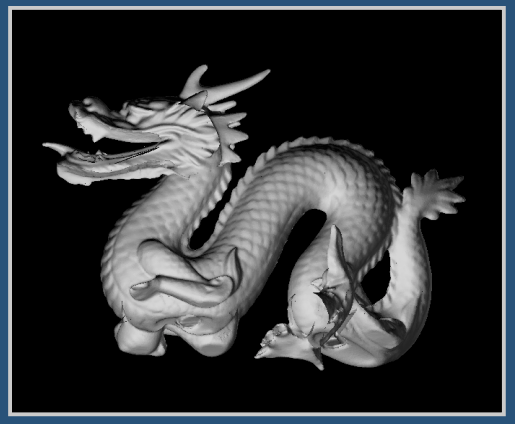


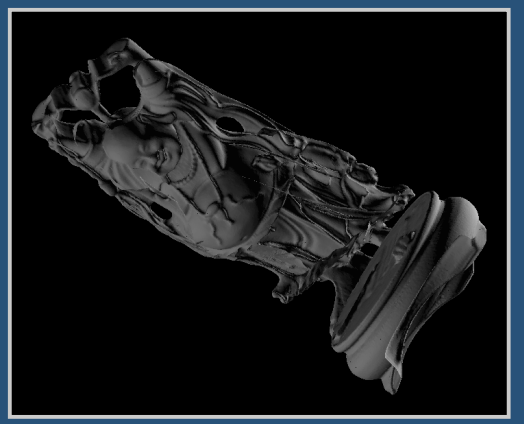
Results : Speedups









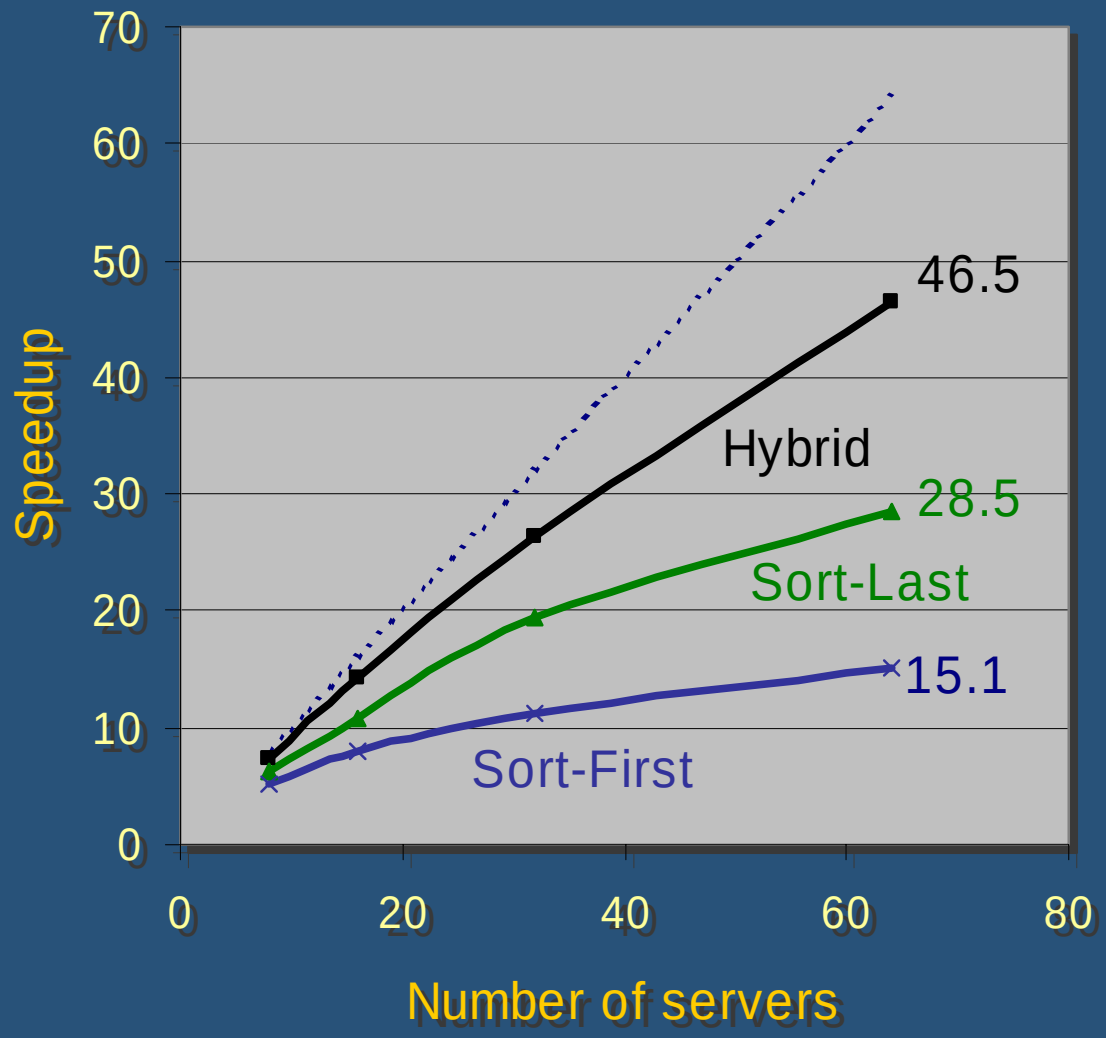


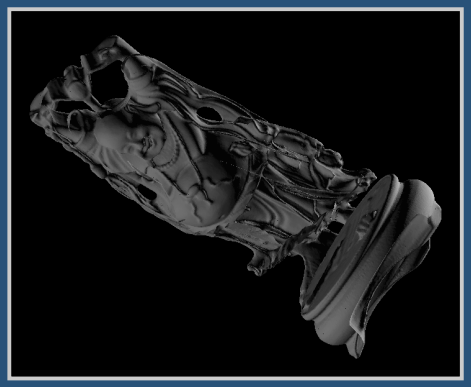
Hand - 655K

Dragon - 871K

Buddha - 1.1M

Results : Speedups

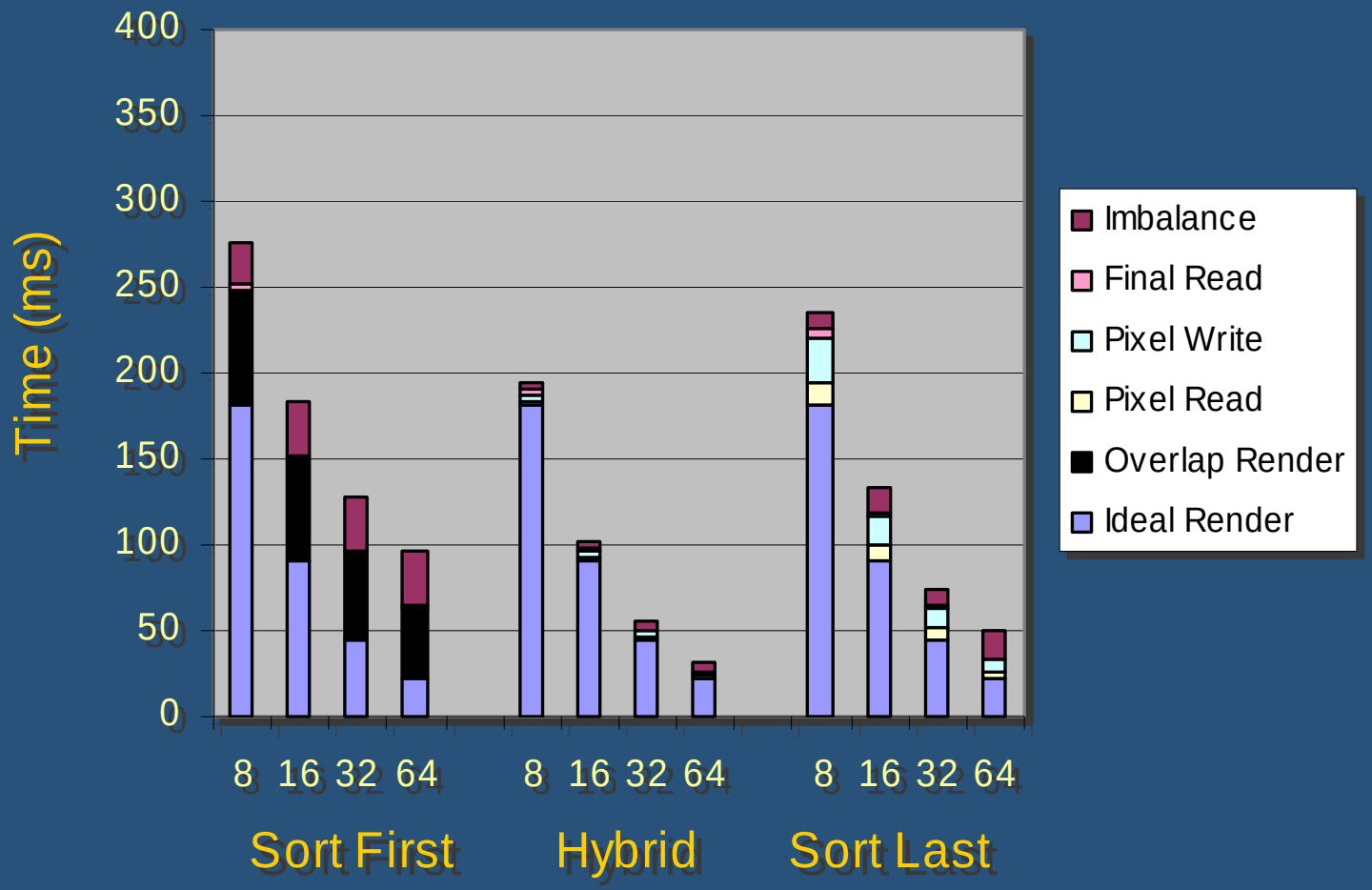




Buddha - 1.1M

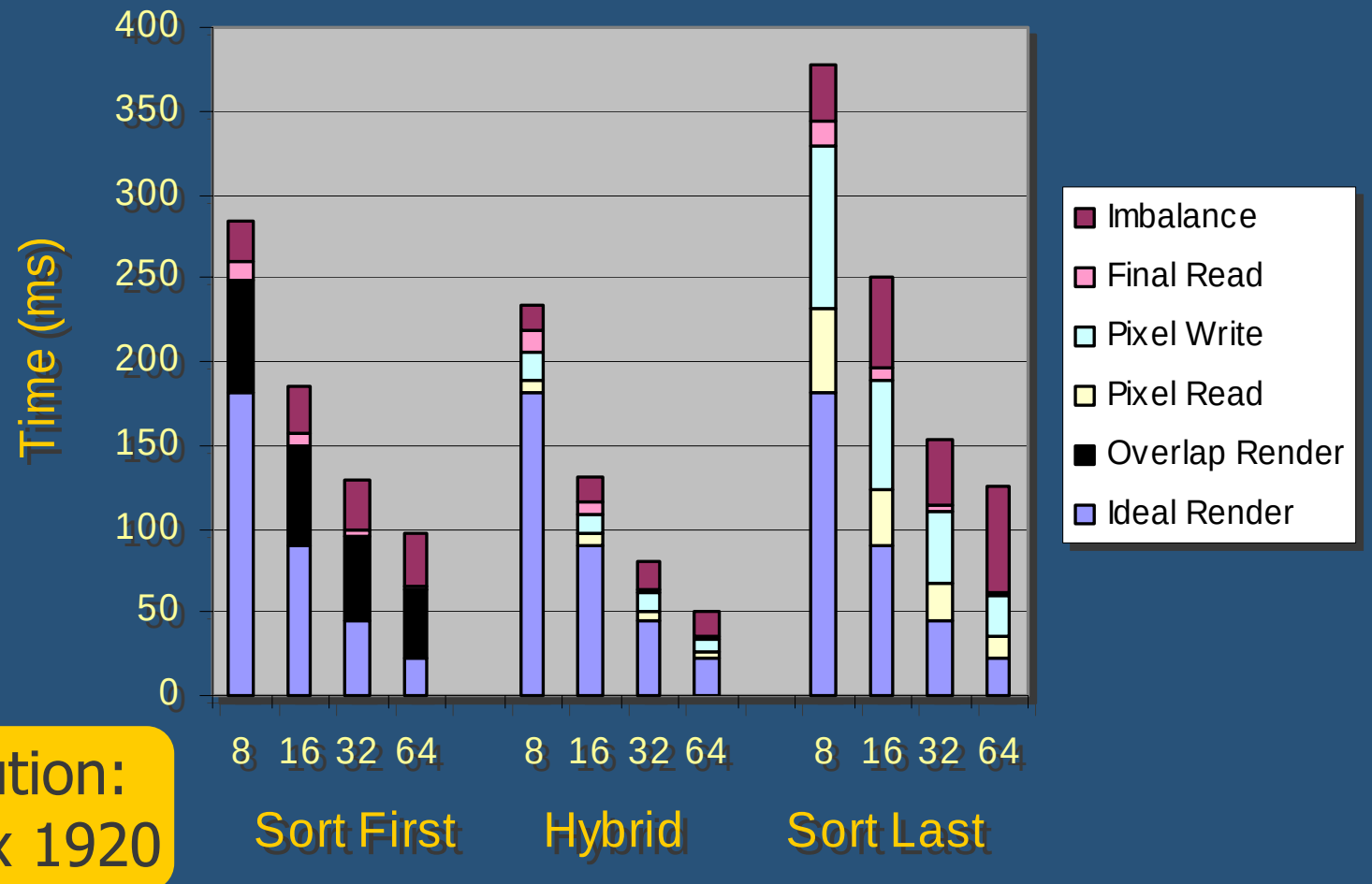
Results : Breakdowns

- Overheads are lowest for Hybrid



Results : Screen Resolution

- Hybrid even better at 2560 x 1920



Resolution:
2560 x 1920

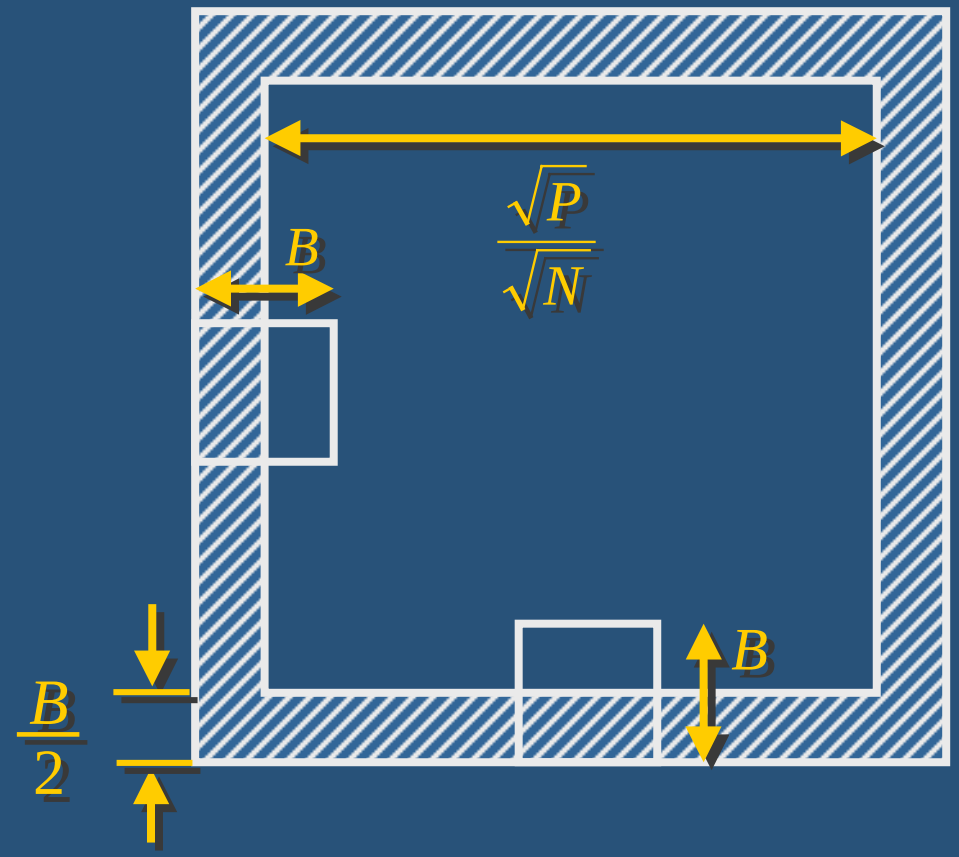
Communication Analysis

- Communication overhead analysis
- Sort-last scheme

$$\frac{P}{N}\sqrt[3]{N} = \frac{P}{\sqrt[3]{N^2}}$$

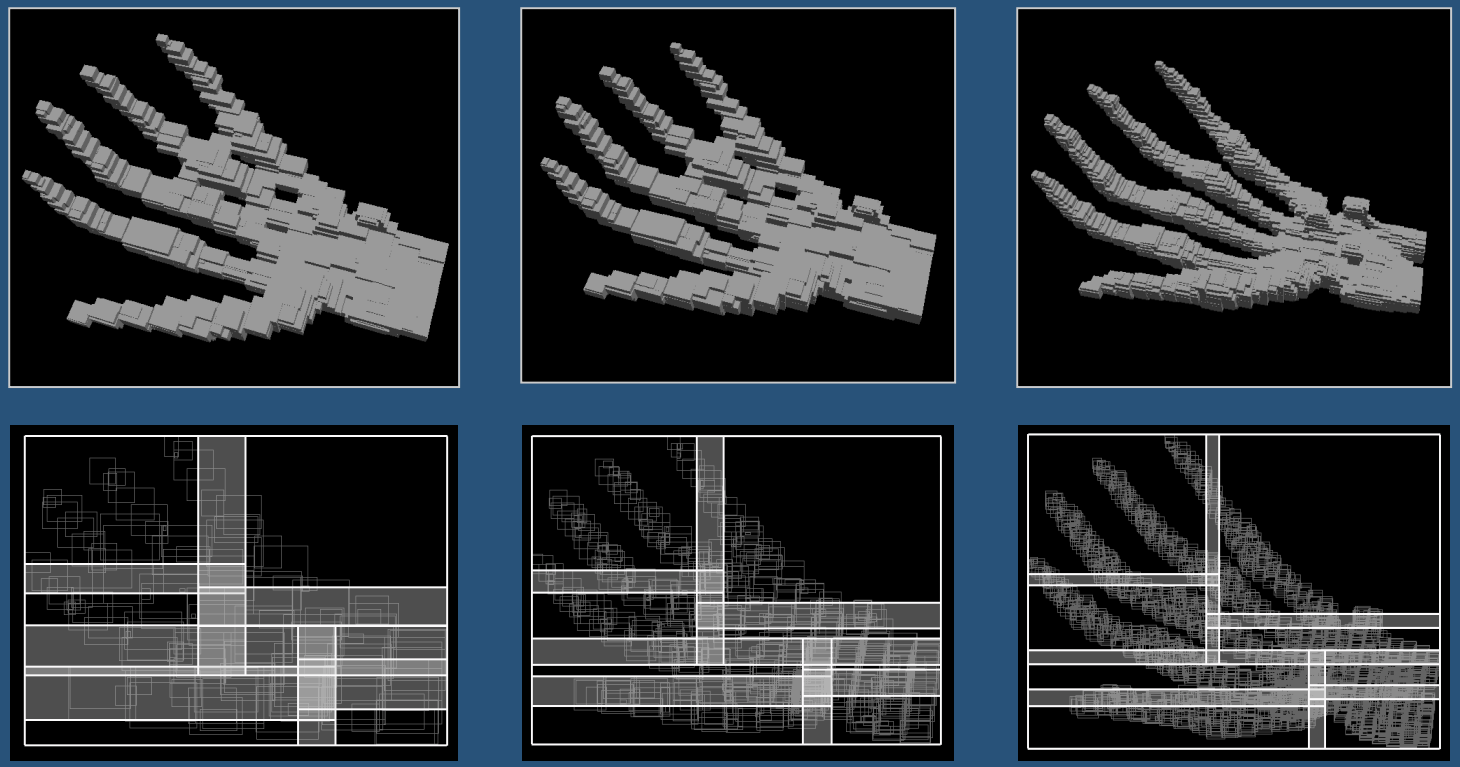
- Hybrid scheme

$$2B\frac{\sqrt{P}}{\sqrt{N}} + B^2$$



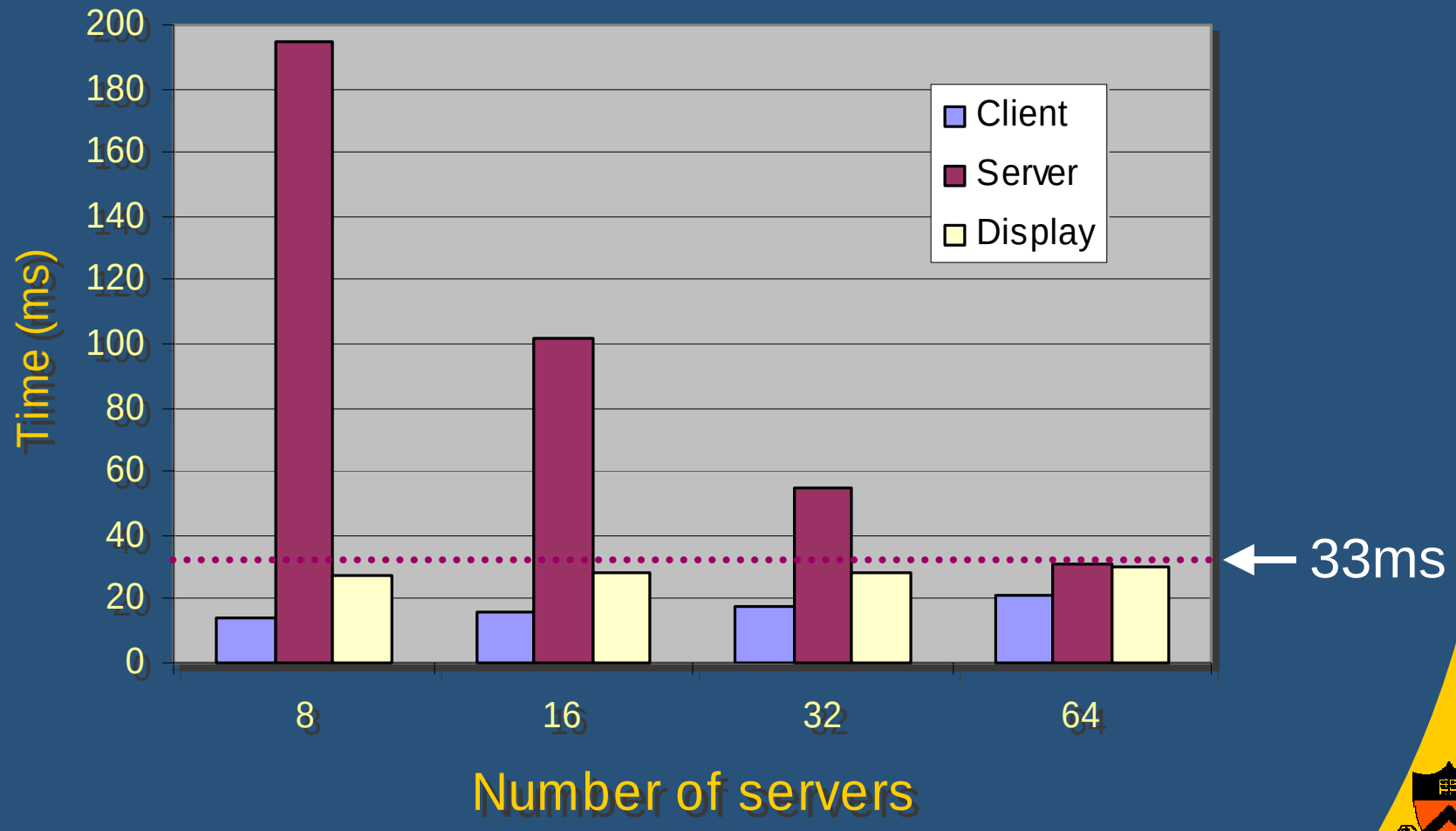
Results : Object Granularity

- Composite areas smaller as object granularity increases



Results : Feasibility

- System architecture feasible up to 64 PCs



Conclusions

- Hybrid algorithms can
  - Reduce compositing bandwidth
  - Provide good speedups (over 70% efficiency)
  - Scale to large number of servers (64)
  - Execute at interactive rates

Future Work

- Non-replicated scene database
- Dynamic models
- Immediate mode graphics interface

Acknowledgements

- Alfred P. Sloan foundation
- DOE ASCI Program
- DOE Corridor One Program
- NSF Next Generation Software Program
- NSF Infrastructure Program
- Intel Corporation
- Jiannan Zheng