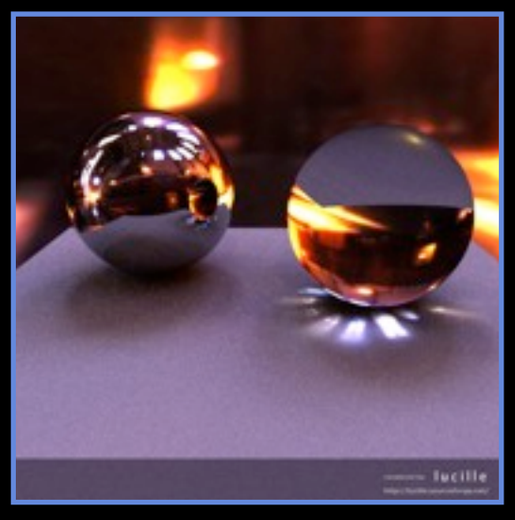
Massively Parallel Ray Tracing

Masahiro Fujita, LTE Inc.
Kenji Ono, The University of Tokyo, and Riken



CISS
Center for Research on
Innovative Simulation Software

RIKEN

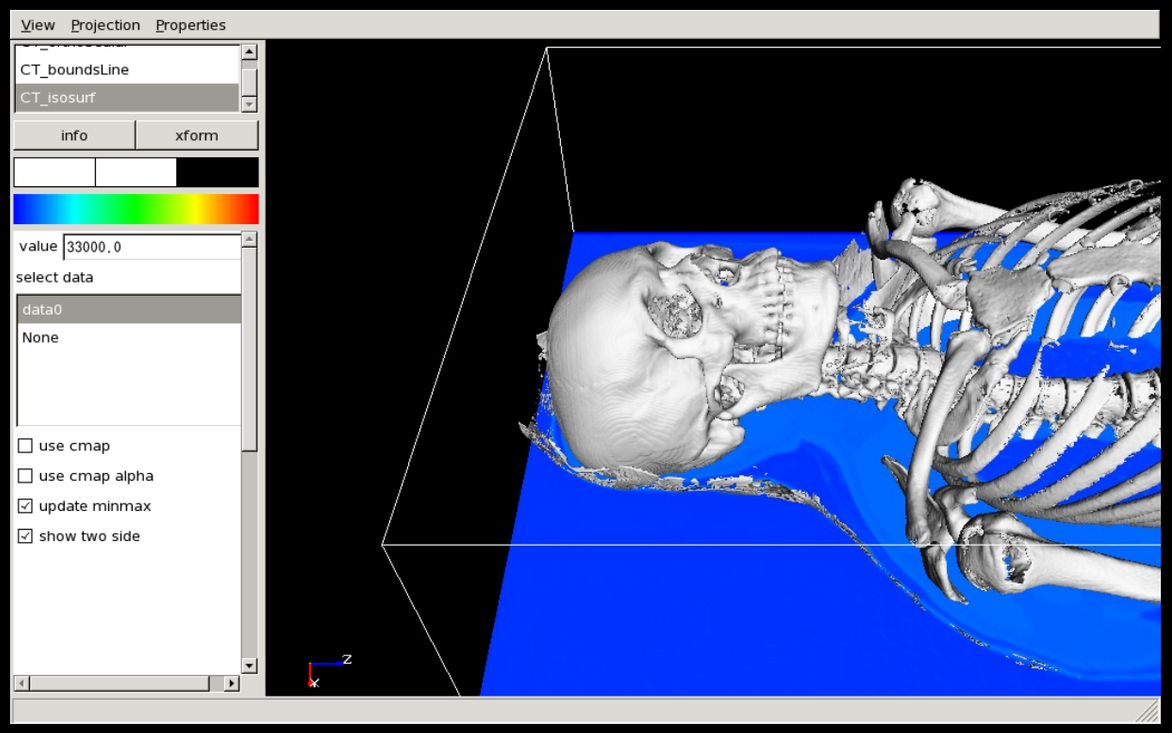
Agenda

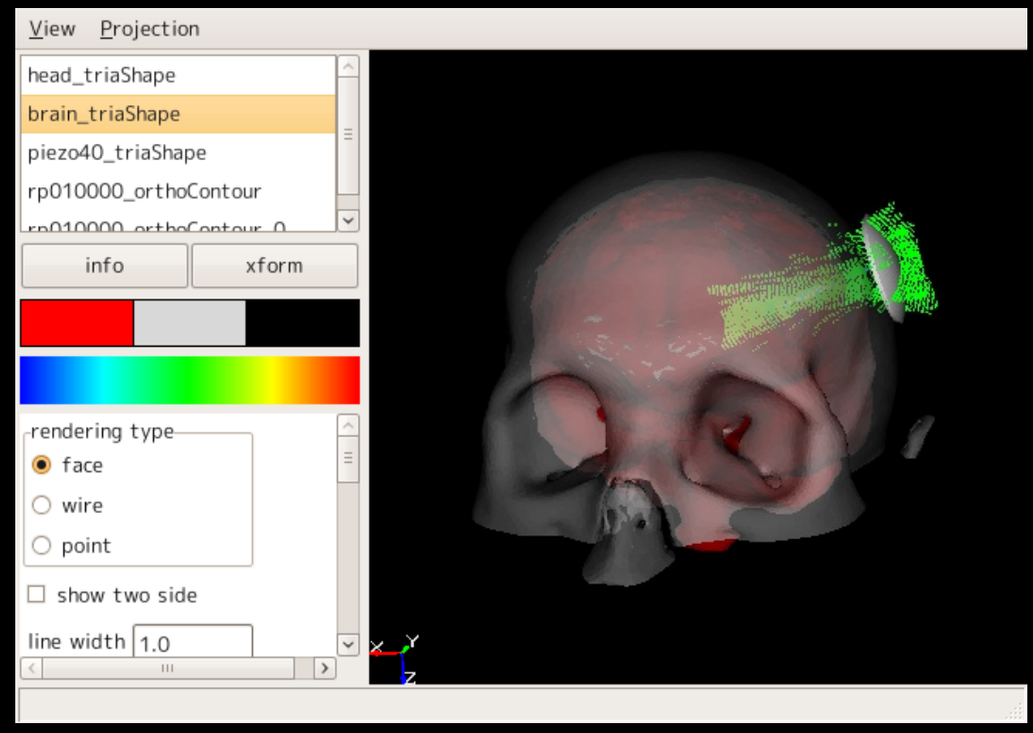
- **Motivation and Challenges**
- Architecture
- Result
- Conclusion
- Future work

Motivation

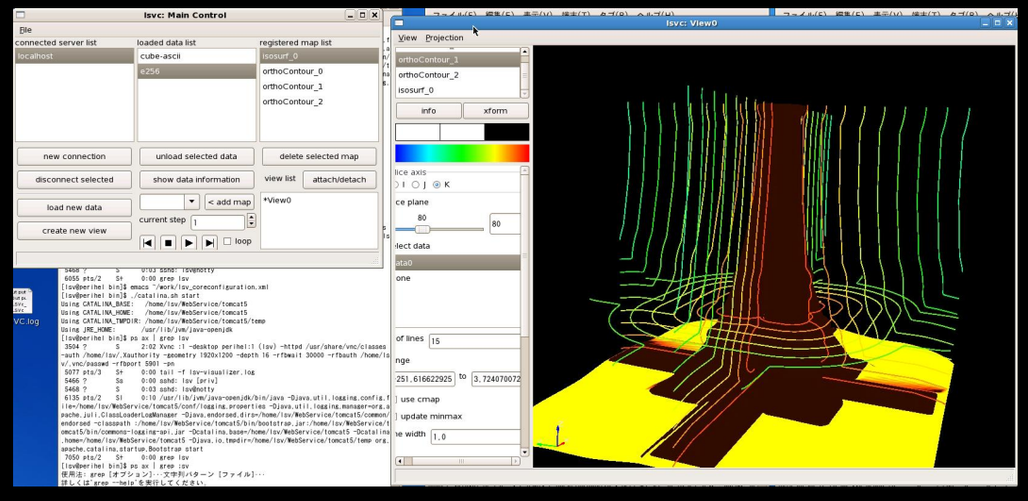
- Visualize **large scale** data
- Compute Units increase
  - Peta: 100K cores, Exa: 10M+ cores,
- Simulation data increase
  - Storage become problem
- Memory per Compute Units decrease
- Increasing demand of **visual quality**

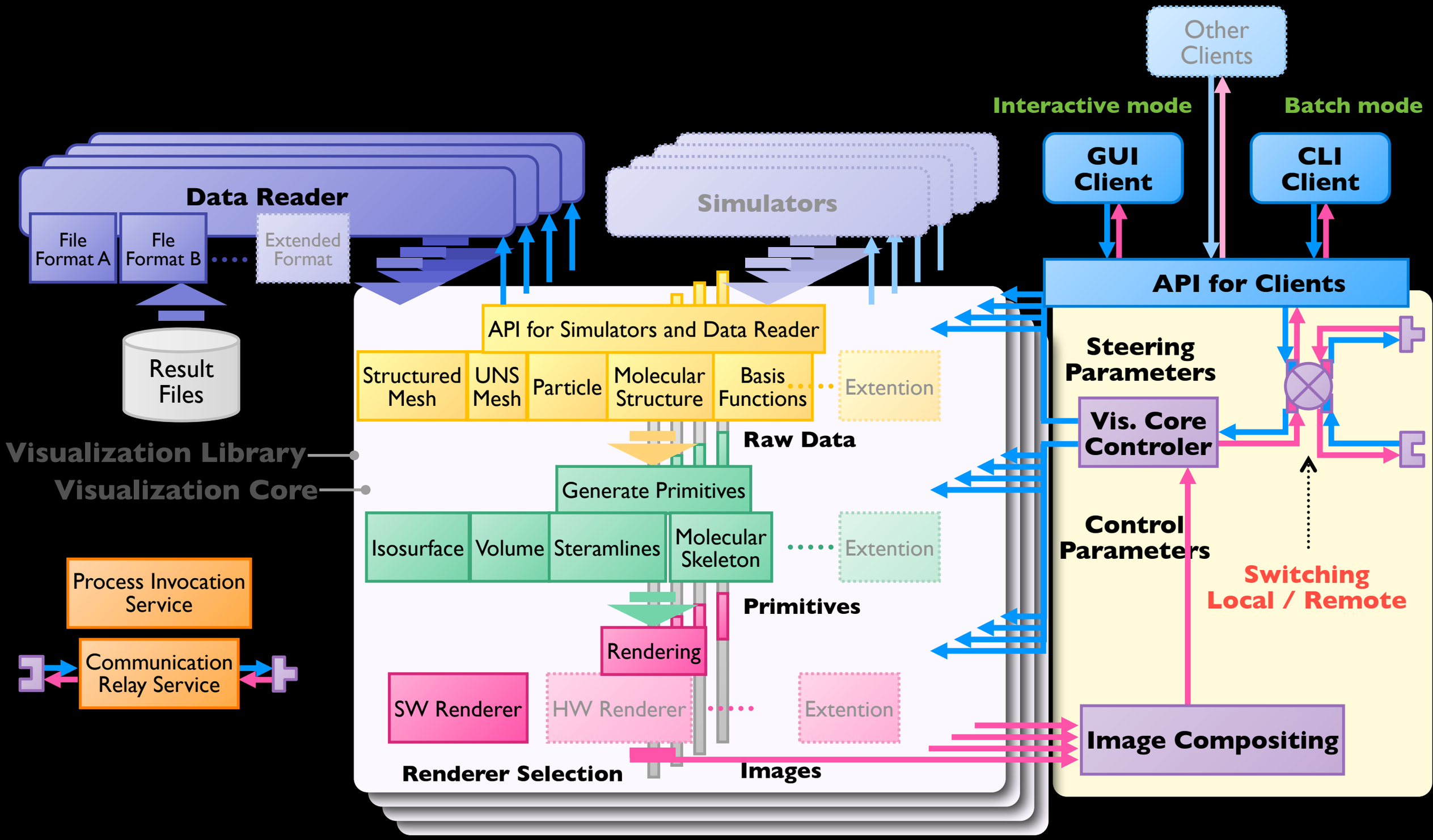
LSV





Large **S**cale
Visualization system
developing at Riken




Other Clients
Interactive mode
Batch mode
GUI Client
CLI Client
Data Reader
File Format A
Fle Format B
Extended Format
Simulators
API for Clients
API for Simulators and Data Reader
Steering Parameters
Vis. Core Controler
Structured Mesh
UNS Mesh
Particle
Molecular Structure
Basis Functions
Extention
Raw Data
Generate Primitives
Isosurface
Volume
Steramlines
Molecular Skeleton
Extention
Primitives
Control Parameters
Switching Local / Remote
Rendering
SW Renderer
HW Renderer
Extention
Renderer Selection
Images
Image Compositing
Result Files
Visualization Library
Visualization Core
Process Invocation Service
Communication Relay Service

Other Clients

Interactive mode

Batch mode

GUI Client

CLI Client

Data Reader

File Format A

Fle Format B

Extended Format

Result Files

Simulators

API for Clients

API for Simulators and Data Reader

Structured Mesh

UNS Mesh

Particle

Molecular Structure

Basis Functions

Extention

Raw Data

Generate Primitives

Isosurface

Volume

Steramlines

Molecular Skeleton

Extention

Primitives

Rendering

SW Renderer

HW Renderer

Extention

Renderer Selection

Images

Steering Parameters

Vis. Core Controler

Control Parameters

Switching Local / Remote

Image Compositing

Visualization Library

Visualization Core

Process Invocation Service

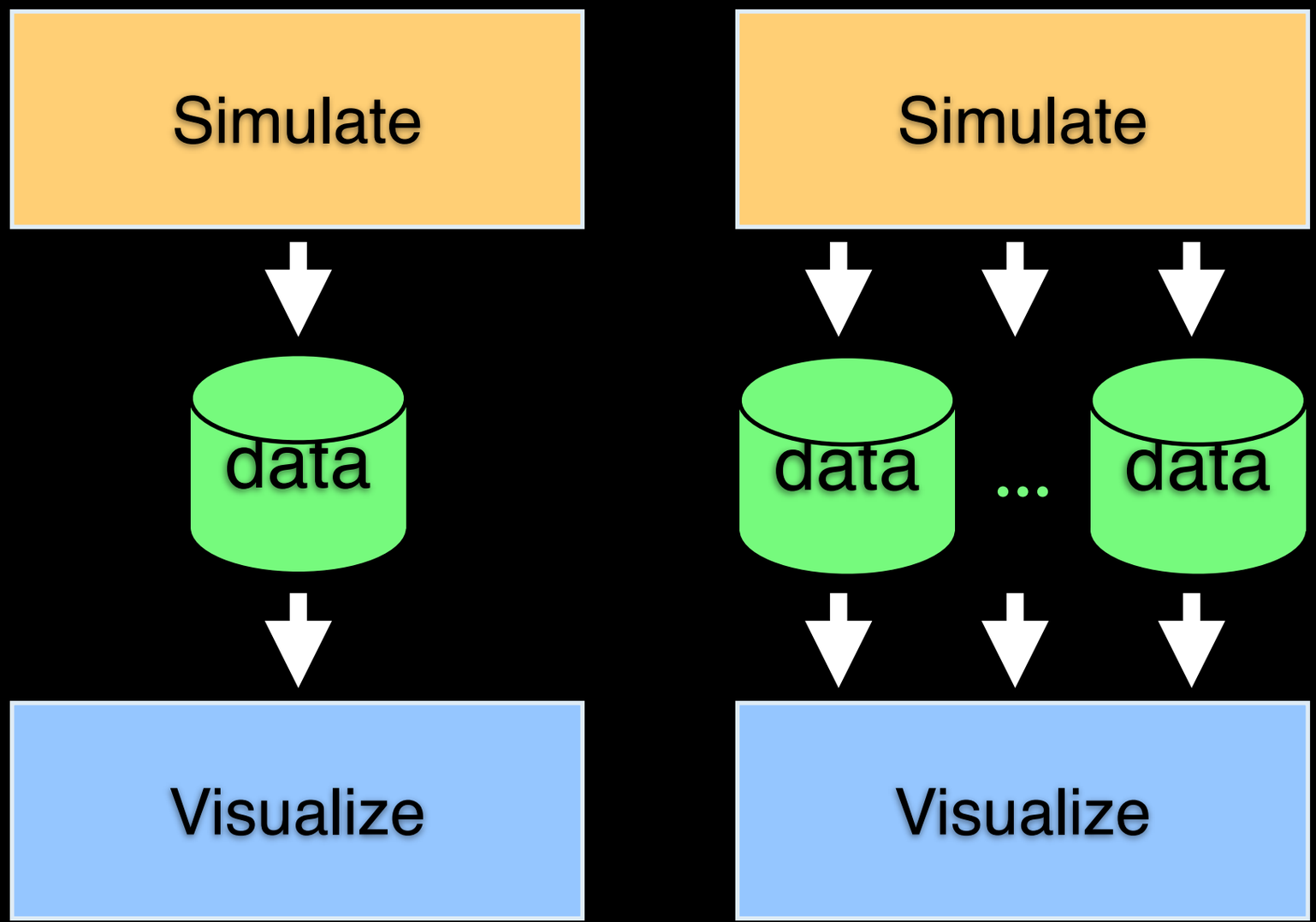
Communication Relay Service

Compute Units

~100

~10,000

~1,000,000



Simulate
&
Visualize

Compute Units

~100

~10,000

Our focus

000

Simulate

Simulate

data

data

...

data

Simulate & Visualize

Visualize

Visualize

Key components

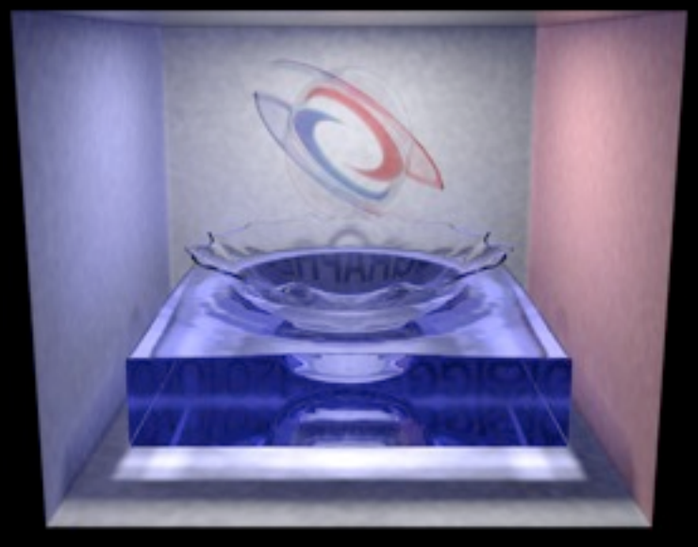
Out-of-core

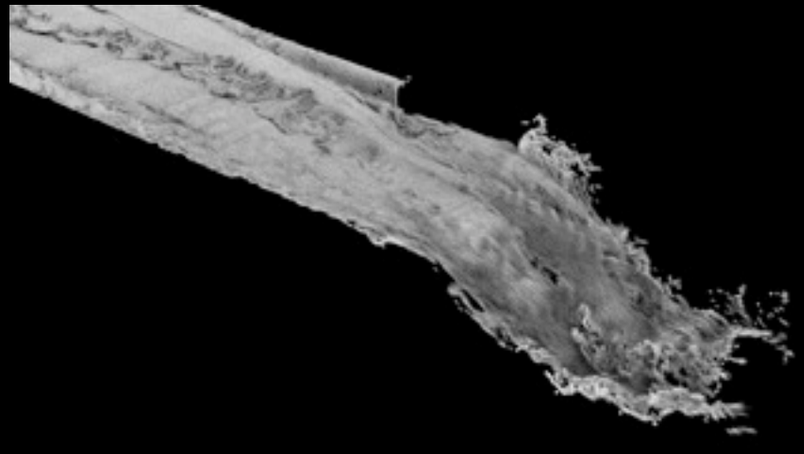
Massively Parallel

Ray Tracing

Why raytracing?

- **Visual quality**. Better than OpenGL!
- Scalable than OpenGL
- Correct handling of transparency, reflection, indirect illumination
- Runs on many **CPU architectures**

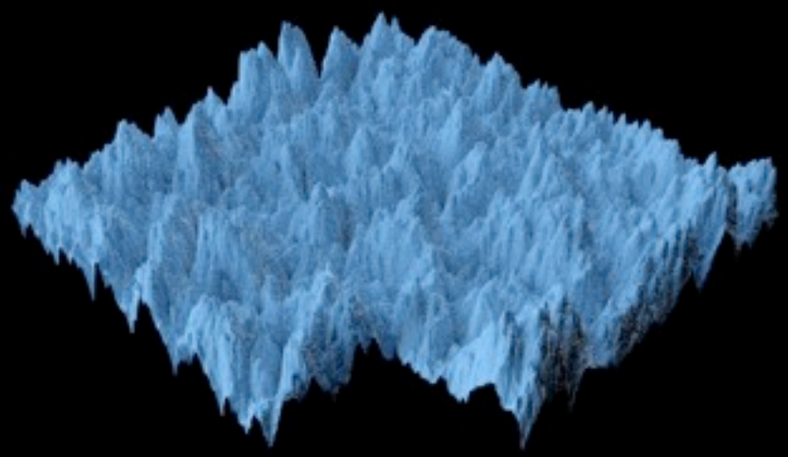




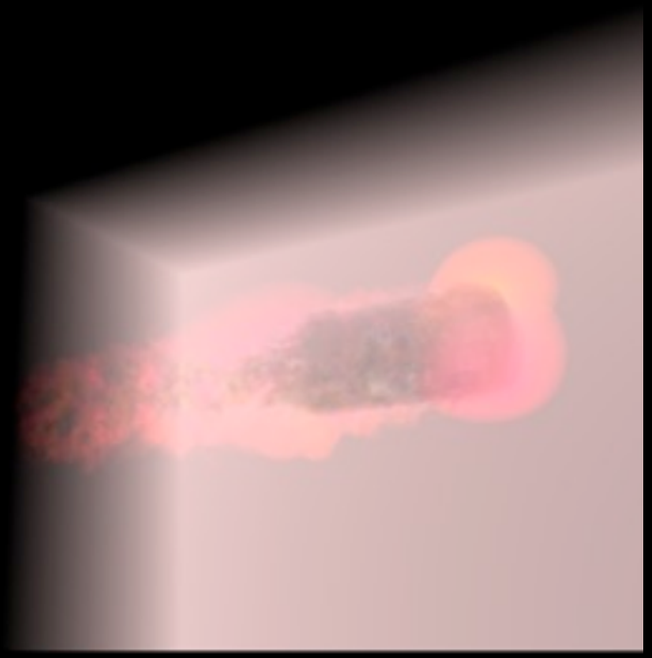
Sponza model: (C) Marko Dabrovic

Primitives

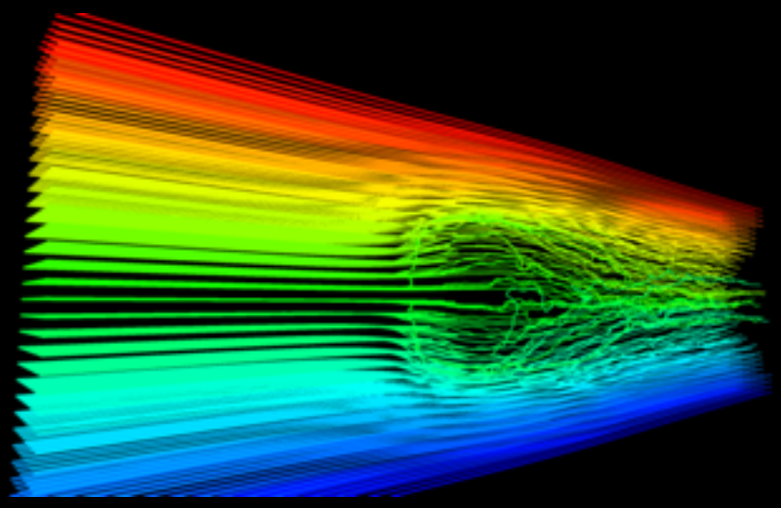
Polygons



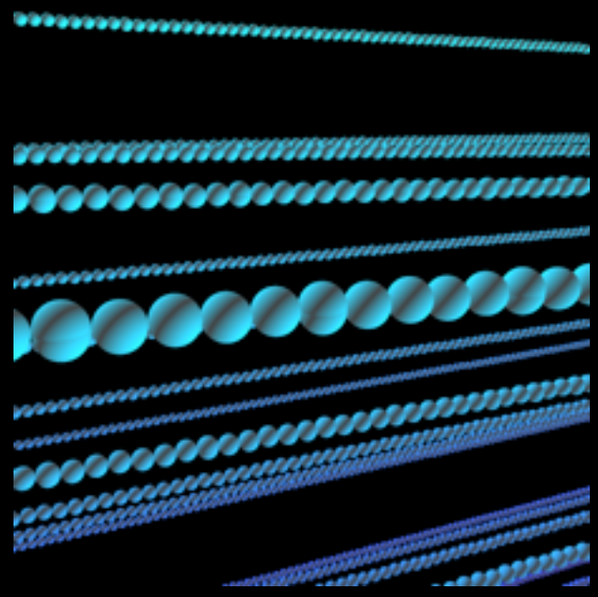
Volumes



Curves



Particles



Example

Challenges

- Parallel raytracing algorithm itself for 1000+ compute units is challenging
- Limited memory per compute unit.
  - We assume 1GB per compute unit.
- 1000+ compute units
  - MPI problem arises, parallel performance, etc.

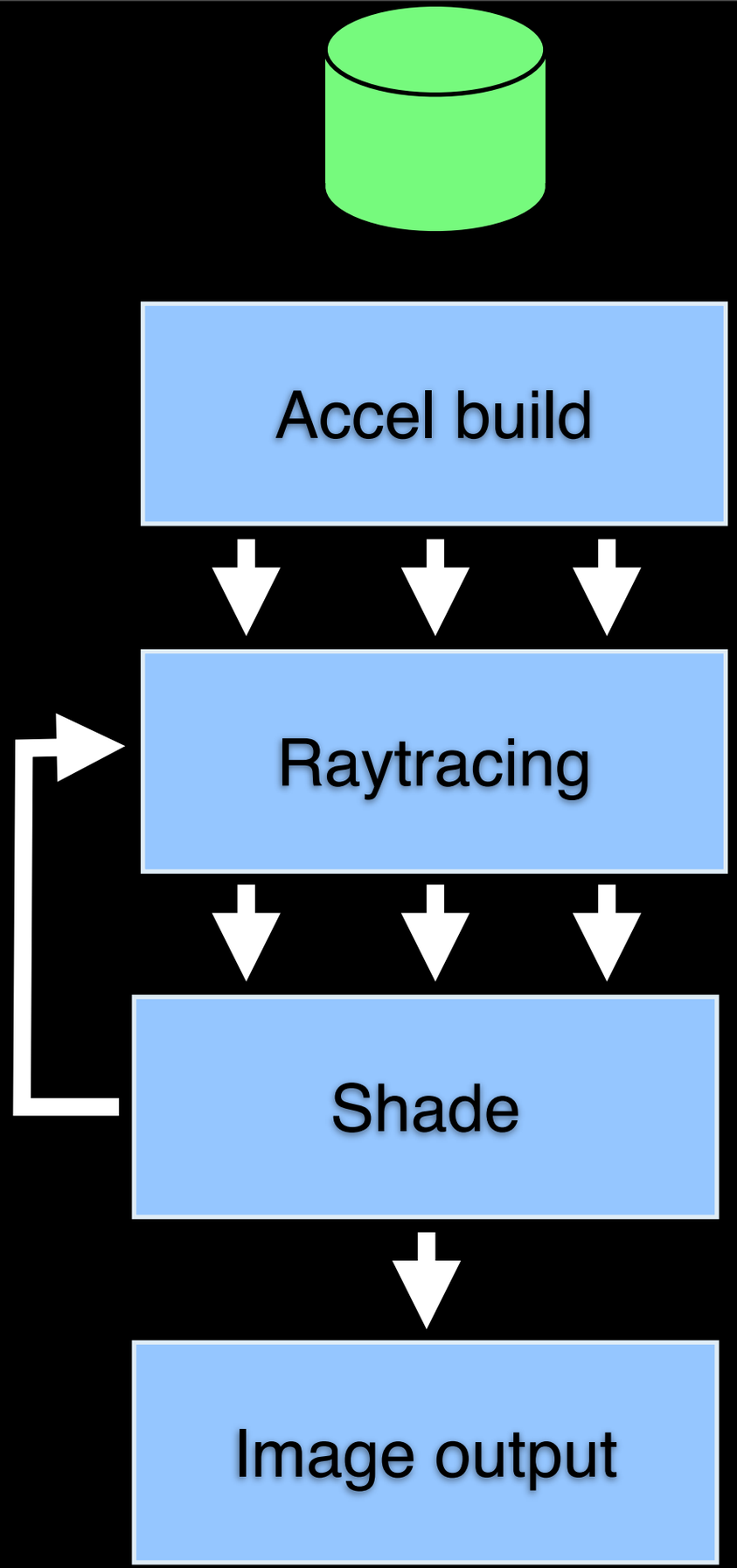
Agenda

- Motivation and Challenges
- **Architecture**
- Result
- Conclusion
- Future work

Architecture

- Out-of-core raytracing
  - acceleration building, traversal of prim
- Exchange rays between Compute Units
  - Enables correct indirect illumination

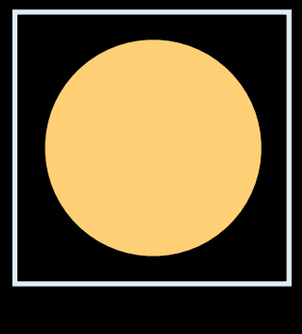
+Indirect

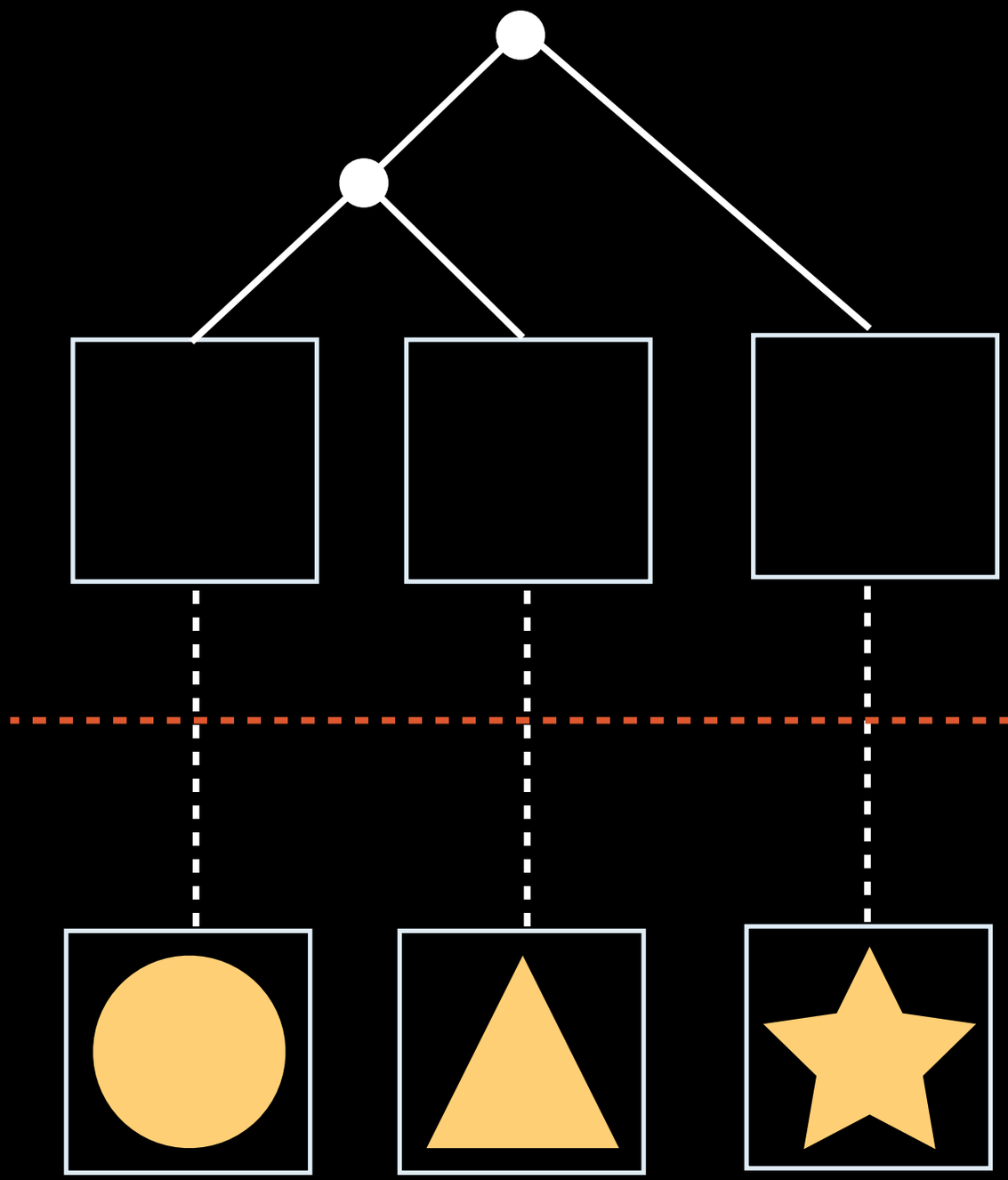

Accel build
Raytracing
Shade
Image output

Acceleration structure

- 2-level BVH(Bounding Volume Hierarcy)
  - Toplevel: Bounding information
  - Bottomlevel: Primitive data, BVH data

Scene



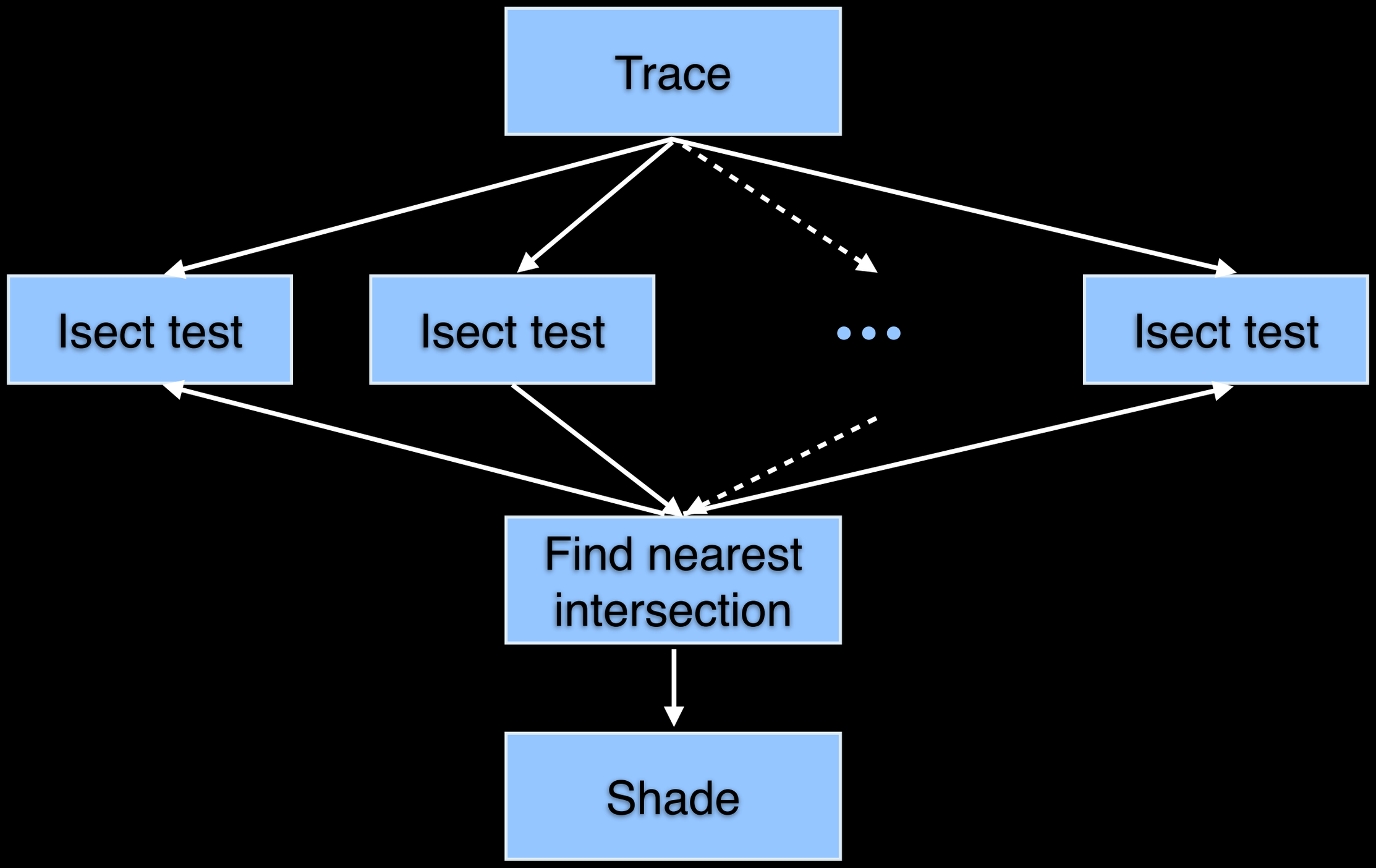


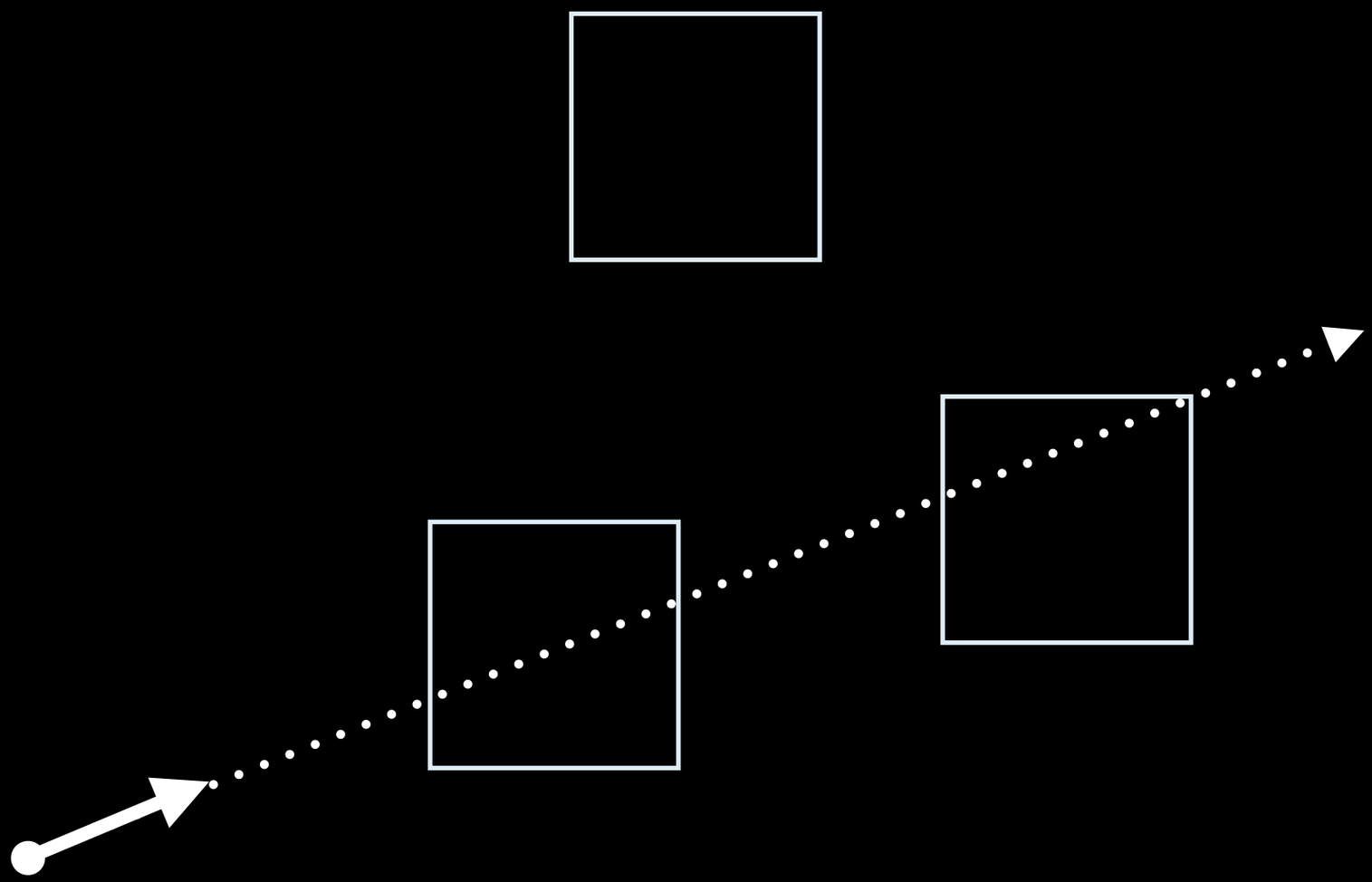
Toplevel

BoundingBox
data
~ 100KB
All CUs share

Bottomlevel

Primitive &
Acceleration data
~500MB
per CU


Trace
Isect test
Isect test
• • •
Isect test
Find nearest intersection
Shade



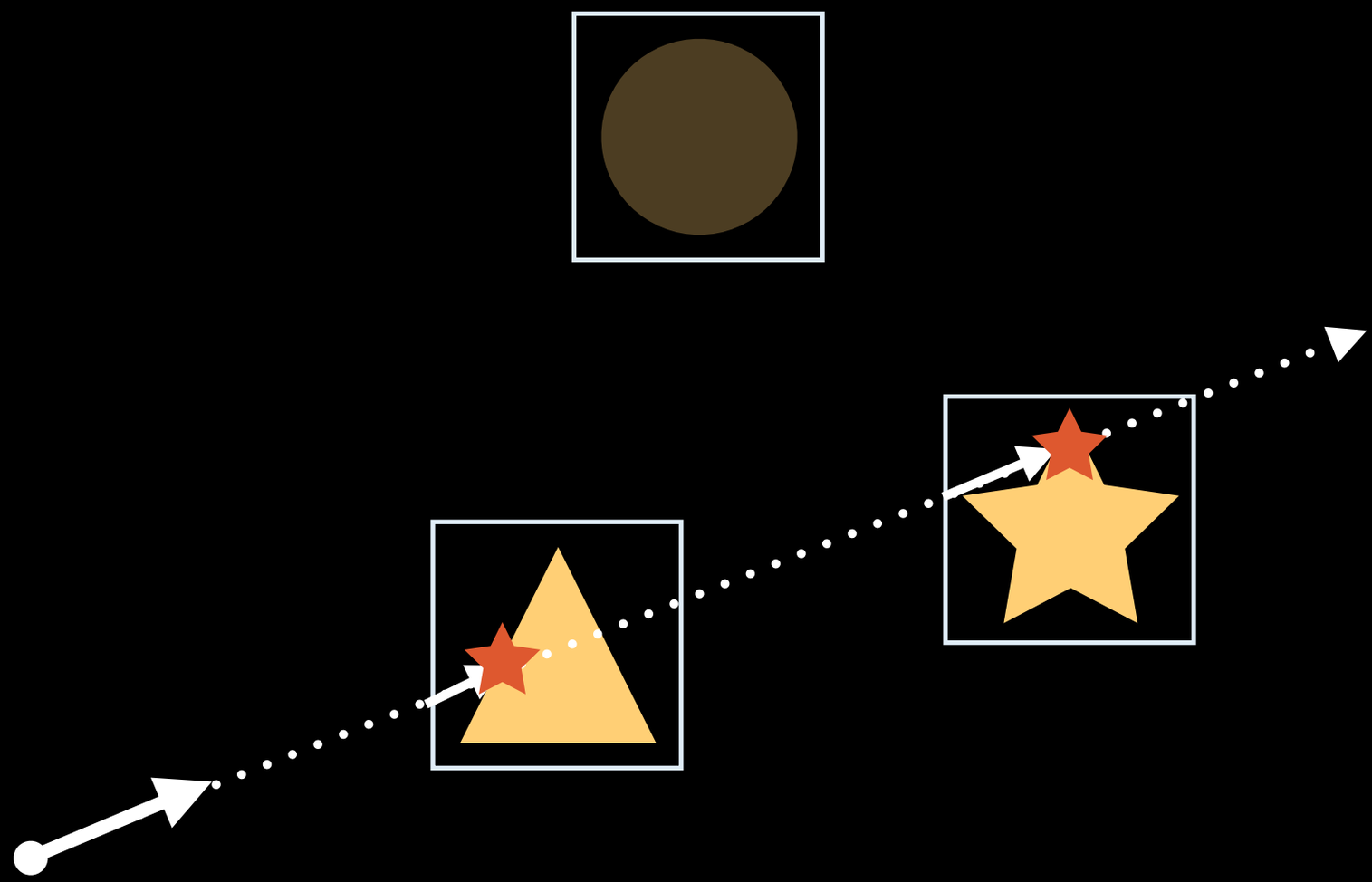
Toplevel

Trace

Isect test

Find nearest intersection

Shade



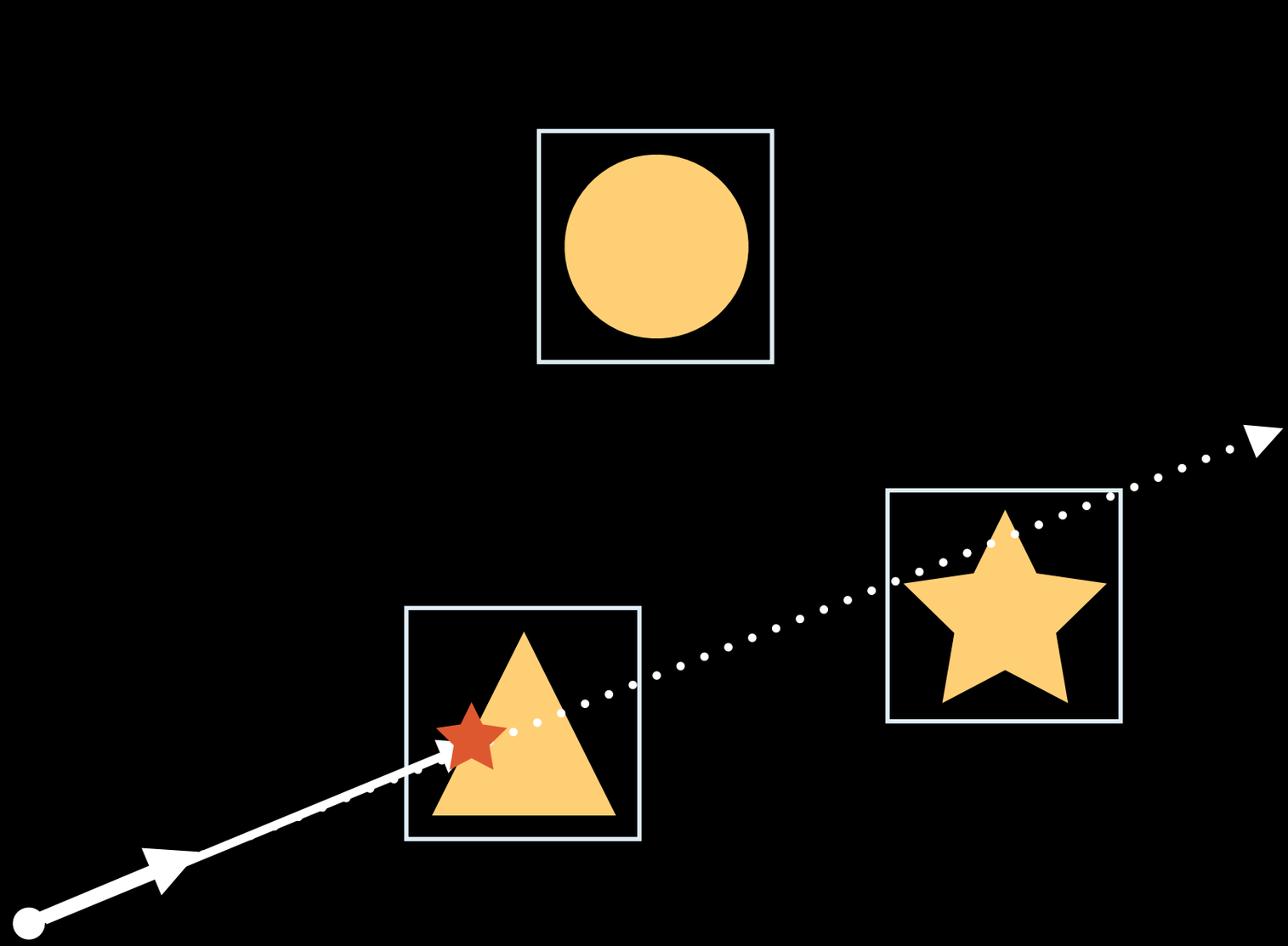
Bottomlevel

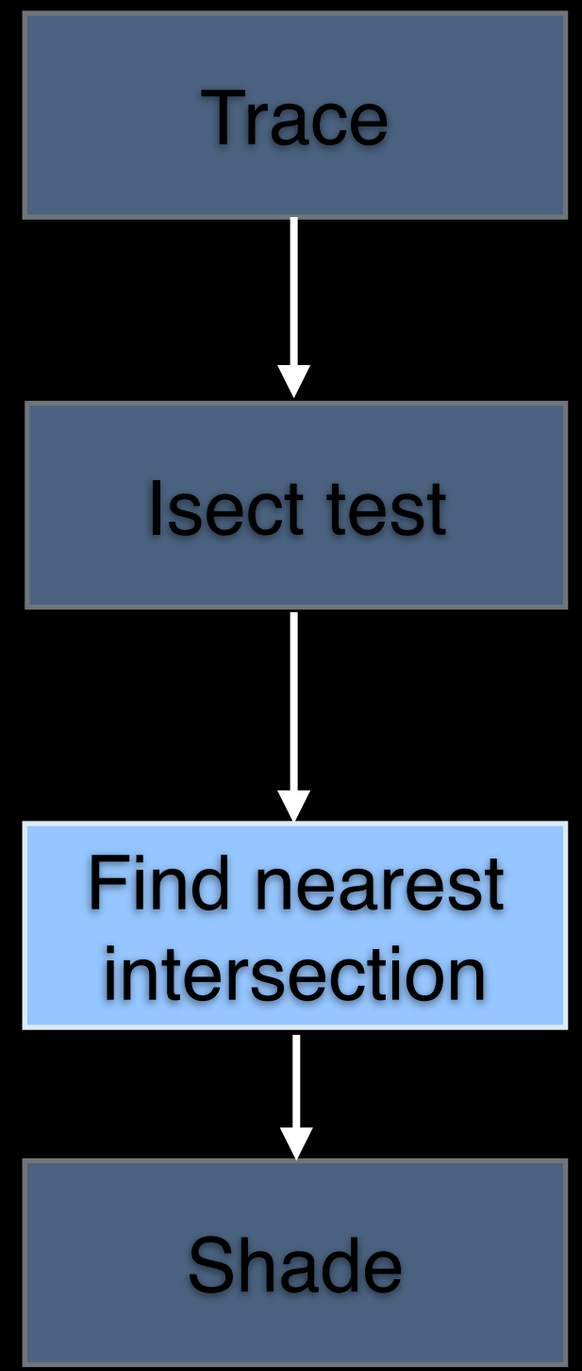
Trace

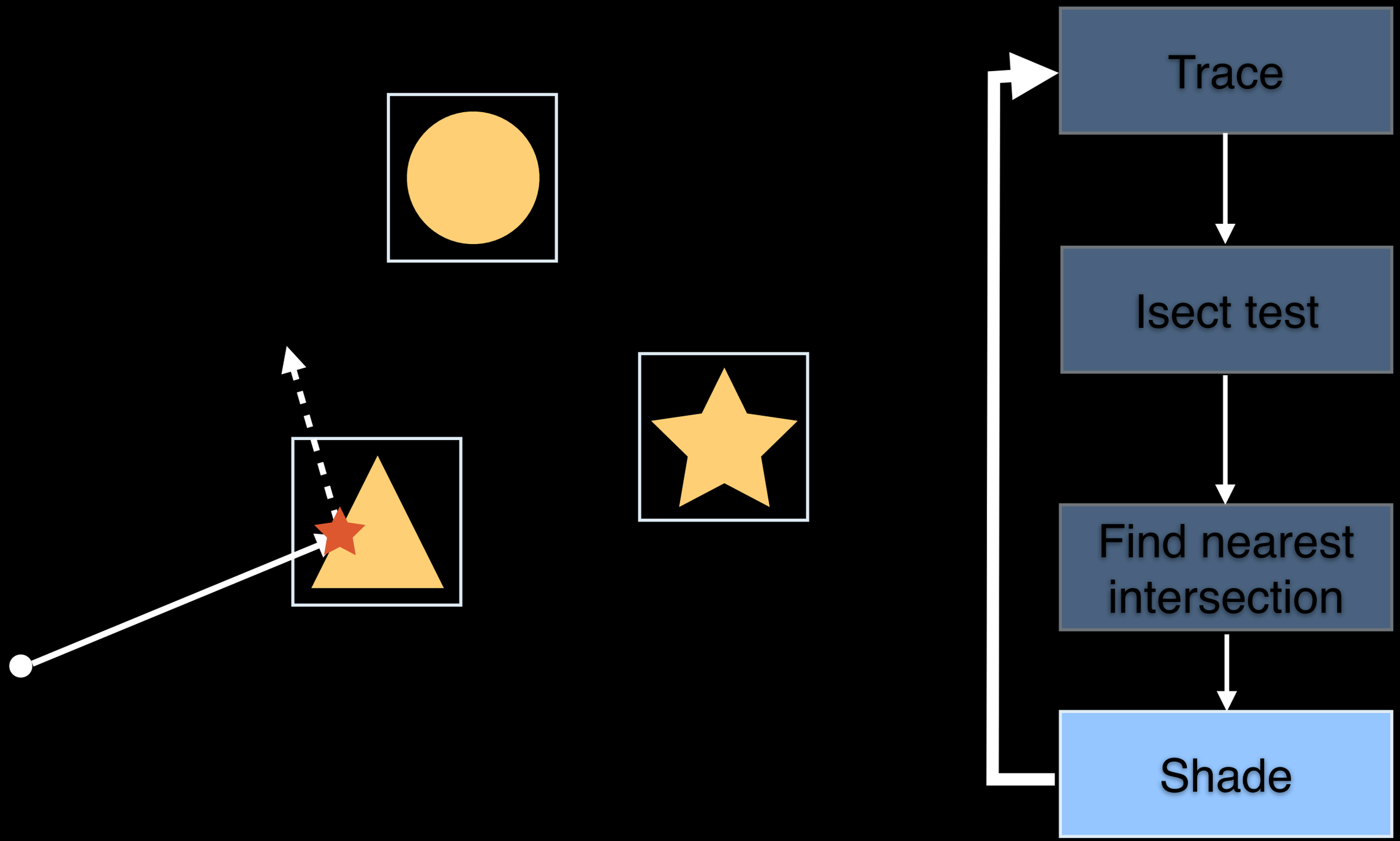
Isect test

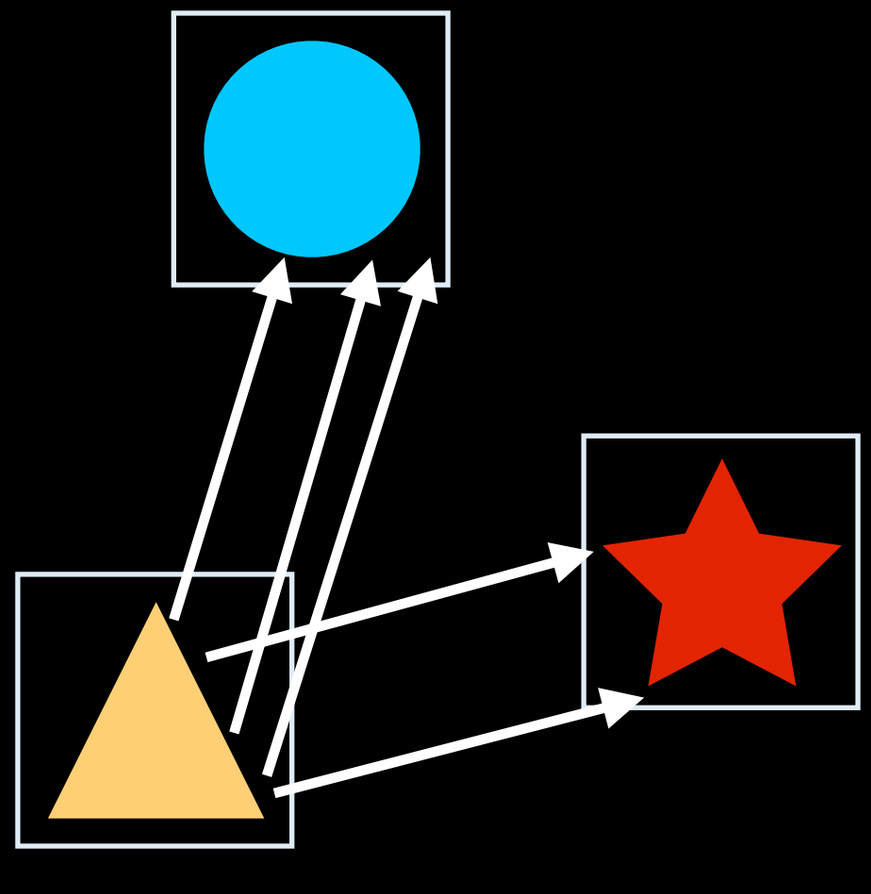
Find nearest intersection

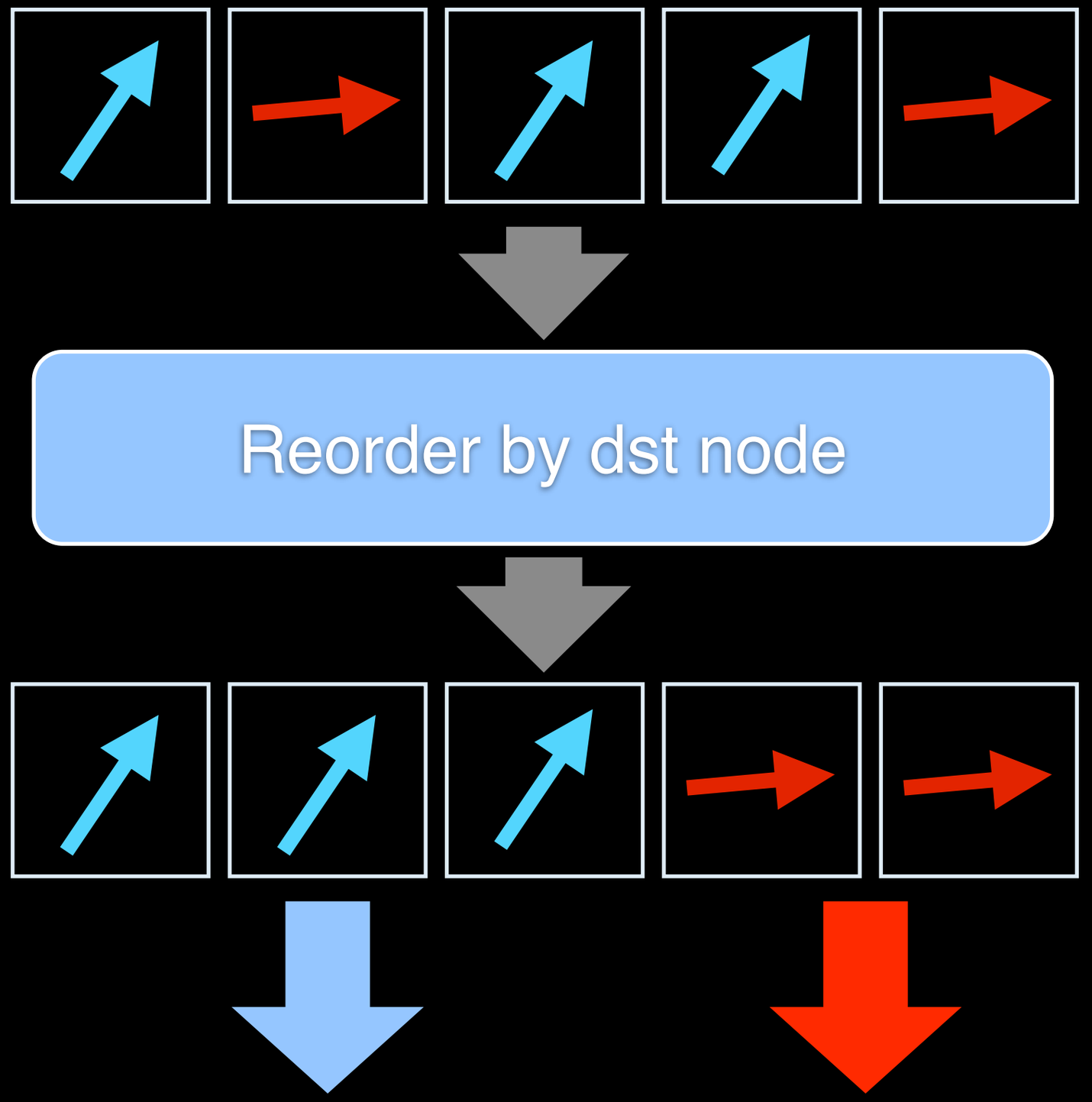
Shade




Trace
Isect test
Find nearest intersection
Shade


Trace
Isect test
Find nearest intersection
Shade



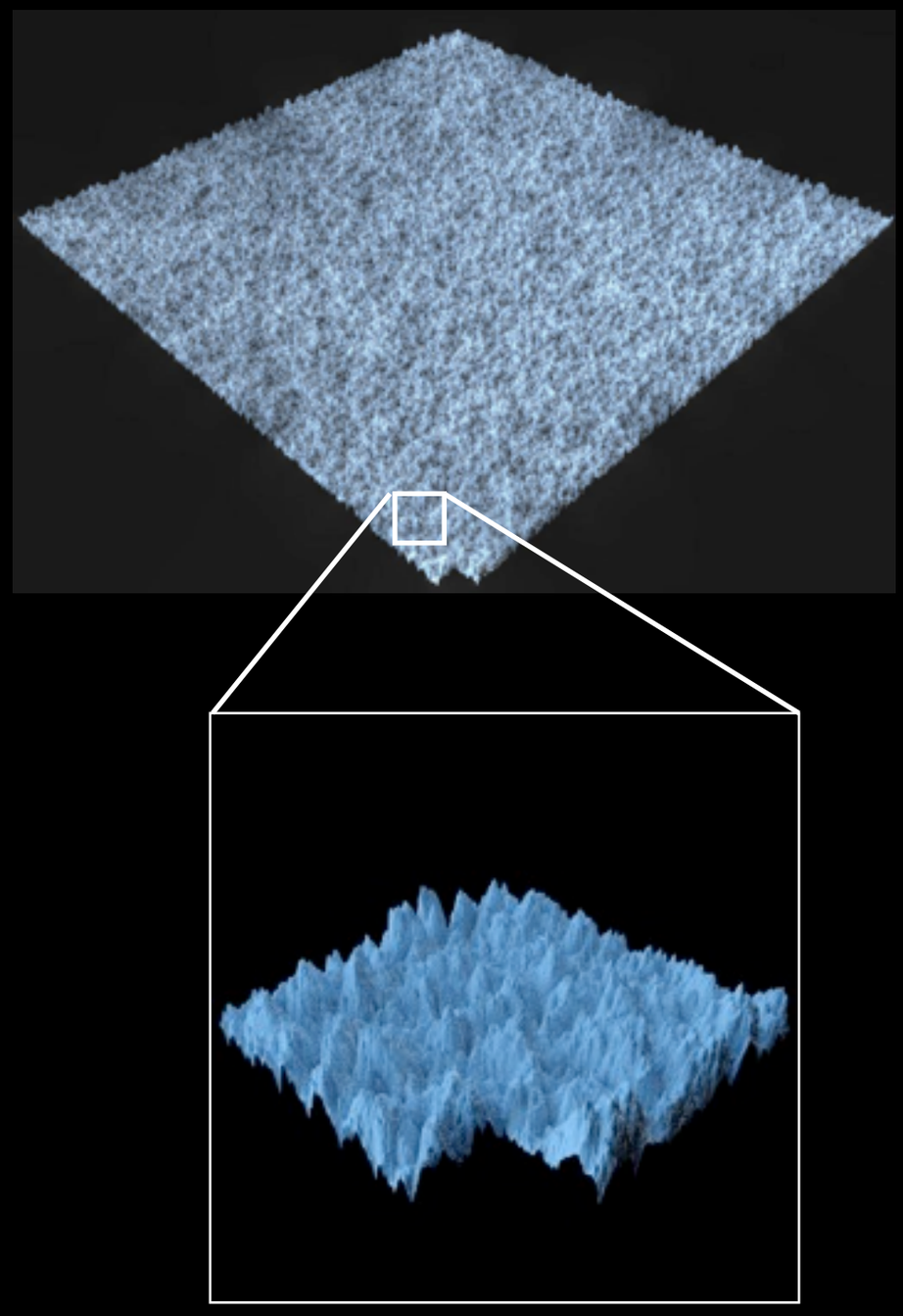

Reorder by dst node

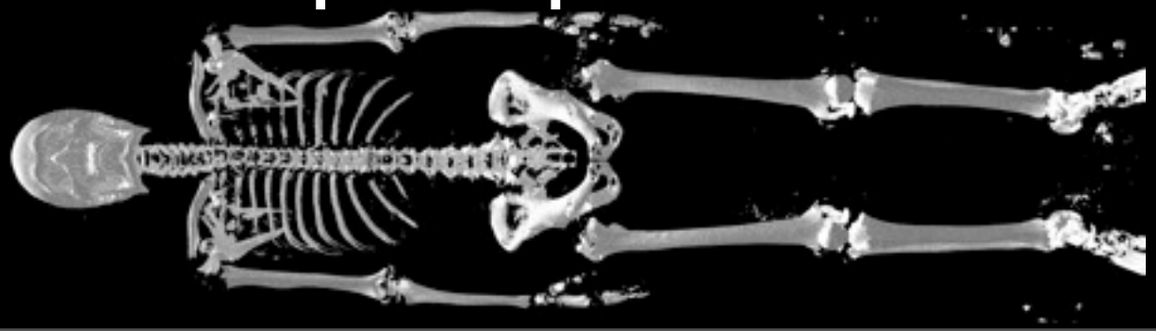
Agenda

- Motivation and Challenges
- Architecture
- **Result**
- Conclusion
- Future work

Result

- Terrain
  - Procedural terrain
  - 1 path/pixel
- CT
  - Volume -> poly
  - 1 path/pixel



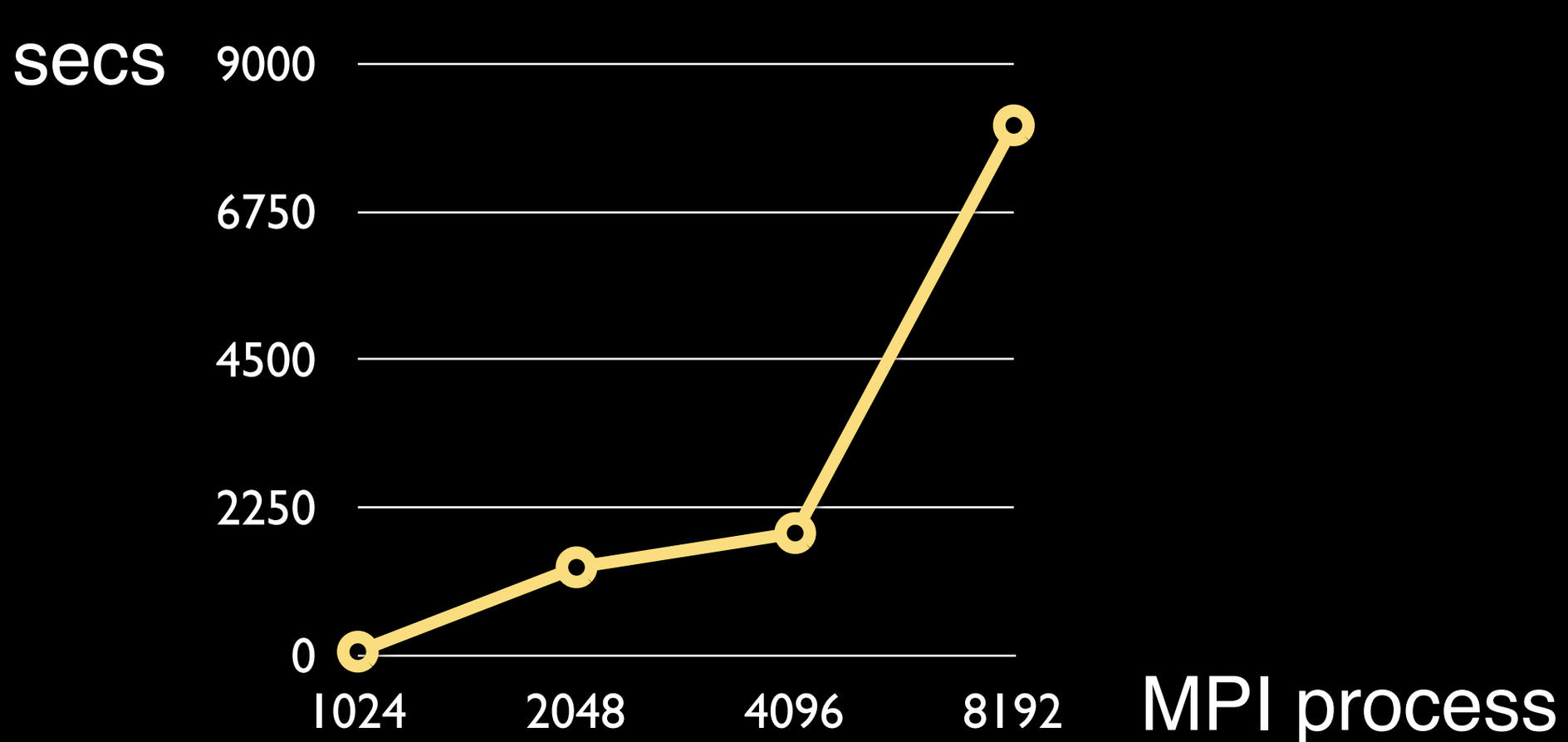


Measured on RICC

- x86 Cluster at Riken
  - 1024 nodes
  - 4 Cores x 2 / node
  - 12GB / node
  - 8192 cores in total, 1.5GB/core
  - InfiniBand DDR
  - MPI

Terrain

- Generates 2M polygon per CU
  - 1024: 2B, 4096: 8B, 8192: 16B polys



Performance factor

- # of surfaces visible to screen
- # of BVH node hits in Toplevel BVH traversal
- # of computation units(MPI processes)
  - N^2 communication
- So, render time =~ N^2 is expected result

CT

- 100 GB volume data input
  - Generate isosurf poly from volume
- 1024 MPI processes
  - 14.34 secs for out-of-core mesh build
  - 26.87 secs for render.

Discussion

- Unstructured NxN communication
  - MPI gather/scatter **doesn't work** well
    - Memory soon exhausts
    - Async, dynamic communication: Tried **ADLB**, but didn't scale
- Simply MPI **sendrecv** is only working solution so far.
- Hierarchical communication will improve the performance

Discussion, cont.

- Memory per process(8,192 MPI procs)
  - 400~500 MB for MPI library
  - 200~300 MB for Prim/BVH
  - **100~200** MB for ray data
    - Avg 100~200 rays/process at max
    - Need a frequent ray exchange to reduce memory(MPI comm increases)

Agenda

- Motivation and Challenges
- Architecture
- Result
- **Conclusion**
- Future work

Conclusion

- Out-of-core, massively parallel raytracing architecture
  - Confirmed it works up to 8,192 MPI processes
- Memory and MPI are the bottleneck for massive environment
  - Need to find new way for 10k+ compute units

Agenda

- Motivation and Challenges
- Architecture
- Result
- Conclusion
- **Future work**

Future work

- Simulate, then Visualize
  - Possible architecture for **Exa** era
- Porting to **K computer**
  - Initial trial successes
  - K specific optimization remains
- Integrate fully into LSV framework
  - Partially integrated already

Acknowledgements

- LSV team
- Riken
  - RICC cluster
  - K computer
- FOCUS for x86 cluster
- Simon Premoze

References

- Matt Pharr, Craig Kolb, Reid Gershbein, and Pat Hanrahan, **Rendering Complex Scenes with Memory-Coherent Ray Tracing**, Proc. SIGGRAPH 1997

- Johannes Hanika, Alexander Keller, Hendrik P. A. Lensch: **Two-level ray tracing with reordering for highly complex scenes**. Graphics Interface 2010: 145-152

- Kirill Garanzha, Alexander Bely, Simon Premoze, Vladimir Galaktionov, **Out-of-core GPU Ray Tracing of Complex Scenes**. Technical talk at SIGGRAPH 2011

Thank you!

- syoyo@lighttransport.com