Lecture 24: Visualization Pipeline

Dec 5, 2024

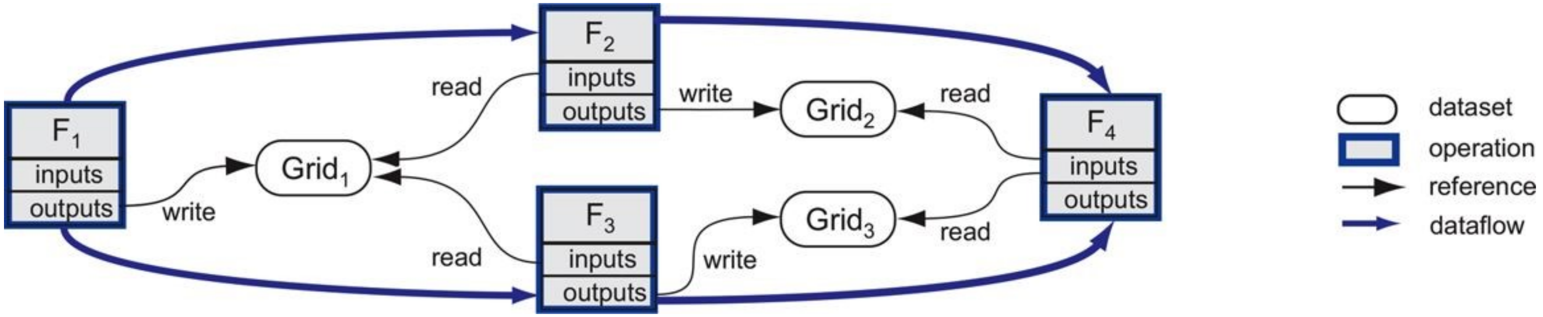
Won-Ki Jeong

(wkjeong@korea.ac.kr)

KOREA UNIVERSITY

Visualization Pipeline

- **A dataflow network** in which computation is described as a collection of executable **modules** that are connected in a **directed graph** representing data movement between modules



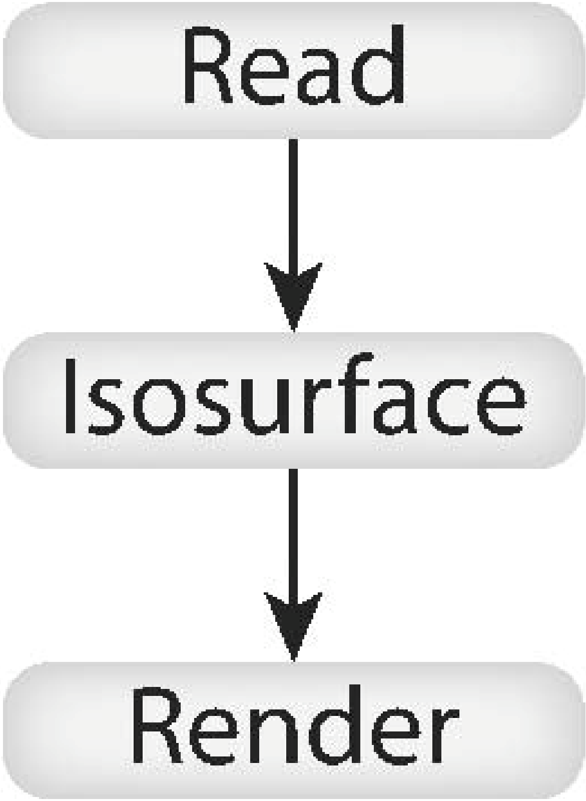
Modules

- Basic building blocks (functional units) of a pipeline
- Encapsulating algorithms
- Having generic connection ports
  - Inputs and outputs
- Interchangeable
  - As long as input and output data is compatible
- Node in a directed graph

Types of Modules

- Source
  - Producing data
  - File reader, synthetic data generator
- Sink
  - Accept data as an input and no further result
  - File writers, rendering modules
- Filter
  - At least one input and one output
  - Transform data

The Simplest Pipeline Example



Read

Isosurface

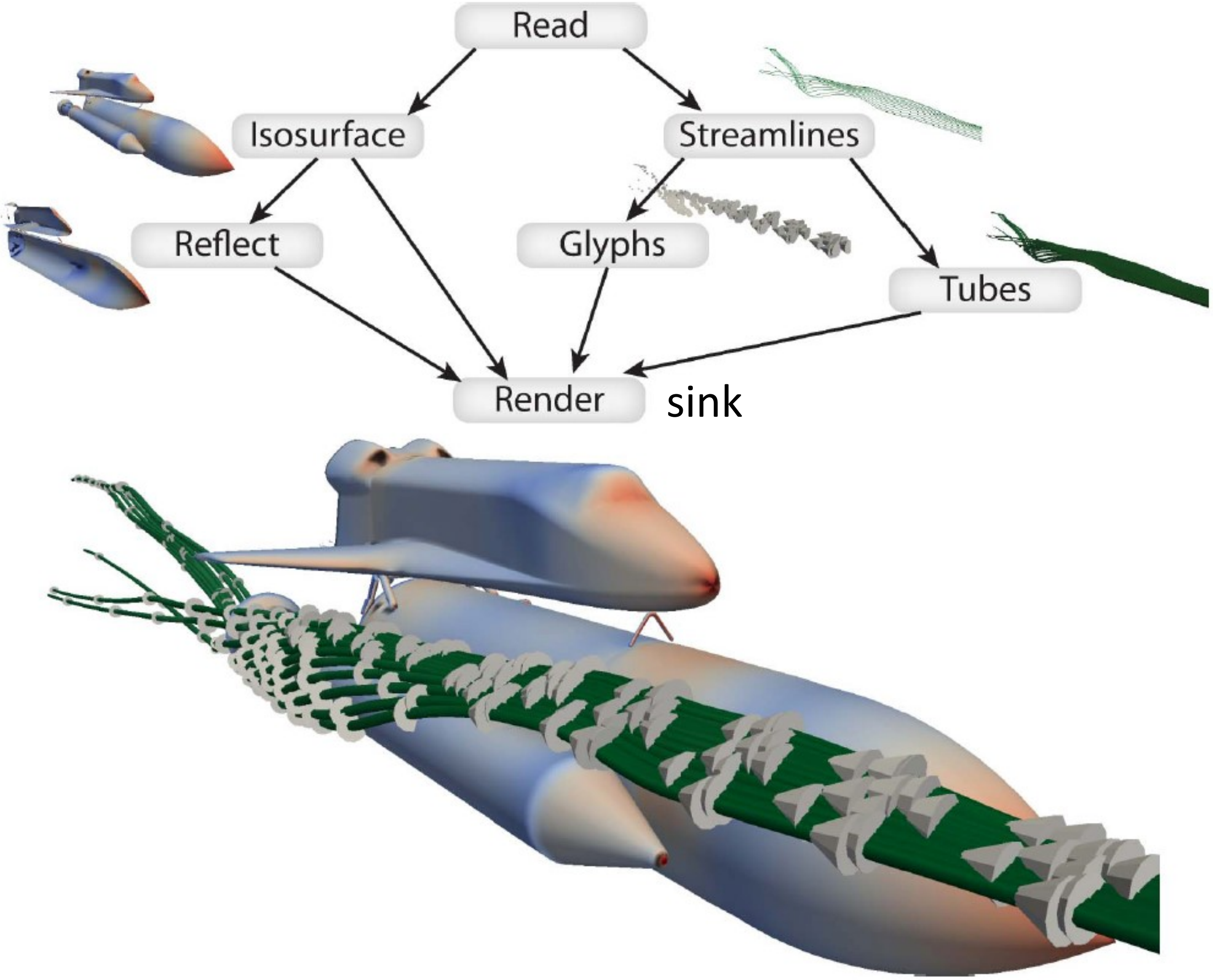
Render

Connections

- Directional attachments from the output port of one module to the input port of another module
- Arc in a directed graph

Visualization Pipeline with Branches

source



sink

Moreland

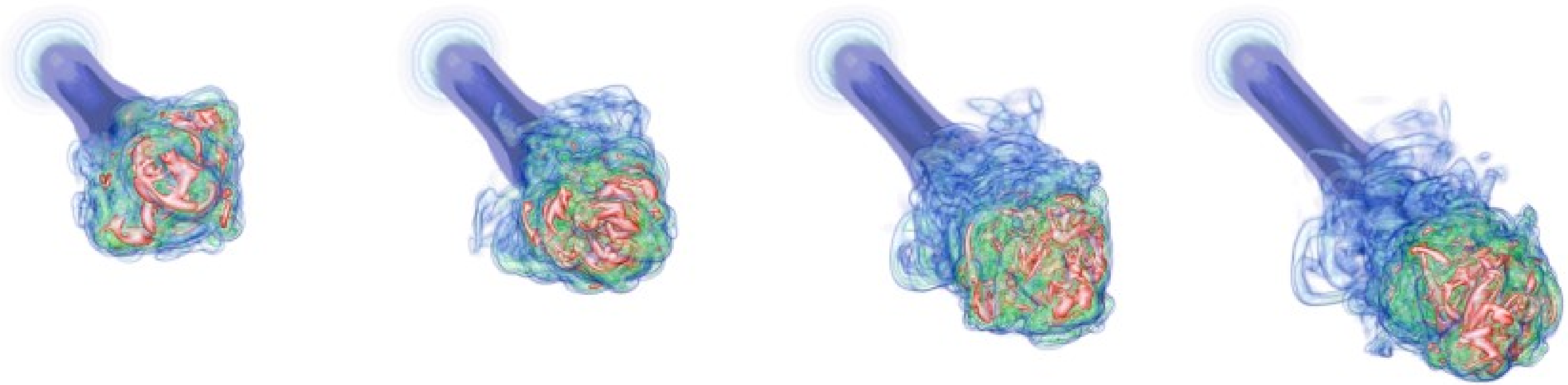
Execution Management

- Determine how and when modules get executed
  - Who starts (drives) execution?
  - Do we need to store intermediate results?
  - Where to control execution?
  - What if the data is too big to fit to a module?

Execution Drivers: Event-driven

- Launch execution as data become available in source module
- Source module pushes the data to down-stream modules and triggers an event to execute them
  - Execution is initiated from source
  - Called *push model*
- ex) time-varying data visualization

Example: Time-varying Volume Rendering

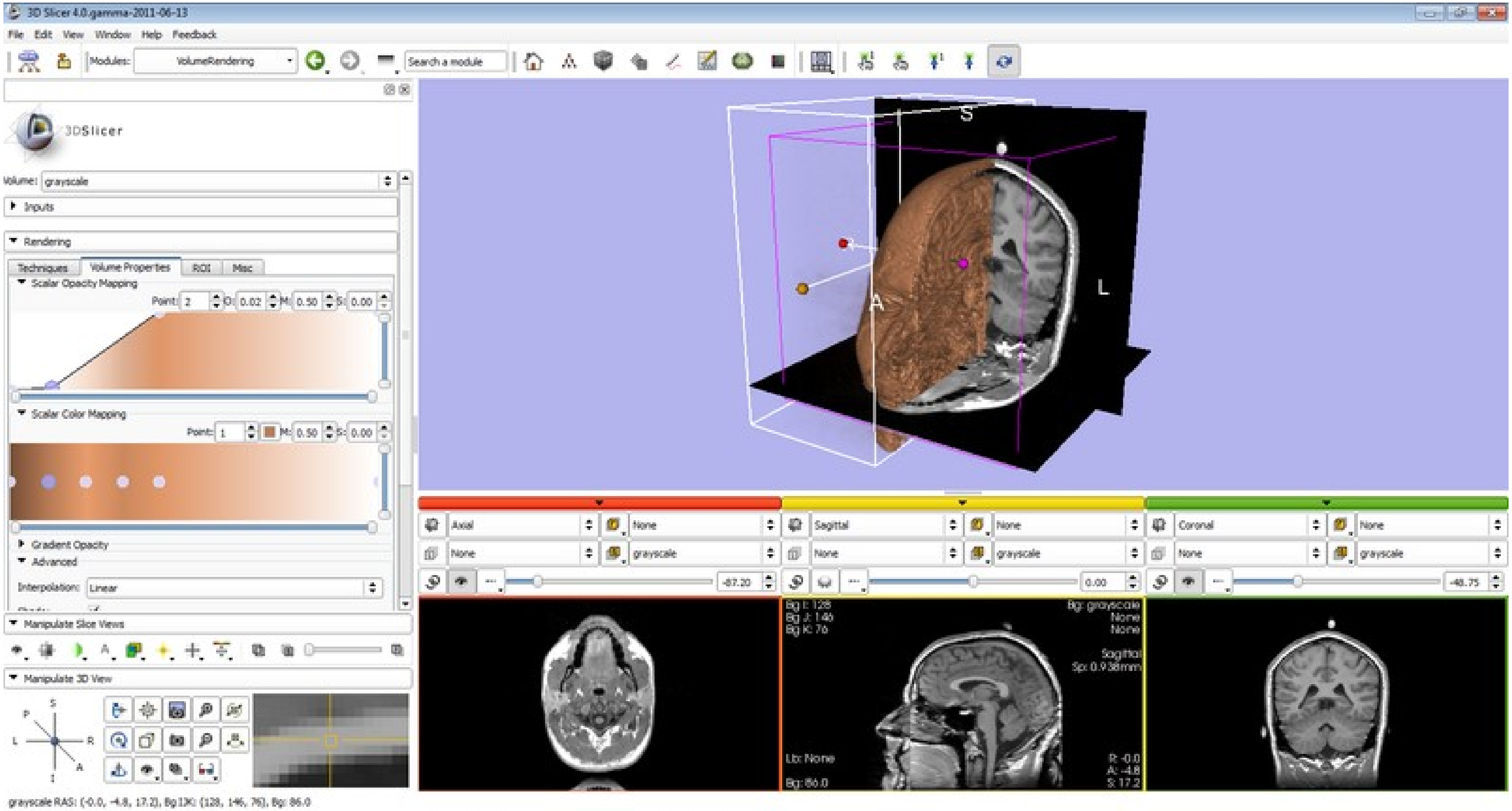


Akiba et al.

Execution Drivers: Demand-driven

- Launch execution in response to requests for data
- Execution is initiated at the bottom of the pipeline in a sink
  - Request is passed to upstream modules to source
  - Once request reaches a source, it produces data and returns execution back to downstream
  - Called *pull model*
- ex) render request to update GUI

Example: Interactive GUI



3D Slicer 4.0.gamma-2011-06-13

Modules: VolumeRendering

Volume: grayscale

Point: 2 O: 0.02 M: 0.50 S: 0.00

Point: 1 M: 0.50 S: 0.00

Interpolation: Linear

Axial None None grayscale -87.20

Sagittal None None grayscale 0.00

Coronal None None grayscale -48.75

Bg I: 128
Bg J: 146
Bg K: 76

Bg: grayscale
None
None

Sagittal
Sp: 0.938mm

Lb: None
Bg: 86.0

R: -0.0
A: -4.8
S: 17.2

grayscale RAS: (-0.0, -4.8, 17.2), Bg IJK: (128, 146, 76), Bg: 86.0

3D slicer

Caching Intermediate Values

- Store intermediate results
- Reduce re-run the entire pipeline
  - For example, if a user change the parameter for a specific module using GUI, modules upstream of that module do not need to be executed if the result is cached
- Speed – memory tradeoff

Control

- Centralized
  - Single unit managing the execution of all modules in the pipeline
  - Control unit has links to all modules
  - Complicated to implement, poor scalability, better cache / load balancing
- Distributed
  - Separate unit for each module
  - Use messages to propagate execution
  - Simple to implement, better scalability

Out-of-core Streaming

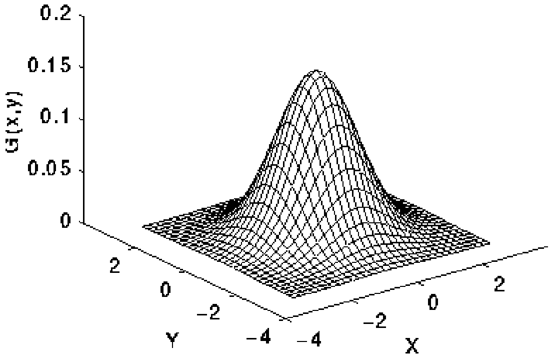
- Out-of-core
  - Data is too large to fit within internal memory
- Streaming
  - Way to perform out-of-core in pipeline
  - Read data in pieces and let it flow through pipeline

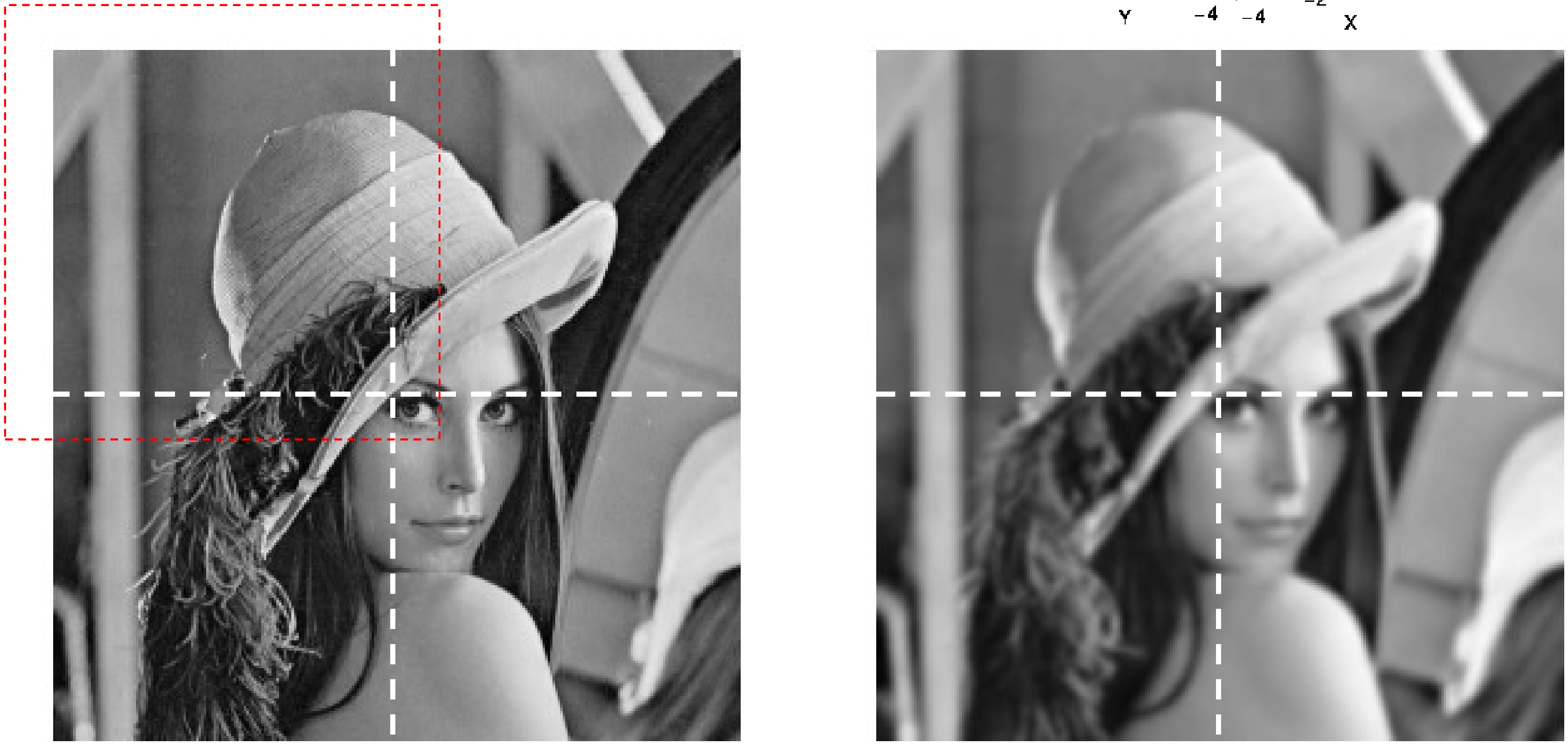
Out-of-core Streaming

- Types of algorithm for streaming
  - Separable (break into pieces)
  - Result invariant (processing order independent)
  - Mappable (should be able to determine which portion of input is required for output)
- Ghost (halo) cells
  - Extra layer of cells at boundary
  - Required for streaming

Example

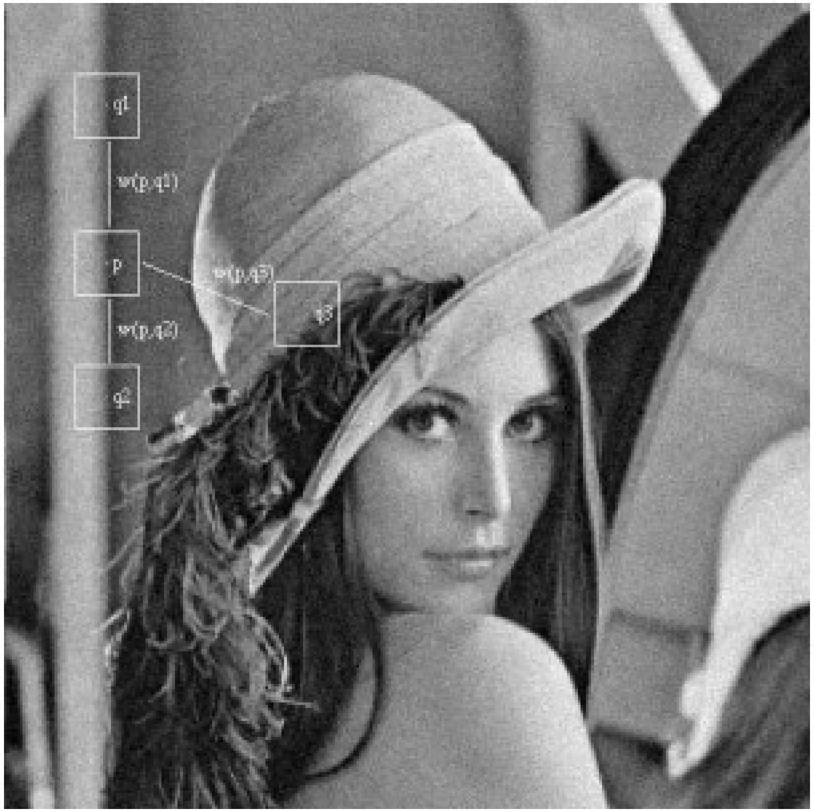
- Gaussian smoothing
  - Separable, result invariant, mappable?





Example

- Non-local Mean
  - Separable, result invariant, mappable?



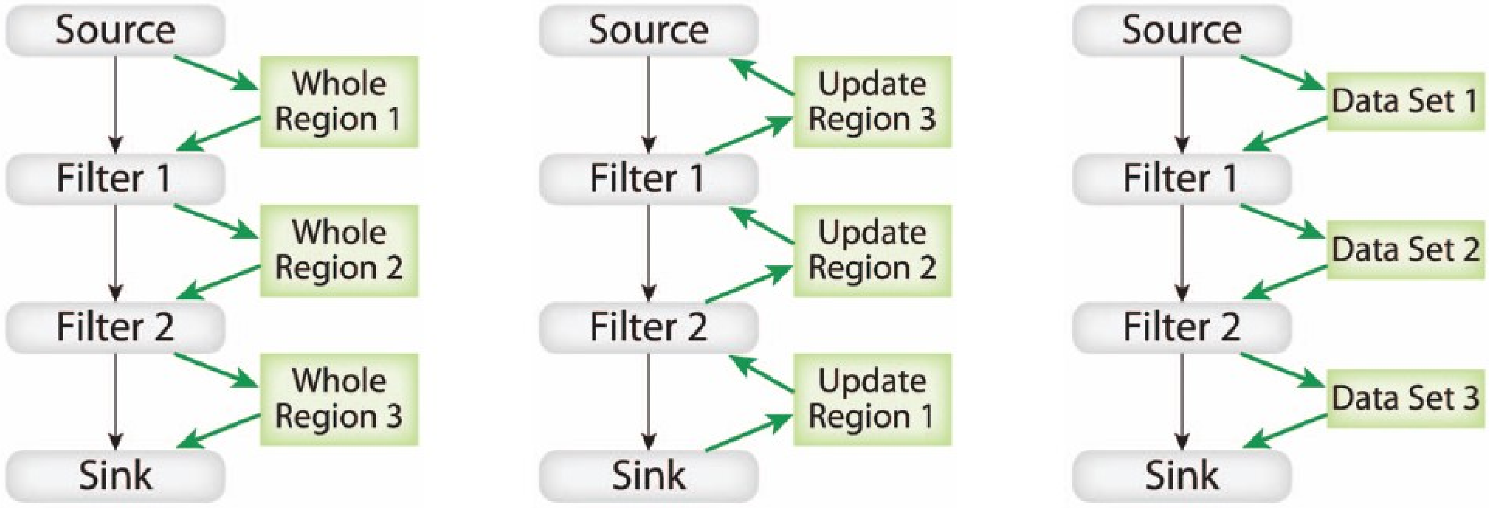
$$NL(v)(i) = \sum_{j \in I} w(i,j)v(j)$$

Metadata

- Modern visualization pipelines employ *metadata* – a brief description of actual data
- Metadata can flow through the pipeline independent from the data
- Regions, time, contracts,...

Regions

- How to split, where to modify
- Extents (index range), pieces (collection of cells), blocks (logical domain decomposition)



(a) Update Information (b) Update Region (c) Update Data

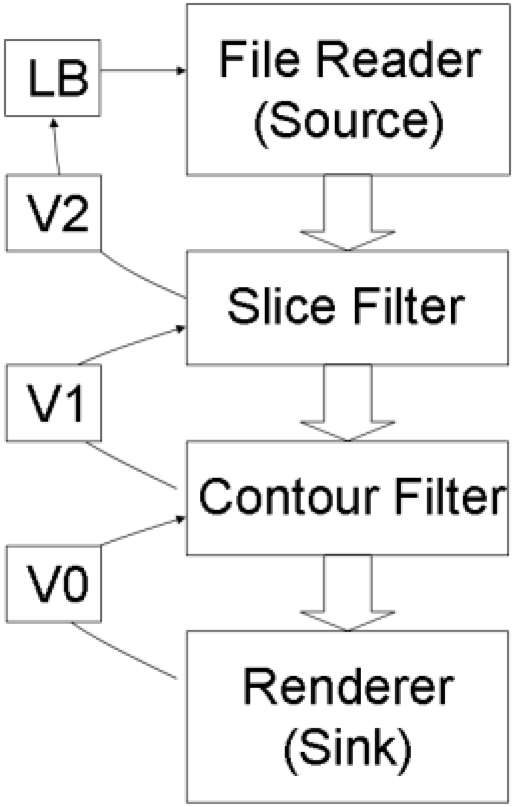
Three pipeline passes for using regional data

Time

- Executing pipeline over a sequence of time steps
- Add time dimension to the region metadata
- Source declares all available time steps
- Each filter can augment time during the update information pass

Contracts

- General data structure for Impact of a filter
  - – Regions, variables, time step, operating restrictions (support of streaming or requiring of ghost cell)
  - – Filters declare their impact by modifying a contract object



v0, v1… : contract versions
LB: load balancer

Example: Prioritized streaming

- Change the order of processing based on priority
- Priority metric
  - Close proximity to the viewer in 3D
  - Least likely to be culled
  - Regions with scalar values in an interesting range
  - Regions with more variability

Example: Query-driven visualization

- Analyze a large data set by identifying "interesting" data that matches some specified criteria
- Need to quickly load small selections of data with arbitrary specification
- Required technology
  - File indexing
    - The pipeline source must be able to identify where the pertinent data is located without reading the entire file (e.g., tree-based)
  - Query language
    - E.g., Boolean expression (all regions where (temperature > 1000K) AND (70kPa < pressure < 90kPa)
  - Pipeline metadata mechanism

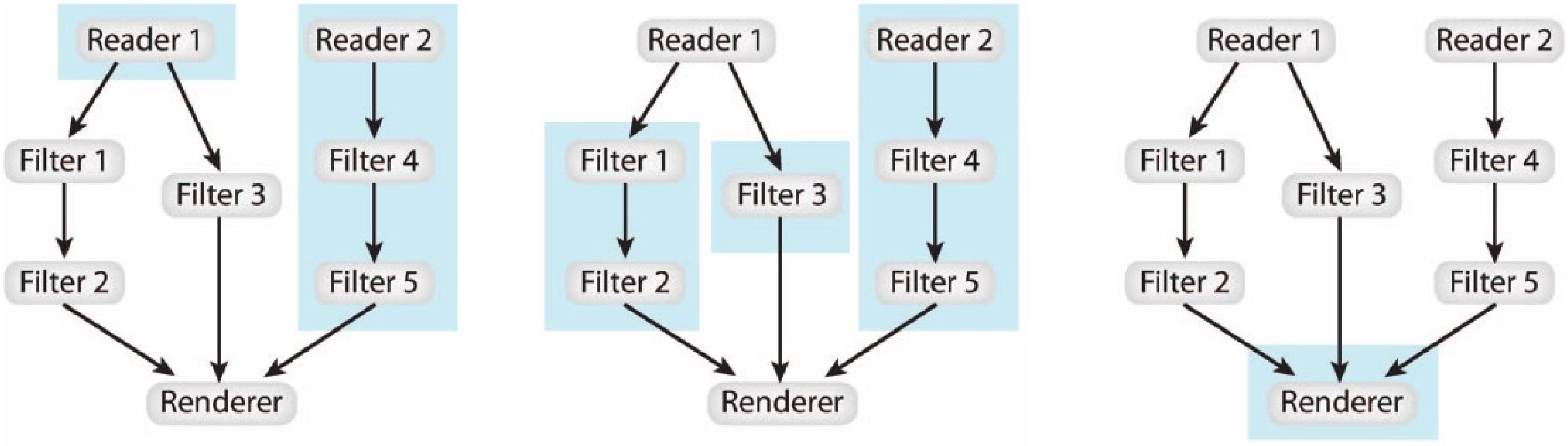
Parallel Execution

- Execute different modules in the pipeline concurrently

- Types of parallel execution
  - Task parallelism
  - Pipeline parallelism
  - Data parallelism

Task Parallelism

- Identify independent portions of the pipeline
  - – Use task dependency
- Load balancing is an issue



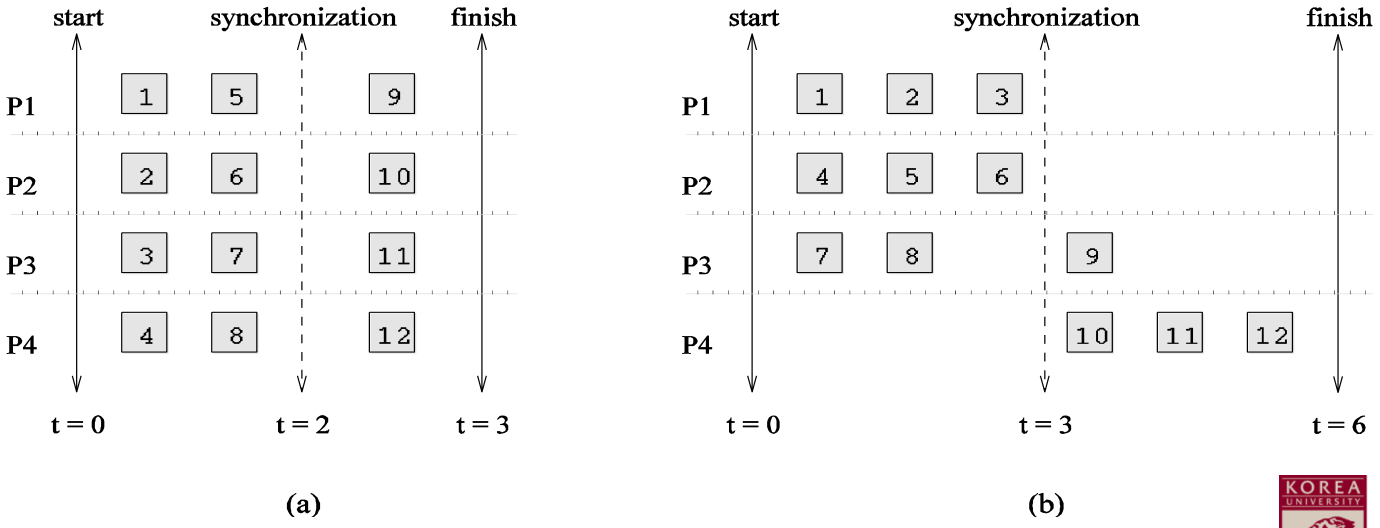
$T = t_0$ $T = t_1$ $T = t_2$

Moreland

Mapping Technique for Load Balancing

- Tasks must be mapped to processes
  - Minimizing overhead is the goal
- Overhead
  - Communication
  - Idling

9-12 can start after 1-8 are done



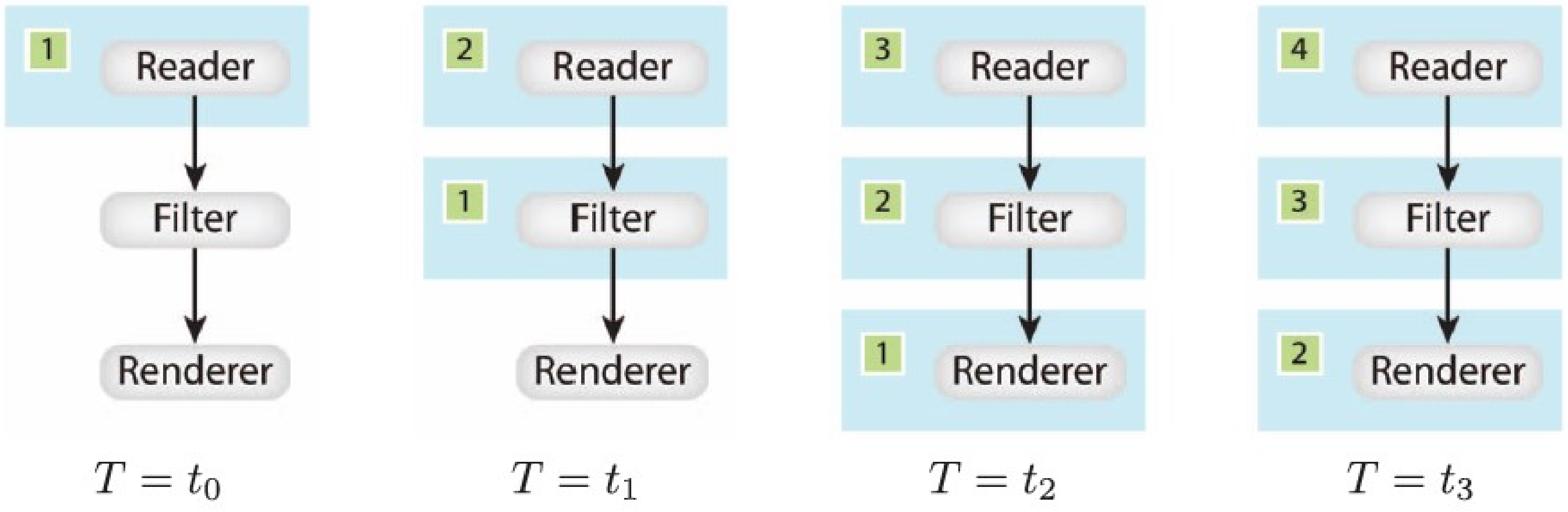
Same load balance but (b) has more idling

Mapping Techniques

- Static Mapping
  - Tasks are mapped to processes a-priori
  - Need a good estimate of the size of each task
  - The mapping problem can be computationally difficult for non-uniform tasks
- Dynamic Mapping
  - Tasks are mapped to processes at runtime
  - Tasks are generated at runtime, or that their sizes are not known

Pipeline Parallelism

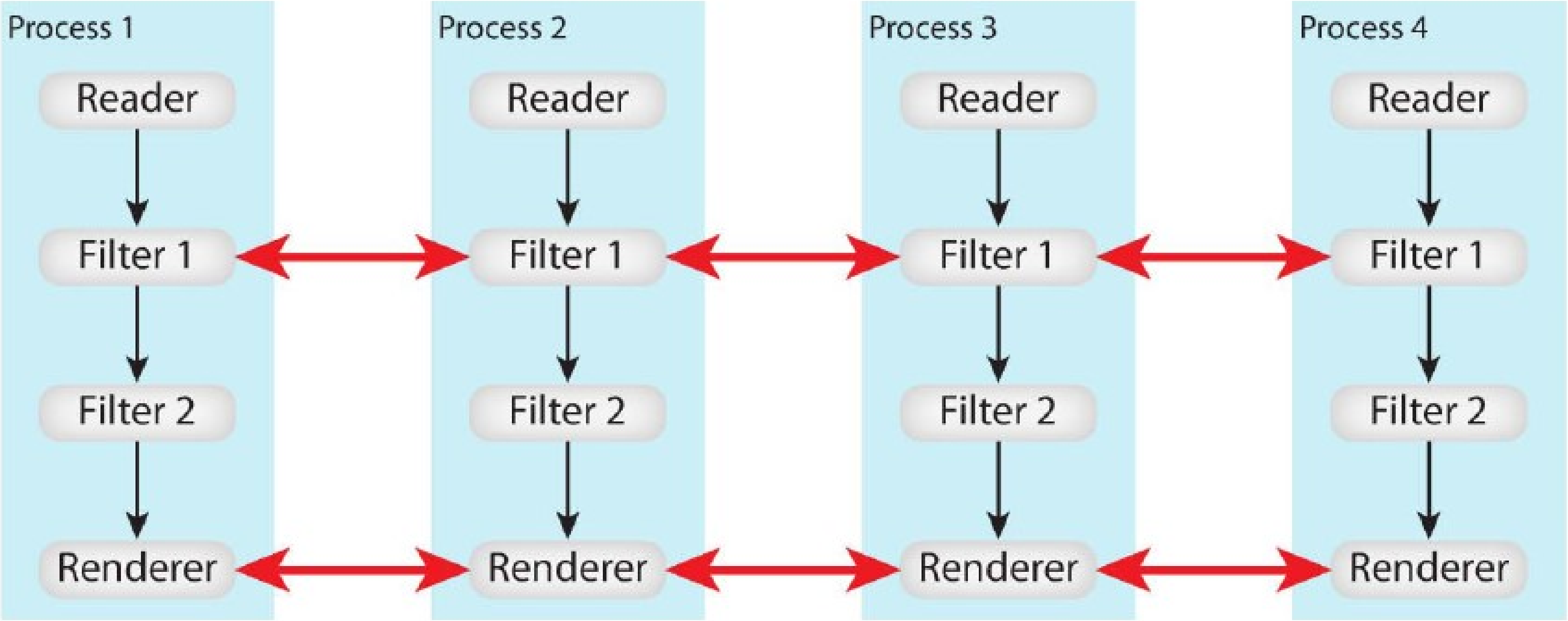
- Streaming data in pieces, execute different modules of the pipeline concurrently



Moreland

Data Parallelism

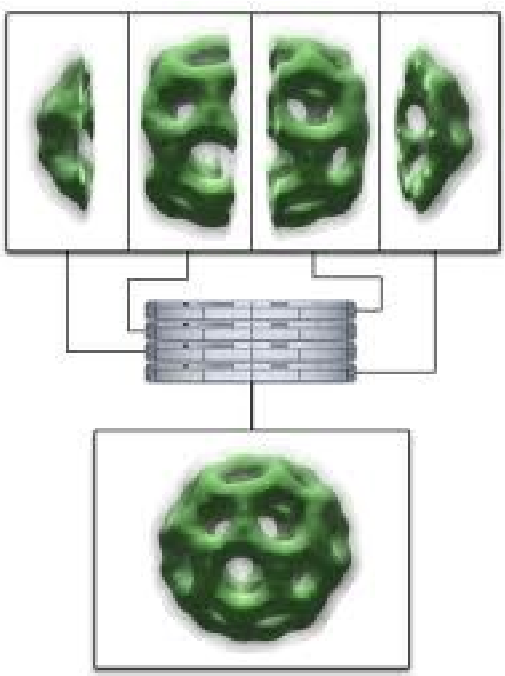
- Partition the input data, replicate pipeline for each piece



Moreland

Sort-last Rendering

- Fits well to data parallelism
- Each piece is rendered independently
- Merge results at the end

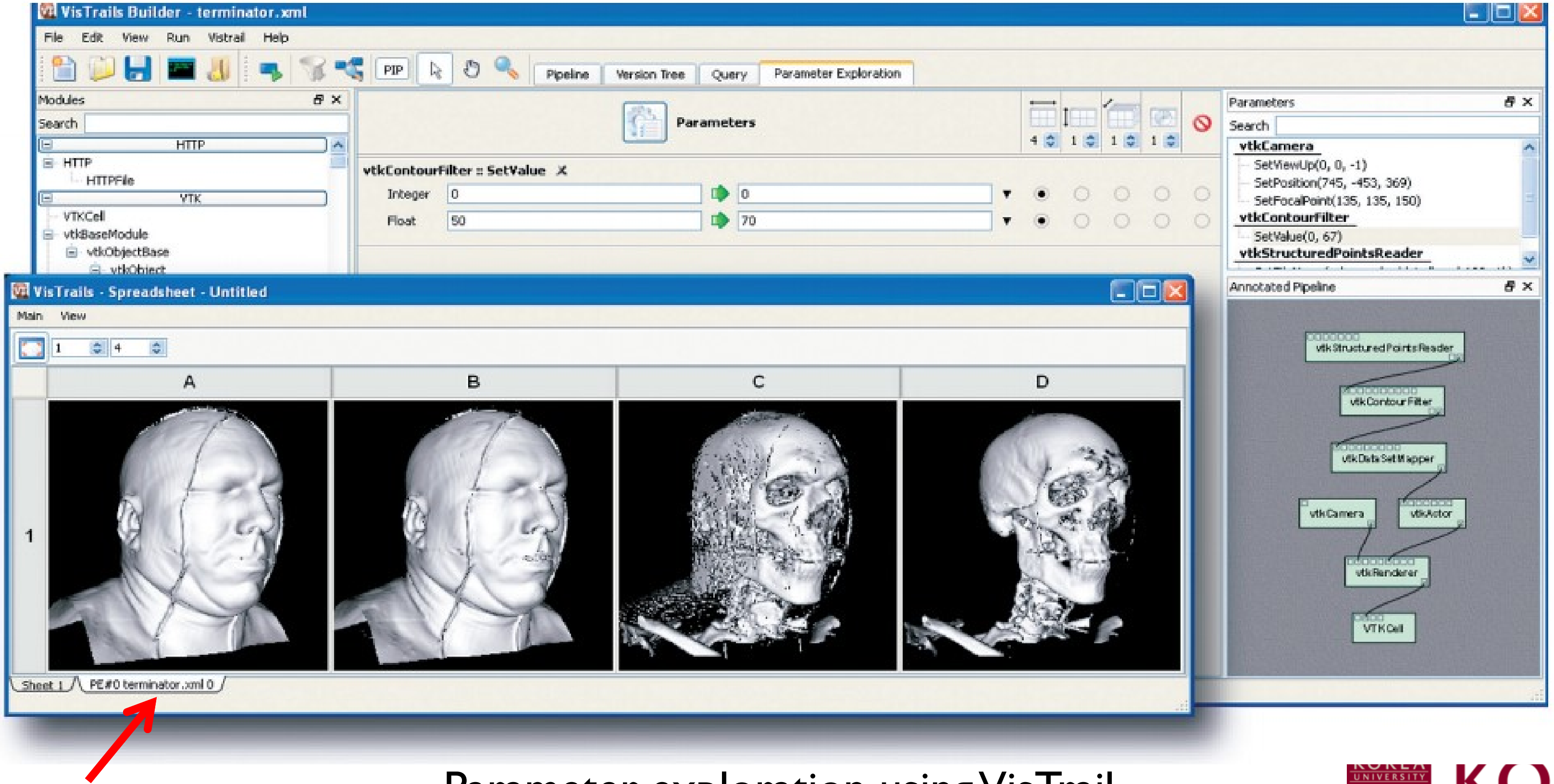


Sort: depth test

equalizergraphics.com

Provenance

- Model exploration as transformations to the visualization pipeline



vtkContourFilter :: SetValue

Integer 0 → 0 ▾ ● ○ ○ ○ ○

Float 50 → 70 ▾ ● ○ ○ ○ ○

vtkCamera
- SetViewUp(0, 0, -1)
- SetPosition(745, -453, 369)
- SetFocalPoint(135, 135, 150)

vtkContourFilter
- SetValue(0, 67)

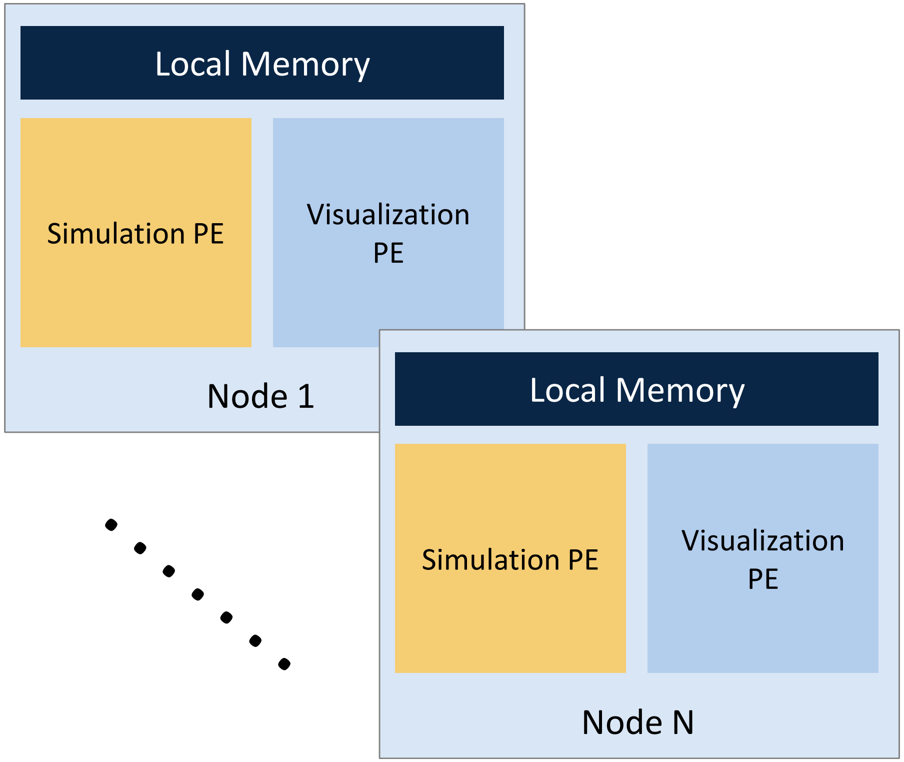
vtkStructuredPointsReader

Spreadsheet

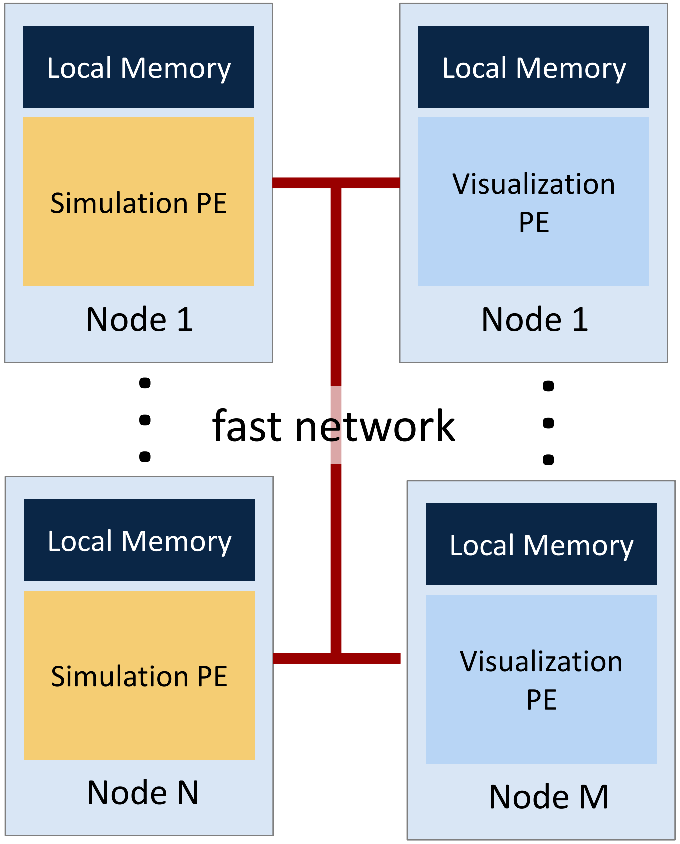
Parameter exploration using VisTrail

In Situ Visualization

- Simulation and visualization run concurrently



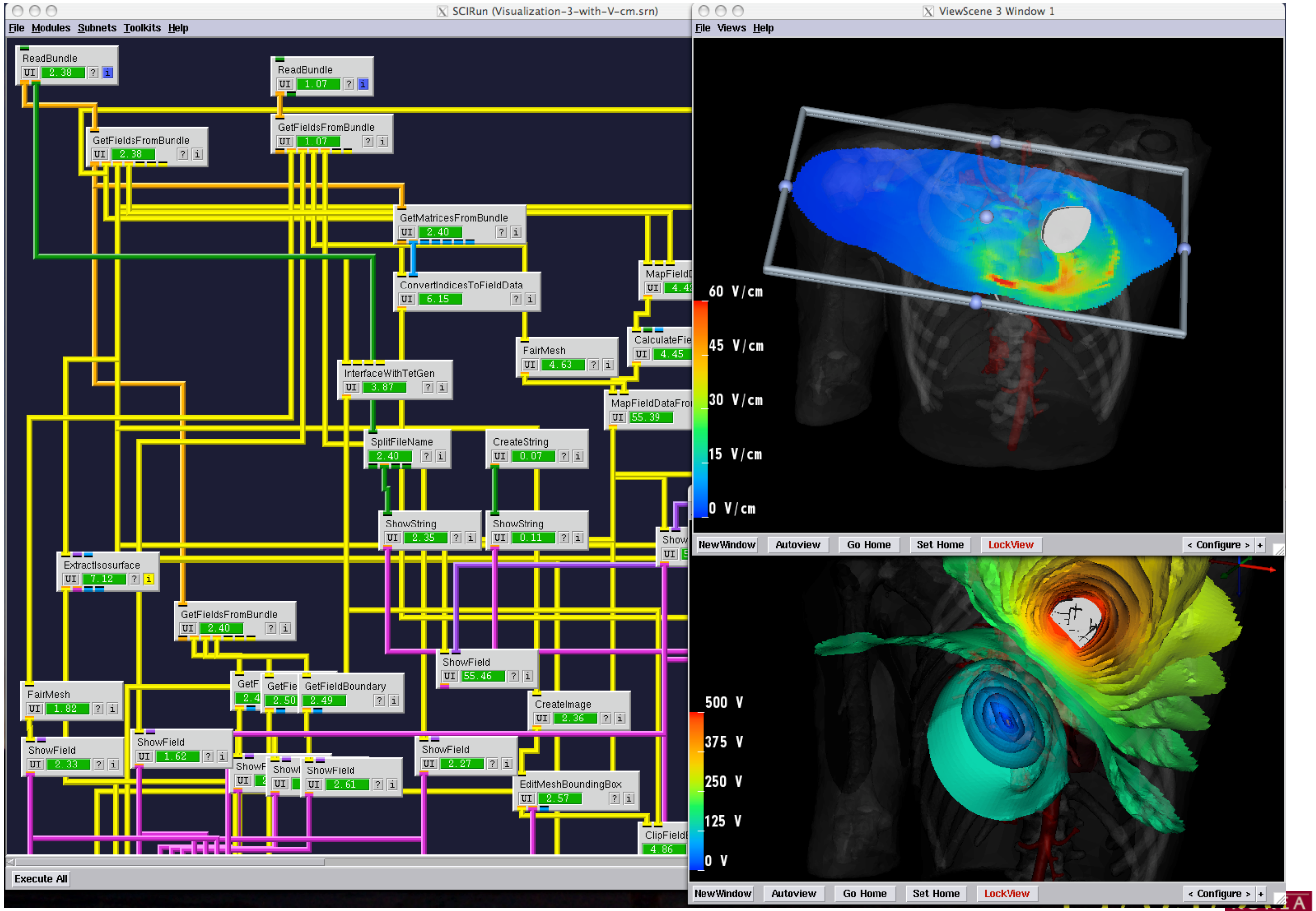
Tightly Coupled N nodes



N simulation nodes,
M visualization nodes

Dataflow Visualization Systems

- SCIRun

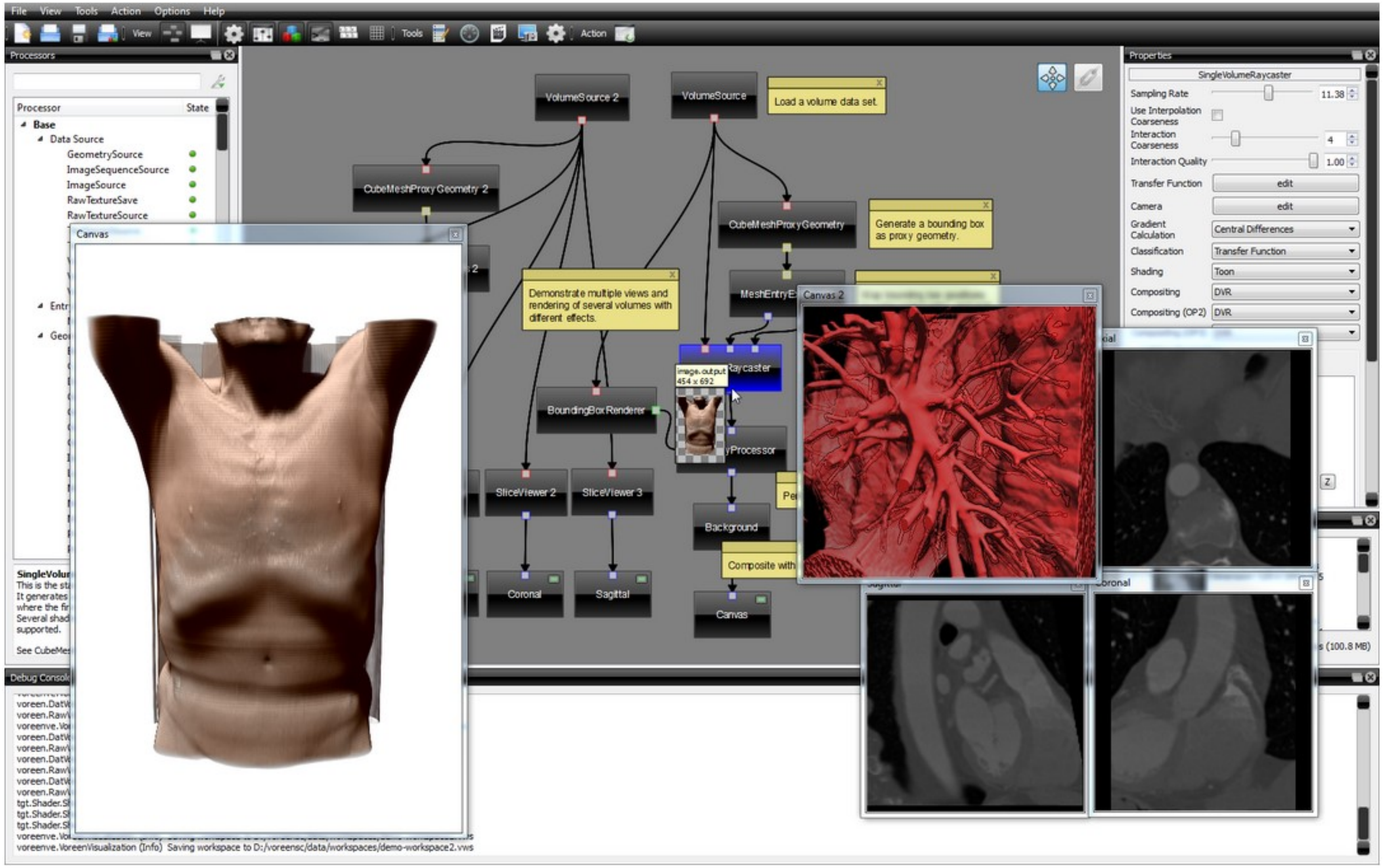


Modeling, simulation, visualization in one package

www.scirun.org

Dataflow Visualization Systems

- Voreen



Load a volume data set.

Generate a bounding box as proxy geometry.

Demonstrate multiple views and rendering of several volumes with different effects.

SingleVolumeRaycaster

- Sampling Rate: 11.38
- Use Interpolation Coarseness: □
- Interaction Coarseness: 4
- Interaction Quality: 1.00
- Transfer Function: edit
- Camera: edit
- Gradient Calculation: Central Differences
- Classification: Transfer Function
- Shading: Toon
- Compositing: DVR
- Compositing (OP2): DVR

SingleVolu
This is the st
It generates
where the fi
Several shad
supported.

See CubeMes

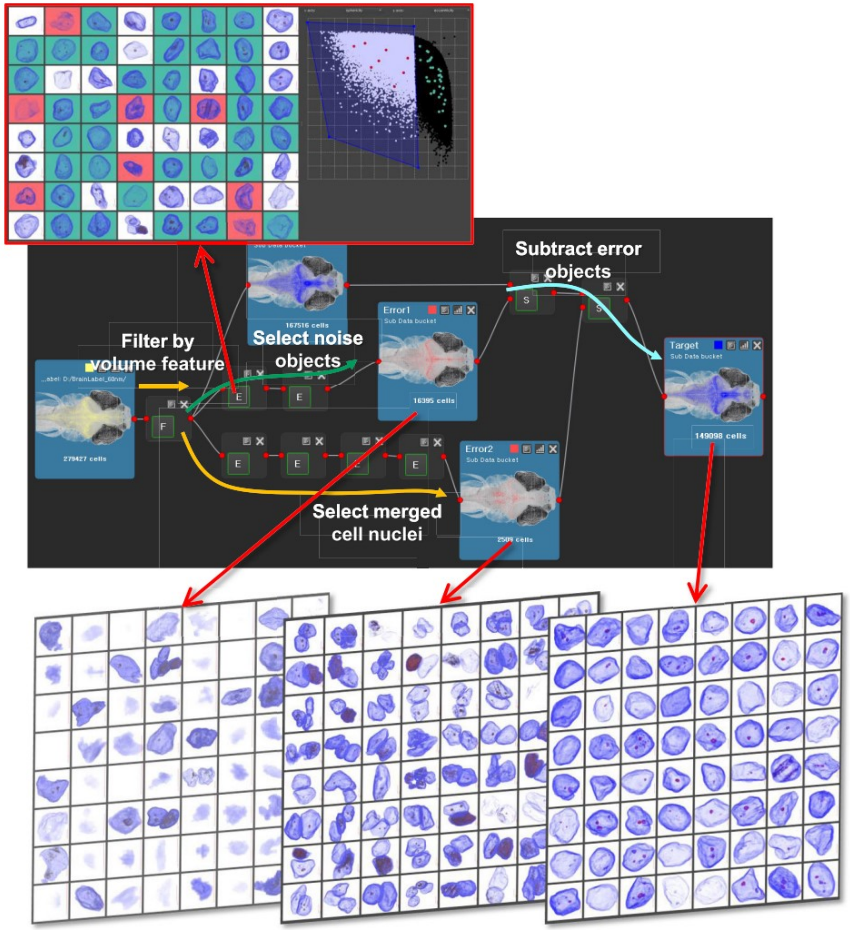
```
voreen.DatV
voreen.RawV
voreenve.Vo
voreen.DatV
voreen.RawV
voreen.DatV
voreen.RawV
voreen.DatV
voreen.RawV
tgt.Shader.S
tgt.Shader.S
tgt.Shader.S
voreenve.Vo
voreenve.VoreenVisualization (Info)  Saving workspace to D:/voreensc/data/workspaces/demo-workspace2.vws
```

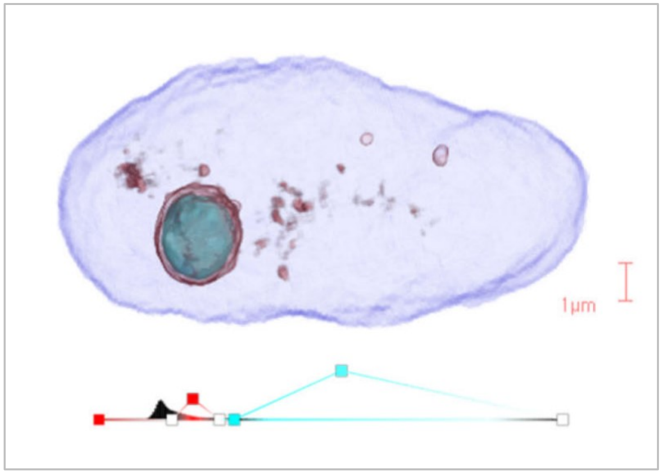
Volume Rendering Engine

www.voreen.org

Dataflow Visualization Systems

- ZeVis: brain cell morphology analysis





Visualization Pipeline Alternative

- Function field model
  - f(x) = y, x : location, y: value
  - Continuous field, derived fields
- MapReduce
  - Simple programming & execution model for distributed processing
- Fine-grained data parallelism
  - Many concurrent threads, multi-CPU or GPU
- Domain-specific languages

Questions?