LA-UR-16-27388
Approved for public release; distribution is unlimited.

Title: Visualization and Data Analysis for High-Performance Computing

Author(s): Sewell, Christopher Meyer

Intended for: Guest Lecture for a class at UT-El Paso

Issued: 2016-09-27

Visualization and Data Analysis for High-Performance Computing

Christopher Sewell

October 2016

Outline

- Trends in HPC
- Scientific Visualization
  - OpenGL
  - Ray Tracing and Volume Rendering
  - VTK
  - ParaView
- Data Science at Scale
  - In-Situ Visualization
  - Image Databases
  - Distributed Memory Parallelism
  - Shared Memory Parallelism
  - VTK-m
  - “Big Data”
  - Analysis Example

Slide Credits

- Trends in HPC
  - Francisco Ortega, Structure Parallel Programming Workshop: http://franciscoraulortega.com/ws/lecture-1-overview.pptx
  - Ken Moreland, UltraViz Workshop 2015: http://m.vtk.org/images/a/a5/VTKmUltraVis2015.pptx
  - Exascale Initiative Steering Committee, SOS 14 2010: http://www.csm.ornl.gov/workshops/SOS14/documents/dosanjh_pres.pdf
- Scientific Visualization
  - OpenGL
    - Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013: https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx
  - Ray Tracing and Volume Rendering
    - Ed Angel, Interactive Computer Graphics 5E, 2009: http://www.cs.utsa.edu/~jpq/Site/teaching/cg-s11/raytracing.ppt
    - Joe Michael Kniss, University of Utah: http://www.cs.utah.edu/~jmk/papers/volumeRendering-g.ppt
  - VTK
    - Robert Putnam, Class Presentation at Boston University, 2010: www.bu.edu/tech/files/2010/10/VTK-Fall-2010.ppt
    - Scott Schaefer, Class Presentation at Texas A&M: http://faculty.cs.tamu.edu/schaefer/teaching/645_Fall2015/lectures/volumetric.ppt
  - ParaView
    - Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015: http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx
- Data Science at Scale
  - In-Situ Visualization
    - Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015: http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx
  - Image Databases
    - Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015: http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx
    - John Patchett, Presentation at Kaiserlautern University, 2014: https://datascience.lanl.gov/data/KaiserslauternInSituPatchett.pptx
  - Distributed Memory Parallelism
    - Pavan Balajii and Torsten Hoefler, Presentation at PPoPP 2013: https://htor.inf.ethz.ch/teaching/mpi_tutorials/ppopp13/2013-02-24-ppopp-mpi-basic.pdf
    - Elliott Slaughter, Presentation at Oak Ridge National Lab, 2016 :https://www.olcf.ornl.gov/wp-content/uploads/2016/01/Legion_Elliott_Slaughter.pptx
  - Shared Memory Parallelism
    - Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016: https://datascience.lanl.gov/data/Presentation2014Oregon.pdf
    - Lars Arge, Presentation at the University of Aarhus, 2013: http://www.slideshare.net/ktoshik/io-efficient-algorithms-and-data-structures-34-lecture-by-lars-arge
  - VTK-m
    - Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016: http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx
  - Big Data
    - David Wheeler, Class Presentation at George Mason University: www.dwheeler.com/essays/hadoop-spark.ppt
    - Matei Zaharia, et. al., Tutorial at spark.apache.org: https://spark.apache.org/talks/overview.pptx
  - Analysis Example
    - Chris Sewell, Presentation at Supercomputing 2015: http://datascience.dsscale.org/wp-content/uploads/sites/3/2016/06/Large-ScaleCompute-IntensiveAnalysisViaACombinedIn-situAndCo-schedulingWorkflowApproach.pdf

Trends in HPC

Power Wall, Cost of Data Movement, On-node Parallelism

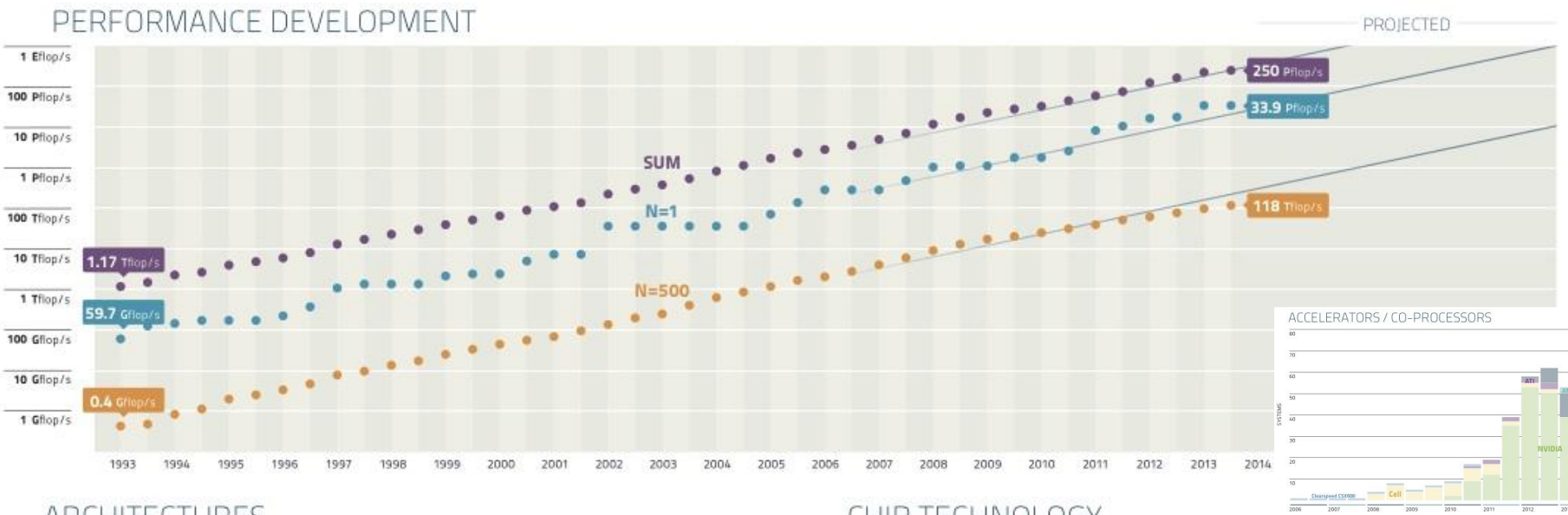
Top 10 Supercomputers (November 2013)

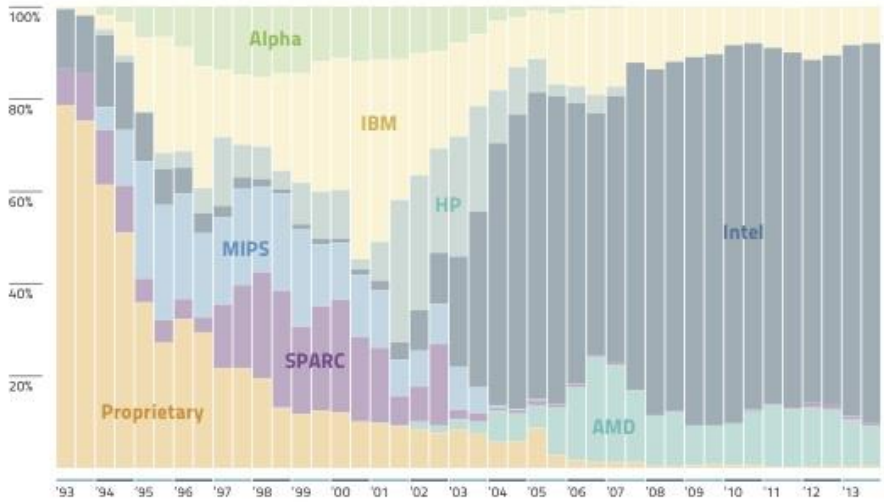
Different architectures

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
5	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945
6	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc.	115,984	6,271.0	7,788.9	2,325
7	Texas Advanced Computing Center/Univ. of Texas United States	**Stampede** - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P Dell	462,462	5,168.1	8,520.1	4,510
8	Forschungszentrum Juelich (FZJ) Germany	**JUQUEEN** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	458,752	5,008.9	5,872.0	2,301
9	DOE/NNSA/LLNL United States	**Vulcan** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	393,216	4,293.3	5,033.2	1,972
10	Leibniz Rechenzentrum Germany	**SuperMUC** - iDataPlex DX360M4, Xeon E5-2680 8C 2.70GHz, Infiniband FDR IBM	147,456	2,897.0	3,185.1	3,423

Francisco Ortega, Structure Parallel Programming Workshop: http://franciscoraulortega.com/ws/lecture-1-overview.pptx

Top 500 – Performance (November 2013)





Francisco Ortega, Structure Parallel Programming Workshop: http://franciscoraulortega.com/ws/lecture-1-overview.pptx

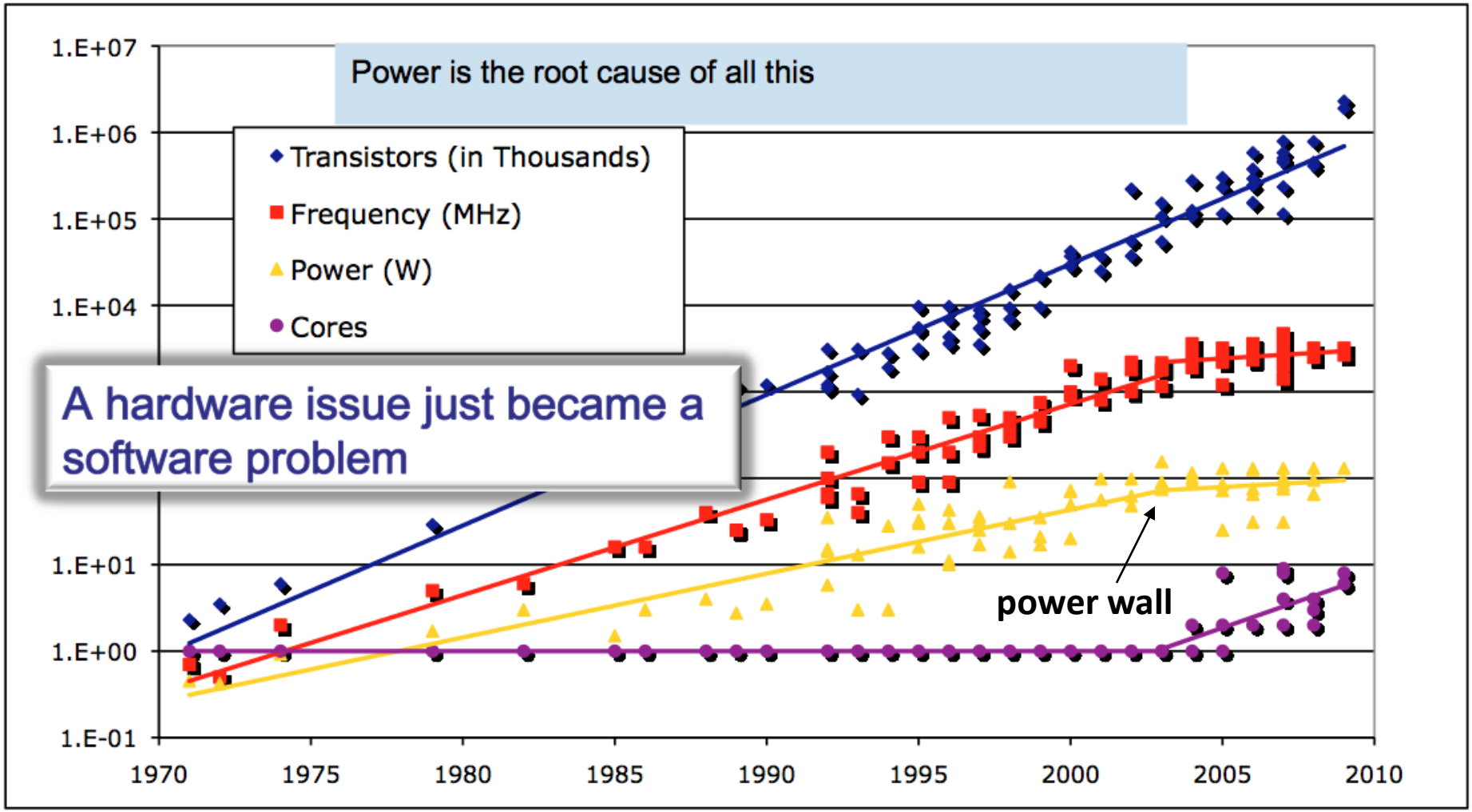
What has happened in the last several years?

- Processing chip manufacturers increased processor performance by increasing CPU clock frequency
  - Riding Moore's law
- Until the chips got too hot!
  - Greater clock frequency $\Rightarrow$ greater electrical power
  - Pentium 4 heat sink ❍ Frying an egg on a Pentium 4

- Add multiple cores to add performance
  - Keep clock frequency same or reduced
  - Keep lid on power requirements

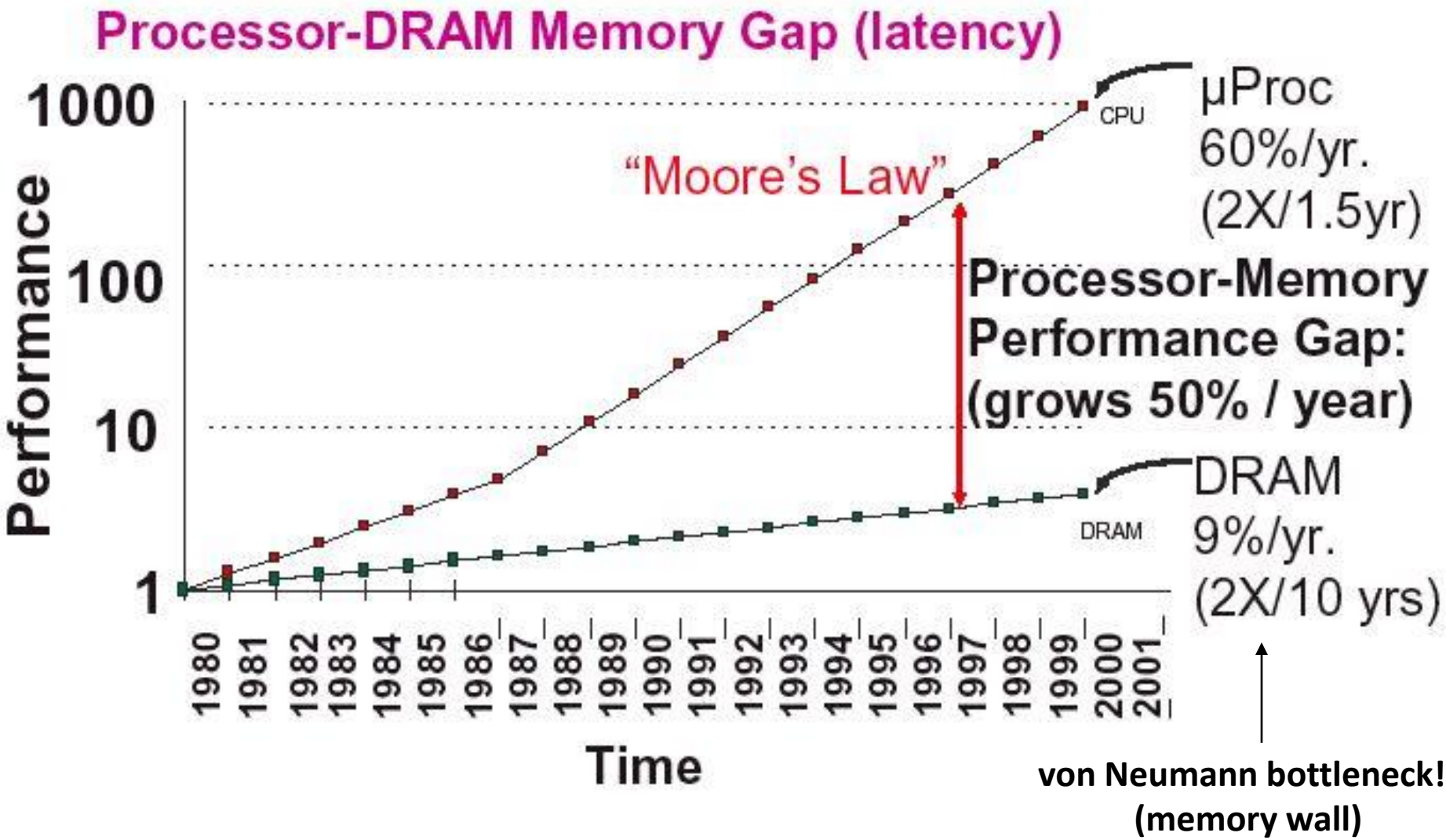
Francisco Ortega, Structure Parallel Programming Workshop:
http://franciscoraulortega.com/ws/lecture-1-overview.pptx

What's Driving Parallel Computing Architecture?



Francisco Ortega, Structure Parallel Programming Workshop: http://franciscoraulortega.com/ws/lecture-1-overview.pptx

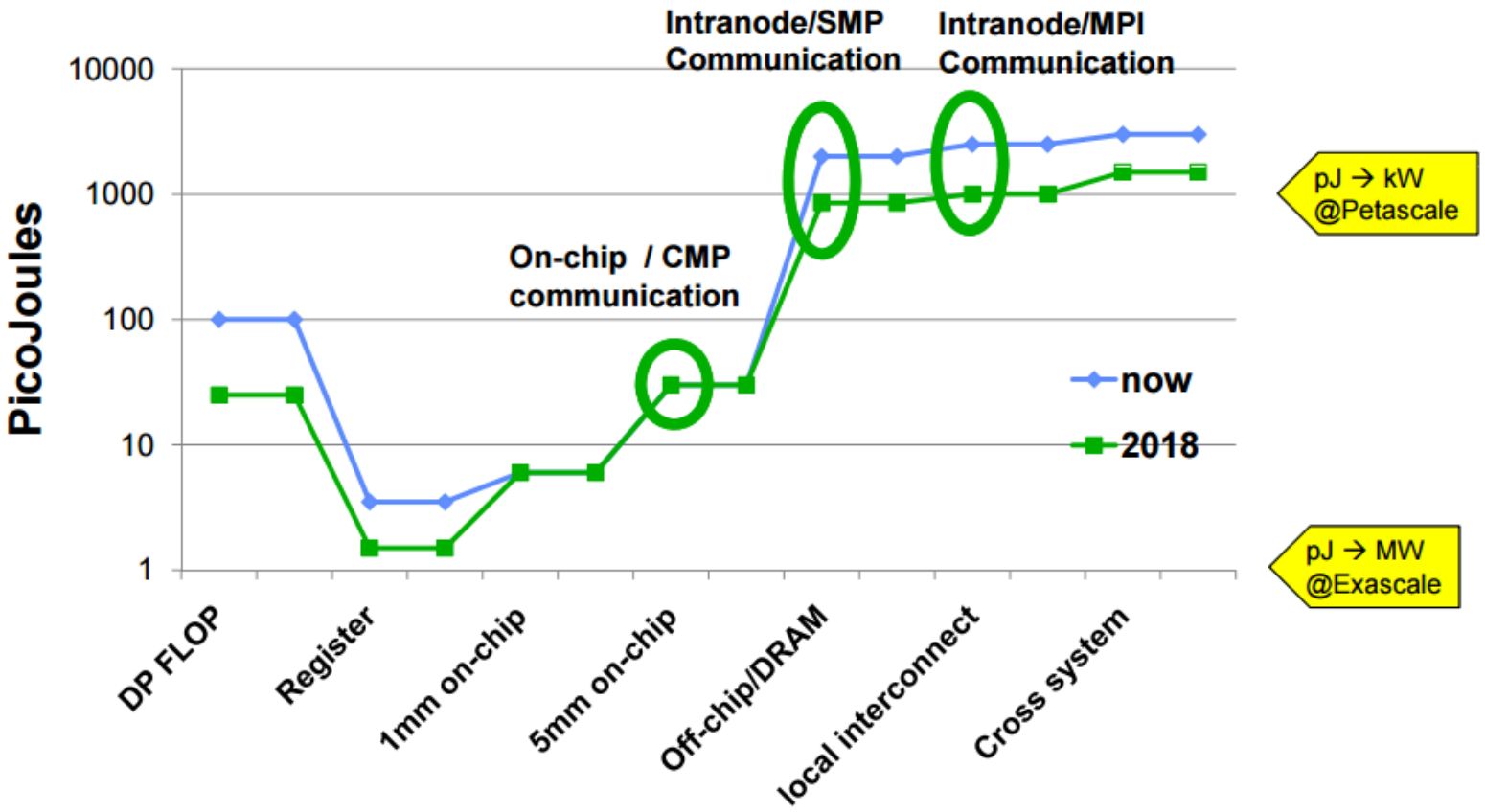
What's Driving Parallel Computing Architecture?



Francisco Ortega, Structure Parallel Programming Workshop:
http://franciscoraulortega.com/ws/lecture-1-overview.pptx

U.S. DEPARTMENT OF ENERGY

Investments in architecture R&D and application locality are critical



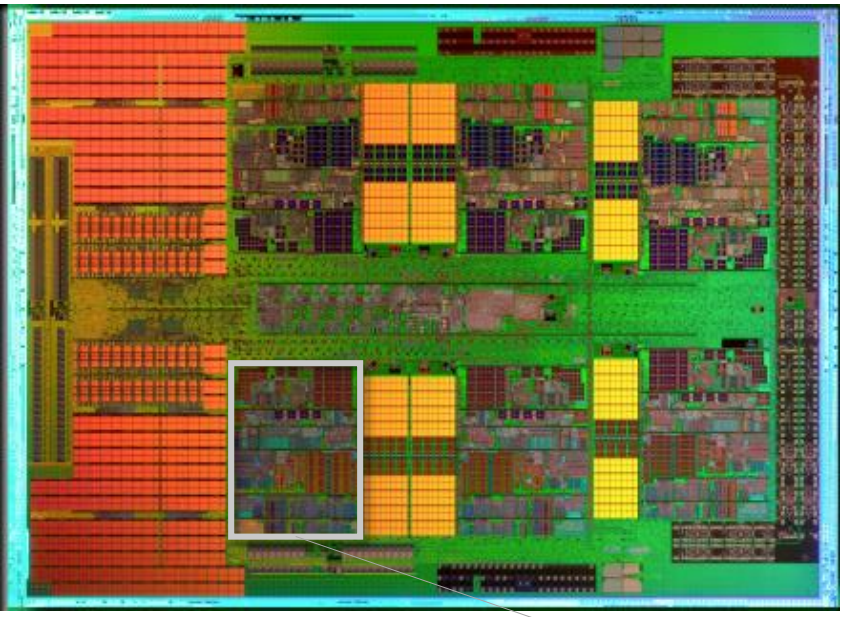
"The Energy and Power Challenge is the most pervasive … and has its roots in the inability of the [study] group to project any combination of currently mature technologies that will deliver sufficiently powerful systems in any class at the desired levels."
DARPA IPTO exascale technology challenge report

Exascale Initiative Steering Committee, SOS 14 2010:
http://www.csm.ornl.gov/workshops/SOS14/documents/dosanjh_pres.pdf

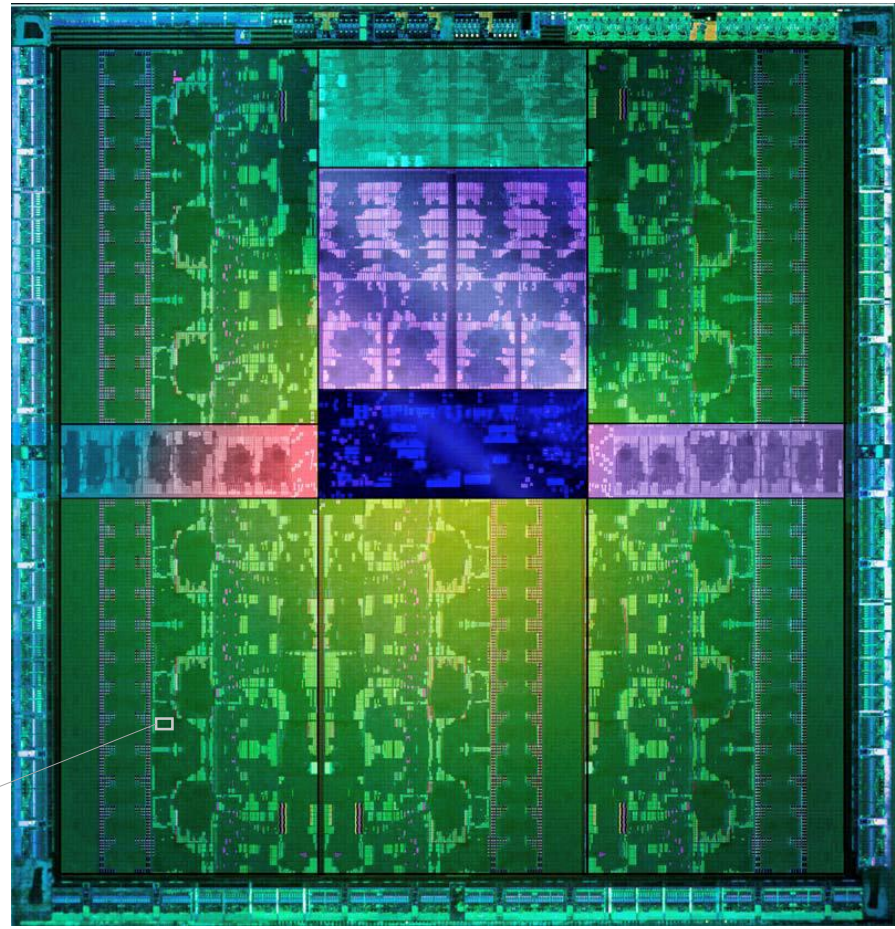
Exascale Design Point

Systems	2012 BG/Q Computer	2020-2024	Difference Today & 2019
System peak	**20 Pflop/s**	**1 Eflop/s**	**O(100)**
Power	**8.6 MW**	**~20 MW**	
System memory	1.6 PB (16*96*1024)	32 - 64 PB	O(10)
Node performance	205 GF/s (16*1.6GHz*8)	1.2 or 15TF/s	O(10) – O(100)
Node memory BW	42.6 GB/s	2 - 4TB/s	O(1000)
Node concurrency	64 Threads	O(1k) or 10k	O(100) – O(1000)
Total Node Interconnect BW	20 GB/s	200-400GB/s	O(10)
System size (nodes)	98,304 (96*1024)	O(100,000) or O(1M)	
Total concurrency	5.97 M	O(billion)	O(1,000)
MTTI	4 days	O(<1 day)	- O(10)

From Marc Snir, Argonne National Laboratory: http://www.slideshare.net/marcsnir/resilience-at-exascale



1mm



AMD x86

Full x86 Core
+ Associated Cache
8 cores per die
MPI-Only feasible

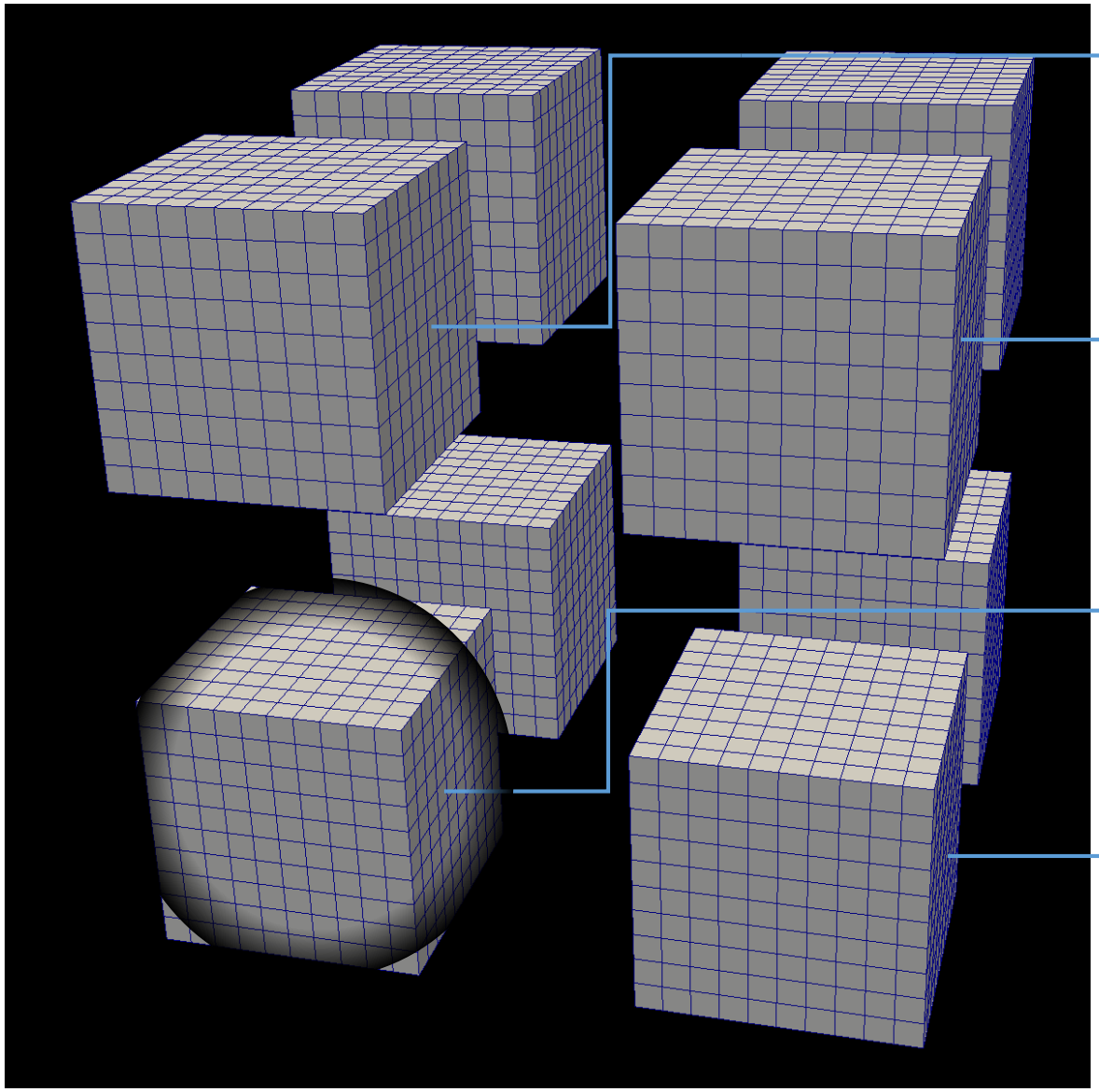
1 x86 core

1 Kepler core

NVIDIA GPU

2,880 cores collected in 15 SMX
Shared PC, Cache, Mem Fetches
Reduced control logic
MPI-Only not feasible

Ken Moreland, UltraViz Workshop 2015:
http://m.vtk.org/images/a/a5/VTKmUltraVis2015.pptx



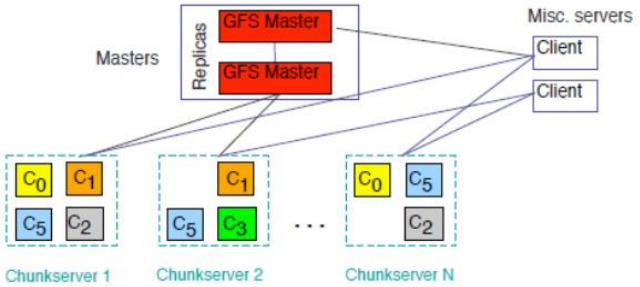
Inter-Node Parallelism

Intra-Node Parallelism

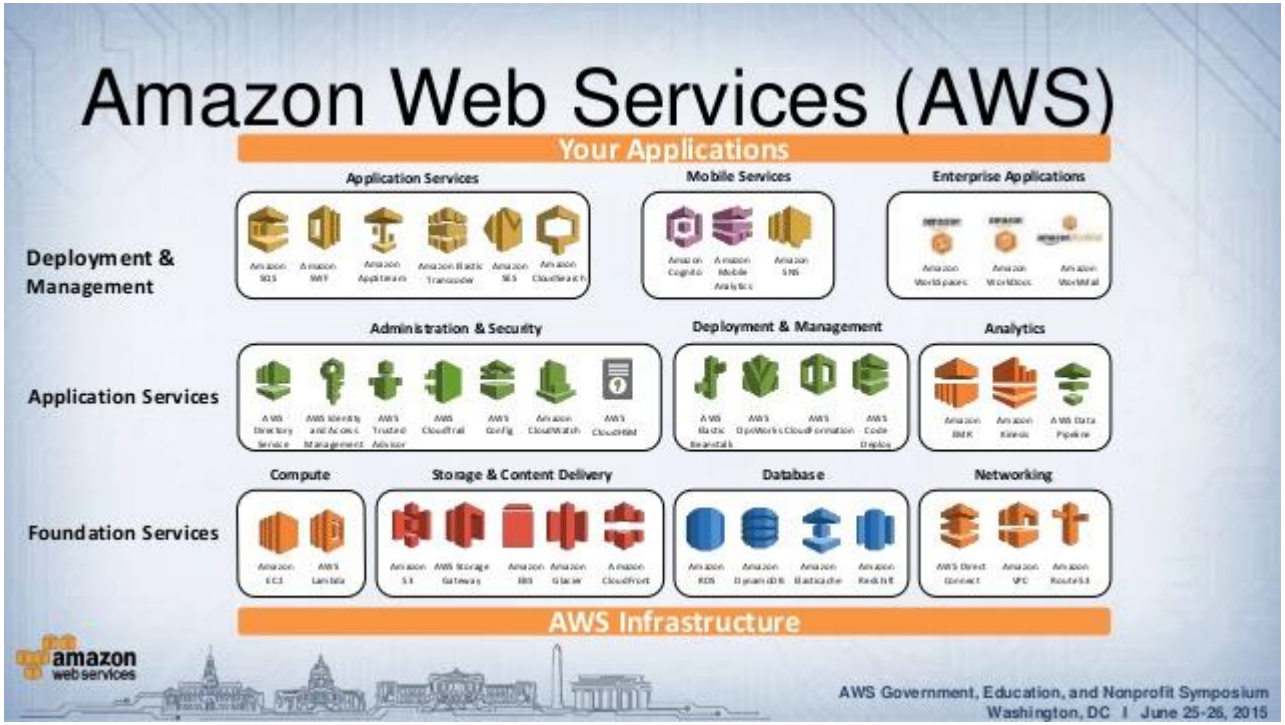
Ken Moreland, UltraViz Workshop 2015:
http://m.vtk.org/images/a/a5/VTKmUltraVis2015.pptx

HPC in the Cloud?

GFS (Google File System) Design



- Master manages metadata
- Data transfers happen directly between clients/chunk servers
- Files broken into chunks (typically 64 MB)



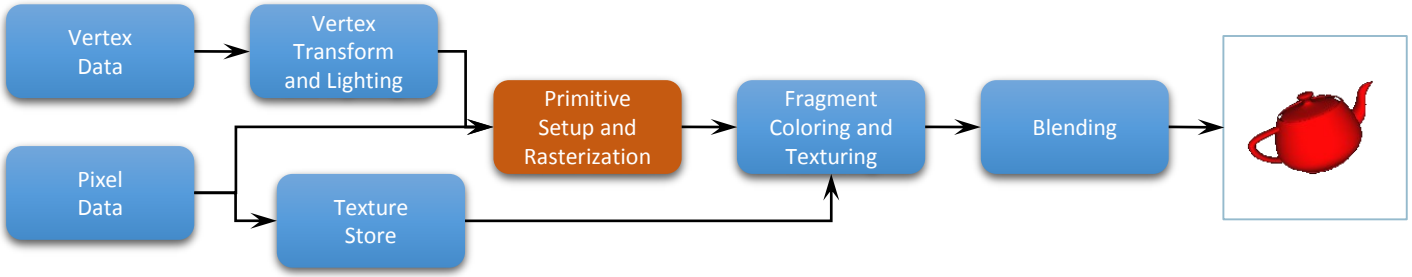
From http://www.slideshare.net/AmazonWebServices/transparency-and-control-with-aws-cloudtrail-and-aws-config and http://www.slideshare.net/abhijeetdesai/google-cluster-architecture

OpenGL

Computer Graphics

In the Beginning …

- OpenGL 1.0 was released on July 1st, 1994
- Its pipeline was entirely *fixed-function*
  - the only operations available were fixed by the implementation

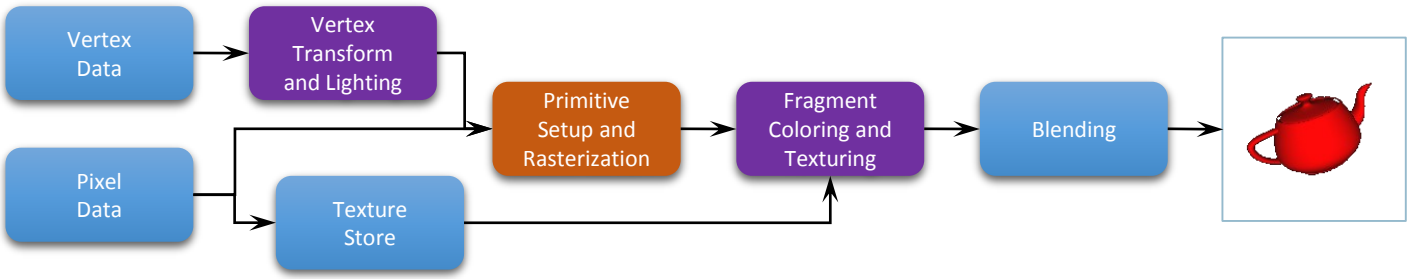


- The pipeline evolved
  - but remained based on fixed-function operation through OpenGL versions 1.1 through 2.0 (Sept. 2004)

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

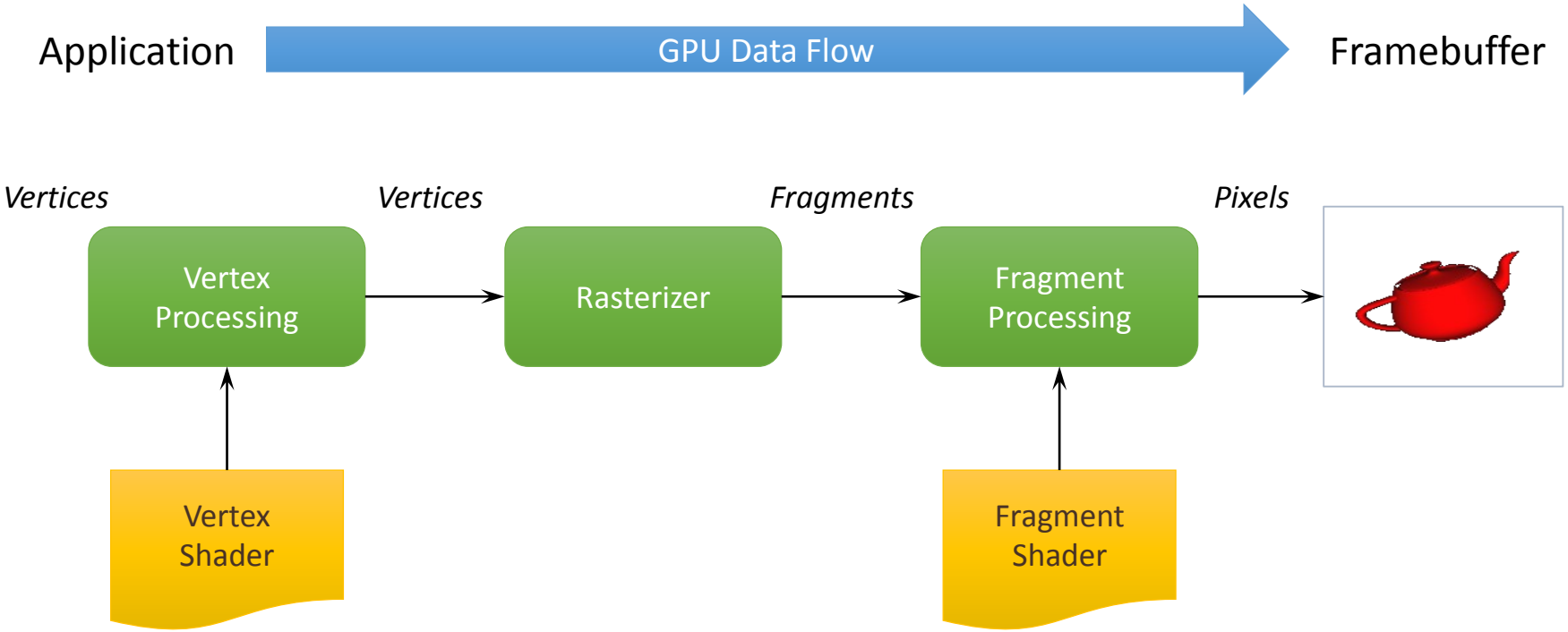
Beginnings of The Programmable Pipeline

- OpenGL 2.0 (officially) added programmable shaders
  - *vertex shading* augmented the fixed-function transform and lighting stage
  - *fragment shading* augmented the fragment coloring stage
- However, the fixed-function pipeline was still available



Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

A Simplified Pipeline Model



Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

OpenGL Programming in a Nutshell

- Modern OpenGL programs essentially do the following steps:
  - Create shader programs
  - Create buffer objects and load data into them
  - “Connect” data locations with shader variables
  - Render

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

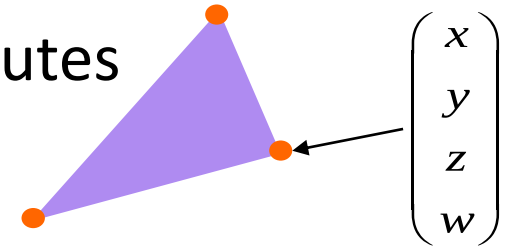
Application Framework Requirements

- OpenGL applications need a place to render into
  - usually an on-screen window
- Need to communicate with native windowing system
- Each windowing system interface is different
- We use GLUT (more specifically, freeglut)
  - simple, open-source library that works everywhere
  - handles all windowing operations:
    - opening windows
    - input processing

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Representing Geometric Objects

- Geometric objects are represented using *vertices*
- A vertex is a collection of generic attributes
  - positional coordinates
  - colors
  - texture coordinates
  - any other data associated with that point in space
- Position stored in 4 dimensional homogeneous coordinates
- Vertex data must be stored in vertex buffer objects (VBOs)
- VBOs must be stored in vertex array objects (VAOs)

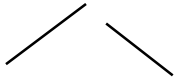


Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

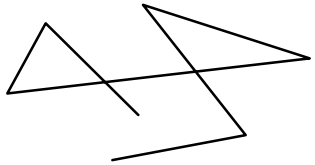
OpenGL's Geometric Primitives

- All primitives are specified by vertices

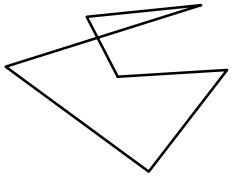
GL_POINTS



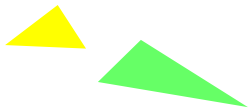
GL_LINES



GL_LINE_STRIP



GL_LINE_LOOP



GL_TRIANGLES

GL_TRIANGLE_STRIP

GL_TRIANGLE_FAN

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Our First Program

- We'll render a cube with colors at each vertex
- Our example demonstrates:
  - initializing vertex data
  - organizing data for rendering
  - simple object modeling
    - building up 3D objects from geometric primitives
    - building geometric primitives from vertices

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Initializing the Cube's Data

- We'll build each cube face from individual triangles
- Need to determine how much storage is required
  - (6 faces)(2 triangles/face)(3 vertices/triangle)

```
const int NumVertices = 36;
```

- To simplify communicating with GLSL, we'll use a `vec4` class (implemented in C++) similar to GLSL's `vec4` type
  - we'll also typedef it to add logical meaning

```
typedef  vec4  point4;
typedef  vec4  color4;
```

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Initializing the Cube's Data (cont'd)

- Before we can initialize our VBO, we need to stage the data
- Our cube has two attributes per vertex
  - position
  - color
- We create two arrays to hold the VBO data

```
point4  vPositions[NumVertices];
color4  vColors[NumVertices];
```

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Cube Data

- Vertices of a unit cube centered at origin
  - sides aligned with axes

```
point4 positions[8] = {
    point4( -0.5, -0.5,  0.5, 1.0 ),
    point4( -0.5,  0.5,  0.5, 1.0 ),
    point4(  0.5,  0.5,  0.5, 1.0 ),
    point4(  0.5, -0.5,  0.5, 1.0 ),
    point4( -0.5, -0.5, -0.5, 1.0 ),
    point4( -0.5,  0.5, -0.5, 1.0 ),
    point4(  0.5,  0.5, -0.5, 1.0 ),
    point4(  0.5, -0.5, -0.5, 1.0 )
};
```

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Cube Data (cont'd)

- We'll also set up an array of RGBA colors

```
color4 colors[8] = {
    color4( 0.0, 0.0, 0.0, 1.0 ),  // black
    color4( 1.0, 0.0, 0.0, 1.0 ),  // red
    color4( 1.0, 1.0, 0.0, 1.0 ),  // yellow
    color4( 0.0, 1.0, 0.0, 1.0 ),  // green
    color4( 0.0, 0.0, 1.0, 1.0 ),  // blue
    color4( 1.0, 0.0, 1.0, 1.0 ),  // magenta
    color4( 1.0, 1.0, 1.0, 1.0 ),  // white
    color4( 0.0, 1.0, 1.0, 1.0 )   // cyan
};
```

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Generating a Cube Face from Vertices

- To simplify generating the geometry, we use a convenience function quad( )
  - create two triangles for each face and assigns colors to the vertices

```
int Index = 0;  // global variable indexing into VBO arrays

void quad( int a, int b, int c, int d )
{
    vColors[Index] = colors[a]; vPositions[Index] = positions[a];
Index++;
    vColors[Index] = colors[b]; vPositions[Index] = positions[b];
Index++;
    vColors[Index] = colors[c]; vPositions[Index] = positions[c];
Index++;
    vColors[Index] = colors[a]; vPositions[Index] = positions[a];
Index++;
    vColors[Index] = colors[c]; vPositions[Index] = positions[c];
Index++;
    vColors[Index] = colors[d]; vPositions[Index] = positions[d];
Index++;
}
```

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Generating the Cube from Faces

- Generate 12 triangles for the cube
  - 36 vertices with 36 colors

```
void colorcube()
{
    quad( 1, 0, 3, 2 );
    quad( 2, 3, 7, 6 );
    quad( 3, 0, 4, 7 );
    quad( 6, 5, 1, 2 );
    quad( 4, 5, 6, 7 );
    quad( 5, 4, 0, 1 );
}
```

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Vertex Array Objects (VAOs)

- VAOs store the data of an geometric object
- Steps in using a VAO
  - generate VAO names by calling `glGenVertexArrays()`
  - bind a specific VAO for initialization by calling `glBindVertexArray()`
  - update VBOs associated with this VAO
  - bind VAO for use in rendering
- This approach allows a single function call to specify all the data for an objects
  - previously, you might have needed to make many calls to make all the data current

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

VAOs in Code

- Create a vertex array object

```
GLuint vao;
glGenVertexArrays( 1, &vao );
glBindVertexArray( vao );
```

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Storing Vertex Attributes

- Vertex data must be stored in a VBO, and associated with a VAO
- The code-flow is similar to configuring a VAO
  - generate VBO names by calling `glGenBuffers()`
  - bind a specific VBO for initialization by calling

    ```
    glBindBuffer( GL_ARRAY_BUFFER, … )
    ```

  - load data into VBO using

    ```
    glBufferData( GL_ARRAY_BUFFER, … )
    ```

  - bind VAO for use in rendering `glBindVertexArray()`

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

VBOs in Code

- Create and initialize a buffer object

```
GLuint buffer;
glGenBuffers( 1, &buffer );
glBindBuffer( GL_ARRAY_BUFFER, buffer );
glBufferData( GL_ARRAY_BUFFER,
              sizeof(vPositions) +
sizeof(vColors),
              NULL, GL_STATIC_DRAW );
glBufferSubData( GL_ARRAY_BUFFER, 0,
                 sizeof(vPositions), vPositions
);
glBufferSubData( GL_ARRAY_BUFFER,
sizeof(vPositions),
                 sizeof(vColors), vColors );
```

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Connecting Vertex Shaders with Geometric Data

- Application vertex data enters the OpenGL pipeline through the vertex shader
- Need to connect vertex data to shader variables
  - requires knowing the attribute location
- Attribute location can either be queried by calling `glGetVertexAttribLocation()`

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Vertex Array Code

- Associate shader variables with vertex arrays
  - do this after shaders are loaded

```
GLuint vPosition =
   glGetAttribLocation( program, “vPosition" );
glEnableVertexAttribArray( vPosition );
glVertexAttribPointer( vPosition, 4, GL_FLOAT,
   GL_FALSE, 0,BUFFER_OFFSET(0) );

GLuint vColor =
   glGetAttribLocation( program,"vColor" );
glEnableVertexAttribArray( vColor );
glVertexAttribPointer( vColor, 4, GL_FLOAT,
   GL_FALSE, 0,BUFFER_OFFSET(sizeof(vPositions)) );
```

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Drawing Geometric Primitives

- For contiguous groups of vertices

```
glDrawArrays( GL_TRIANGLES, 0, NumVertices );
```

- Usually invoked in display callback
- Initiates vertex shader

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Simple Vertex Shader for Cube Example

```
#version 430

in vec4 vPosition;
in vec4 vColor;

out vec4 color;

void main()
{
    color = vColor;
    gl_Position = vPosition;
}
```

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

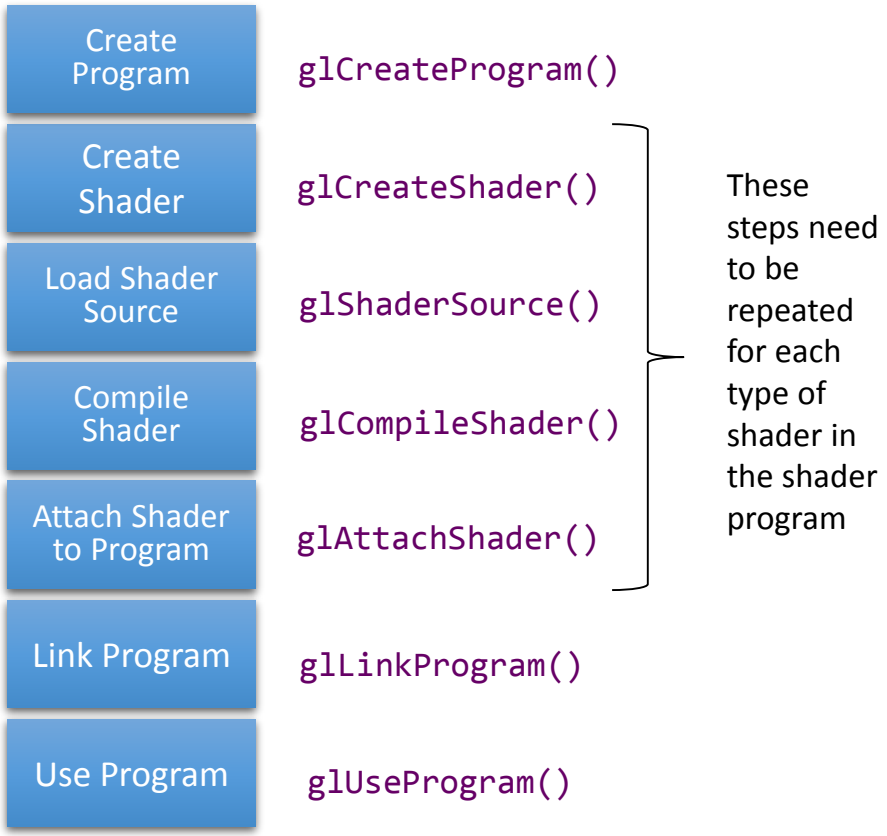
The Simplest Fragment Shader

```
#version 430

in vec4 color;

out vec4 fColor; // fragment's final color

void main()
{
    fColor = color;
}
```

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Getting Your Shaders into OpenGL

- Shaders need to be compiled and linked to form an executable shader program
- OpenGL provides the compiler and linker
- A program must contain
  - vertex and fragment shaders
  - other shaders are optional



Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Determining Locations After Linking

- Assumes you already know the variables' names

```
GLint loc = glGetAttribLocation( program,
"name" );

GLint loc = glGetUniformLocation( program,
"name" );
```

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Initializing Uniform Variable Values

- Uniform Variables

```
glUniform4f( index, x, y, z, w );

GLboolean  transpose = GL_TRUE;

// Since we're C programmers
GLfloat  mat[3][4][4] = { … };
glUniformMatrix4fv( index, 3, transpose, mat );
```

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Finishing the Cube Program

```
int main( int argc, char **argv )
{
    glutInit( &argc, argv );
    glutInitDisplayMode( GLUT_RGBA | GLUT_DOUBLE |GLUT_DEPTH
);
    glutInitWindowSize( 512, 512 );
    glutCreateWindow( "Color Cube" );

    glewInit();
    init();

    glutDisplayFunc( display );
    glutKeyboardFunc( keyboard );
    glutMainLoop();

    return 0;
}
```

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

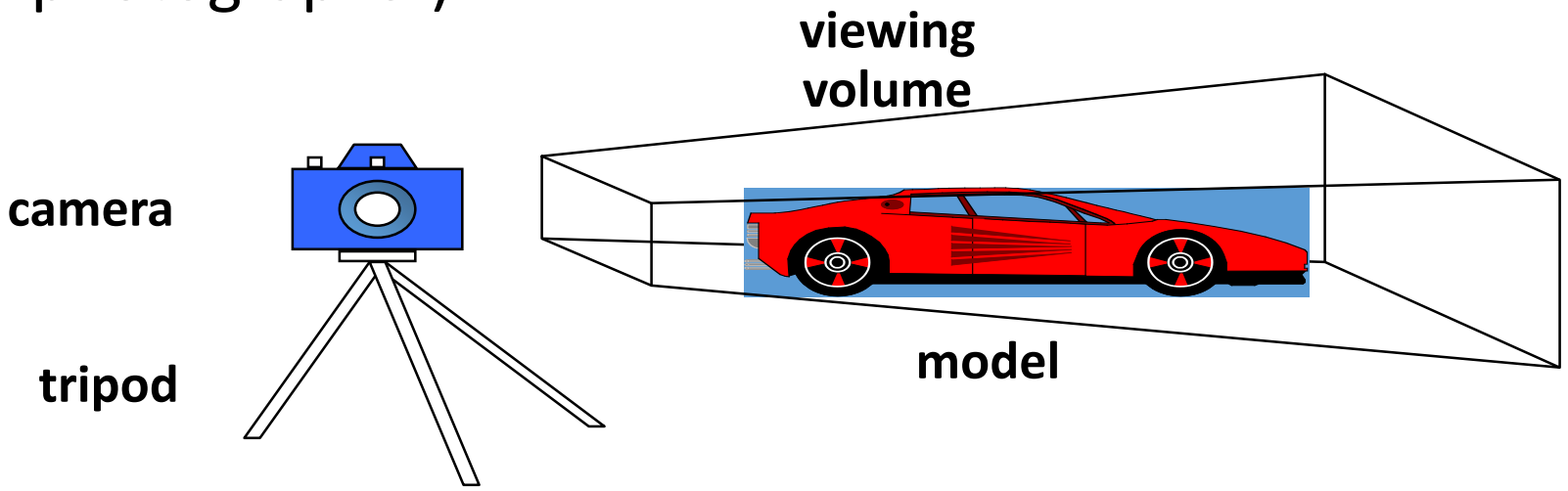
Cube Program's GLUT Callbacks

```
void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT
);
    glDrawArrays( GL_TRIANGLES, 0, NumVertices );
    glutSwapBuffers();
}

void keyboard( unsigned char key, int x, int y )
{
    switch( key ) {
        case 033: case 'q': case 'Q':
            exit( EXIT_SUCCESS );
            break;
    }
}
```

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Camera Analogy

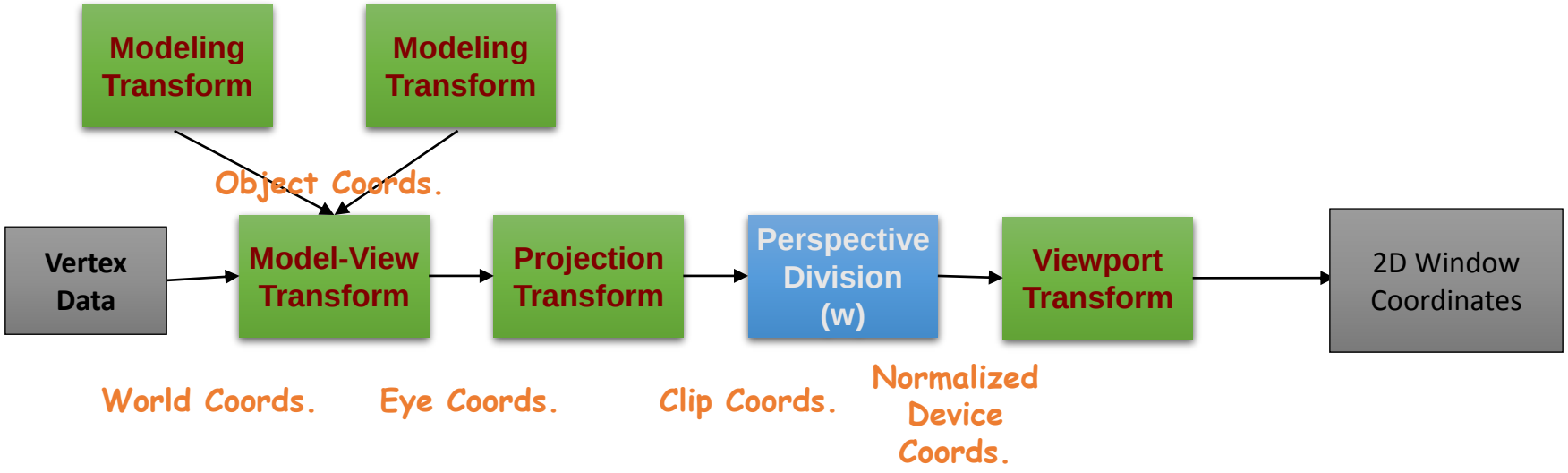
- 3D is just like taking a photograph (lots of photographs!)



Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Transformations

- Transformations take us from one “space” to another
  - All of our transforms are 4×4 matrices



Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Camera Analogy and Transformations

- Projection transformations
  - adjust the lens of the camera
- Viewing transformations
  - tripod–define position and orientation of the viewing volume in the world
- Modeling transformations
  - moving the model
- Viewport transformations
  - enlarge or reduce the physical photograph

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

3D Transformations

- A vertex is transformed by 4×4 matrices
  - all affine operations are matrix multiplications

- All matrices are stored column-major in OpenGL
  - this is opposite of what “C” programmers expect

- Matrices are always post-multiplied
  - product of matrix and vector is $\mathbf{M}\vec{v}$

$$\mathbf{M} = \begin{bmatrix} m_0 & m_4 & m_8 & m_{12} \\ m_1 & m_5 & m_9 & m_{13} \\ m_2 & m_6 & m_{10} & m_{14} \\ m_3 & m_7 & m_{11} & m_{15} \end{bmatrix}$$

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Specifying What You Can See

- Set up a viewing frustum to specify how much of the world we can see
- Done in two steps
  - specify the size of the frustum (projection transform)
  - specify its location in space (model-view transform)
- Anything outside of the viewing frustum is clipped
  - primitive is either modified or discarded (if entirely outside frustum)

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

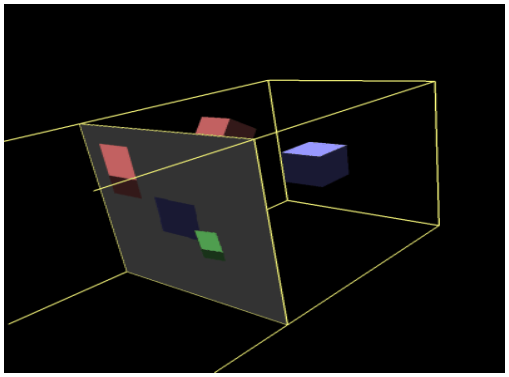
Specifying What You Can See (cont'd)

- OpenGL projection model uses eye coordinates
  - the "eye" is located at the origin
  - looking down the -z axis
- Projection matrices use a six-plane model:
  - near (image) plane and far (infinite) plane
    - both are distances from the eye (positive values)
  - enclosing planes
    - top & bottom, left & right

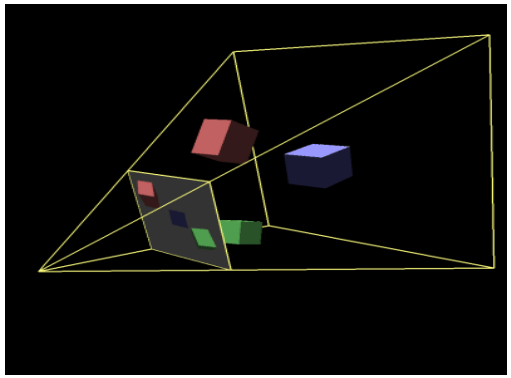
Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Specifying What You Can See (cont'd)

Orthographic View



Perspective View



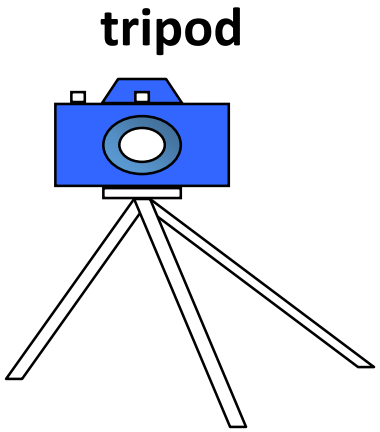
$$O = \begin{pmatrix} \frac{2}{r-l} & 0 & 0 & \frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & \frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & \frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$P = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Viewing Transformations

- Position the camera/eye in the scene
  - place the tripod down; aim camera
- To “fly through” a scene
  - change viewing transformation and redraw scene
- `LookAt( eyex, eyey, eyez, lookx, looky, lookz, upx, upy, upz )`
  - up vector determines unique orientation
  - careful of degenerate positions



Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Creating the LookAt Matrix

$$\hat{n} = \frac{\overrightarrow{look} - \overrightarrow{eye}}{\left\| \overrightarrow{look} - \overrightarrow{eye} \right\|}$$

$$\hat{u} = \frac{\hat{n} \times \overrightarrow{up}}{\left\| \hat{n} \times \overrightarrow{up} \right\|}$$

$$\hat{v} = \hat{u} \times \hat{n}$$

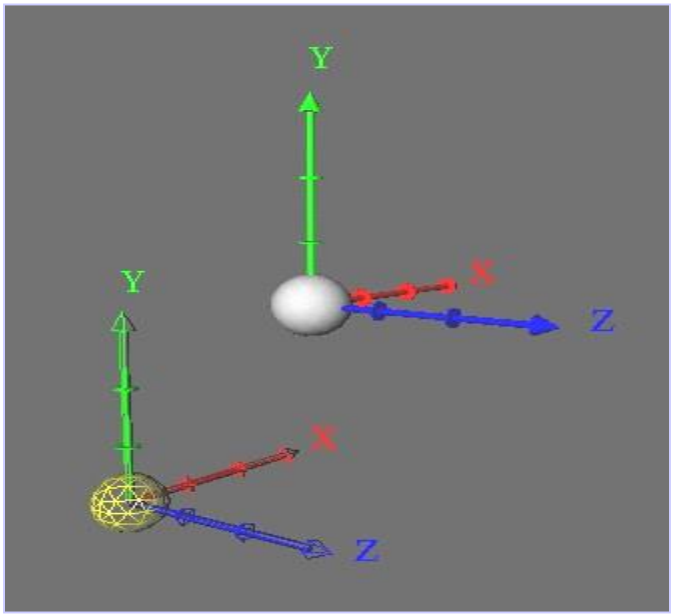
$$\Rightarrow \begin{pmatrix} u_x & u_y & u_z & -(eye \cdot \vec{u}) \\ v_x & v_y & v_z & -(eye \cdot \vec{v}) \\ -n_x & -n_y & -n_z & -(eye \cdot \vec{n}) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Translation

- Move the origin to a new location

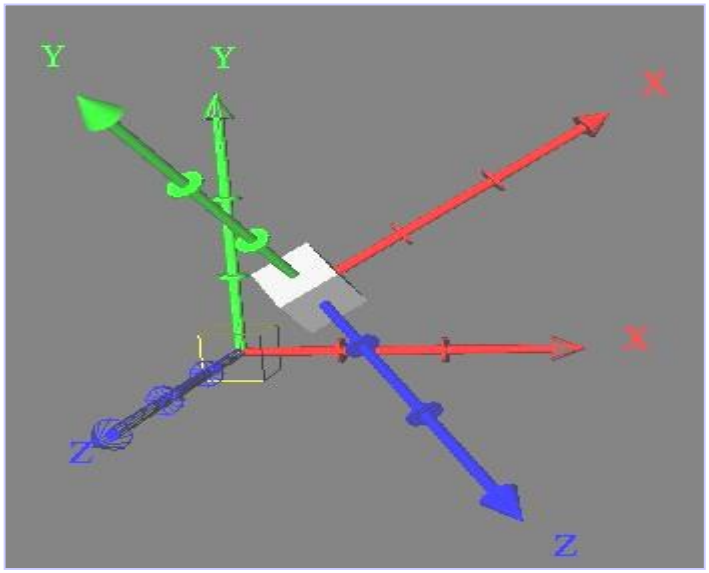
$$T(t_x, t_y, t_z) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Rotation

- Rotate coordinate system about an axis in space



Note, there's a translation applied here to make things easier to see

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Rotation (cont'd)

$$\vec{v} = (x \quad y \quad z)$$

$$\vec{u} = \frac{\vec{v}}{\|\vec{v}\|} = (x' \quad y' \quad z')$$

$$M = \vec{u}^t\vec{u} + \cos(q)(I - \vec{u}^t\vec{u}) + \sin(q)S$$

$$S = \begin{pmatrix} 0 & -z' & y' \\ z' & 0 & -x' \\ -y' & x' & 0 \end{pmatrix}$$

$$R_{\vec{v}}(q) = \begin{pmatrix} & & & 0 \\ & M & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Vertex Shader for Rotation of Cube

```
in vec4 vPosition;
in vec4 vColor;
out vec4 color;
uniform vec3 theta;

void main()
{
    // Compute the sines and cosines of theta for
    // each of the three axes in one computation.
    vec3 angles = radians( theta );
    vec3 c = cos( angles );
    vec3 s = sin( angles );
```

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Vertex Shader for Rotation of Cube (cont'd)

```
// Remember: these matrices are column-major

mat4 rx = mat4( 1.0,  0.0,  0.0, 0.0,
                0.0,  c.x,  s.x, 0.0,
                0.0, -s.x,  c.x, 0.0,
                0.0,  0.0,  0.0, 1.0 );

mat4 ry = mat4( c.y, 0.0, -s.y, 0.0,
                0.0, 1.0,  0.0, 0.0,
                s.y, 0.0,  c.y, 0.0,
                0.0, 0.0,  0.0, 1.0 );
```

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Vertex Shader for Rotation of Cube (cont'd)

```
    mat4 rz = mat4( c.z, -s.z, 0.0, 0.0,
                    s.z,  c.z, 0.0, 0.0,
                    0.0,  0.0, 1.0, 0.0,
                    0.0,  0.0, 0.0, 1.0 );

    color = vColor;
    gl_Position = rz * ry * rx * vPosition;
}
```

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Sending Angles from Application

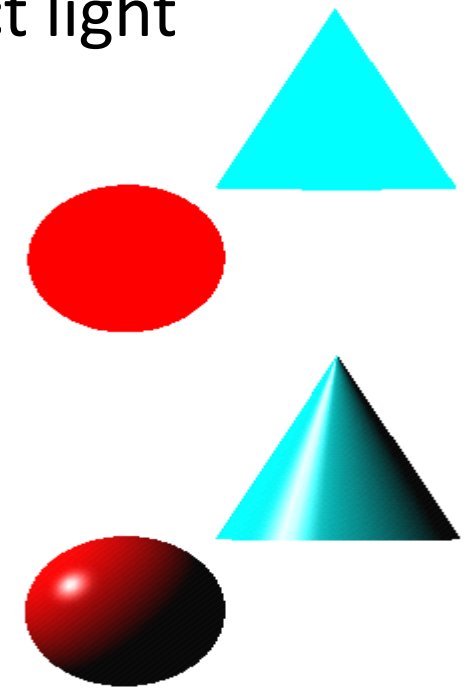
- Here, we compute our angles (`Theta`) in our mouse callback

```
GLuint theta;  // theta uniform location
vec3  Theta;   // Axis angles

void display( void )
{
   glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT
);

   glUniform3fv( theta, 1, Theta );
   glDrawArrays( GL_TRIANGLES, 0, NumVertices );

   glutSwapBuffers();
}
```

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Lighting Principles

- Lighting simulates how objects reflect light
  - material composition of object
  - light's color and position
  - global lighting parameters
- Usually implemented in
  - vertex shader for faster speed
  - fragment shader for nicer shading



Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Modified Phong Model

- Computes a color for each vertex using
  - Surface normals
  - Diffuse and specular reflections
  - Viewer's position and viewing direction
  - Ambient light
  - Emission
- Vertex colors are interpolated across polygons by the rasterizer
  - *Phong shading* does the same computation per pixel, interpolating the normal across the polygon
    - more accurate results

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Fragment Shaders

- A shader that's executed for each "potential" pixel
  - fragments still need to pass several tests before making it to the framebuffer
- There are lots of effects we can do in fragment shaders
  - Per-fragment lighting
  - Texture and bump Mapping
  - Environment (Reflection) Maps

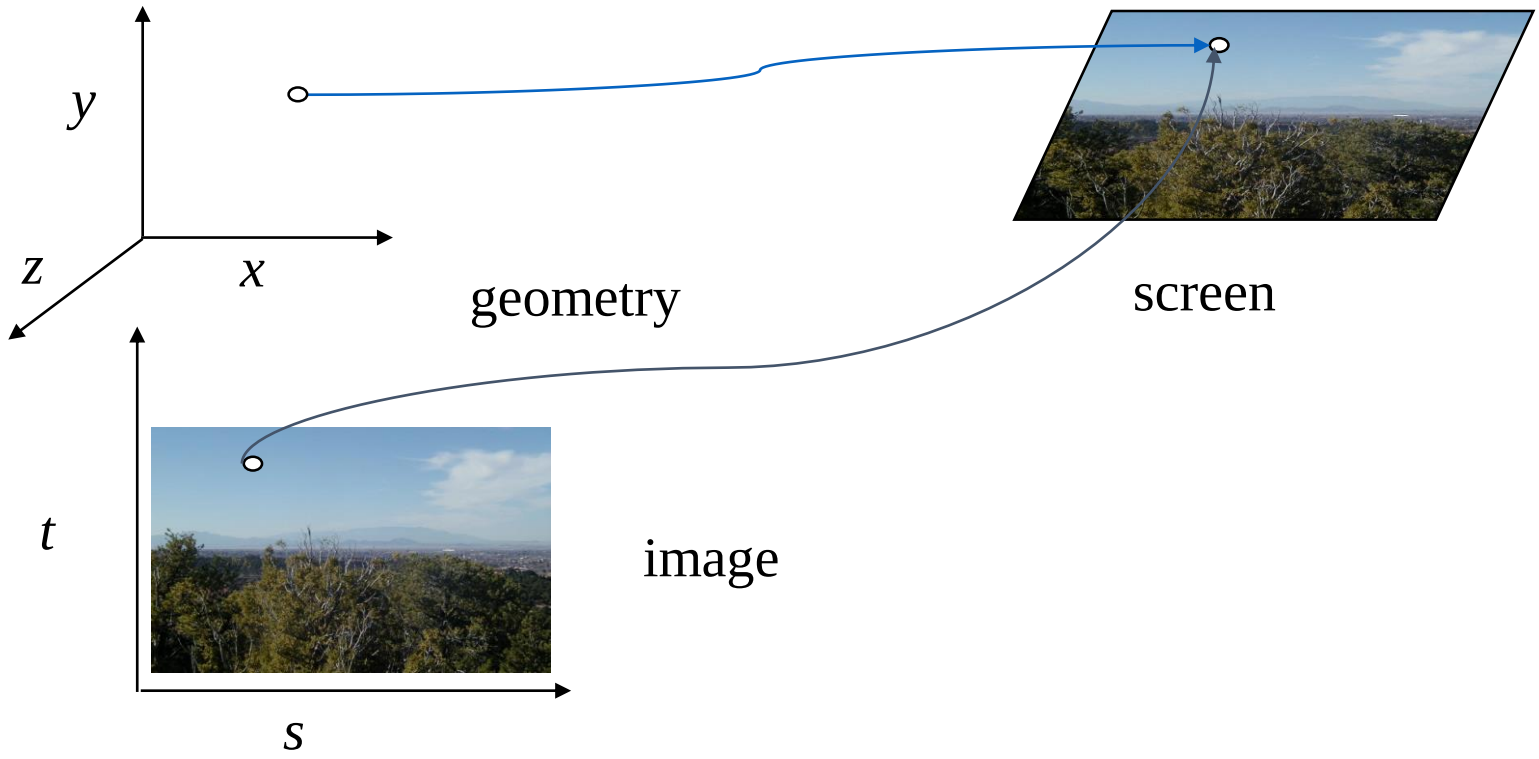
Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Shader Examples

- Vertex Shaders
  - Moving vertices: height fields
  - Per vertex lighting: height fields
  - Per vertex lighting: cartoon shading
- Fragment Shaders
  - Per vertex vs. per fragment lighting: cartoon shader
  - Samplers: reflection Map
  - Bump mapping

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

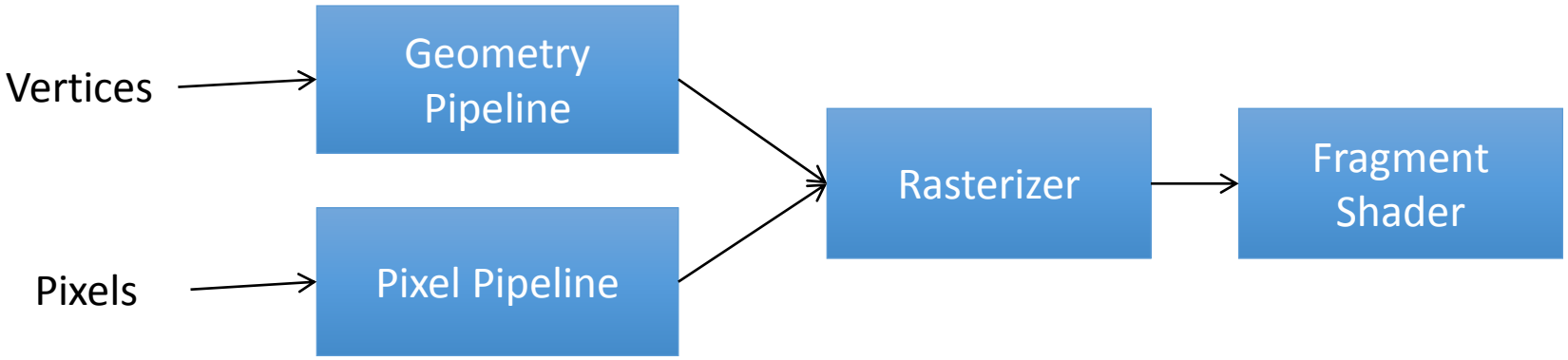
Texture Mapping



Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Texture Mapping and the OpenGL Pipeline

- Images and geometry flow through separate pipelines that join at the rasterizer
  - "complex" textures do not affect geometric complexity



Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Applying Textures

- Three basic steps to applying a texture
  1. specify the texture
     - read or generate image
     - assign to texture
     - enable texturing
  2. assign texture coordinates to vertices
  3. specify texture parameters
     - wrapping, filtering

Ed Angel and Dave Shreiner, Presentation at SIGGRAPH 2013
https://www.cs.unm.edu/~angel/SIGGRAPH13/An%20Introduction%20to%20OpenGL%20Programming.pptx

Ray Tracing and Volume Rendering

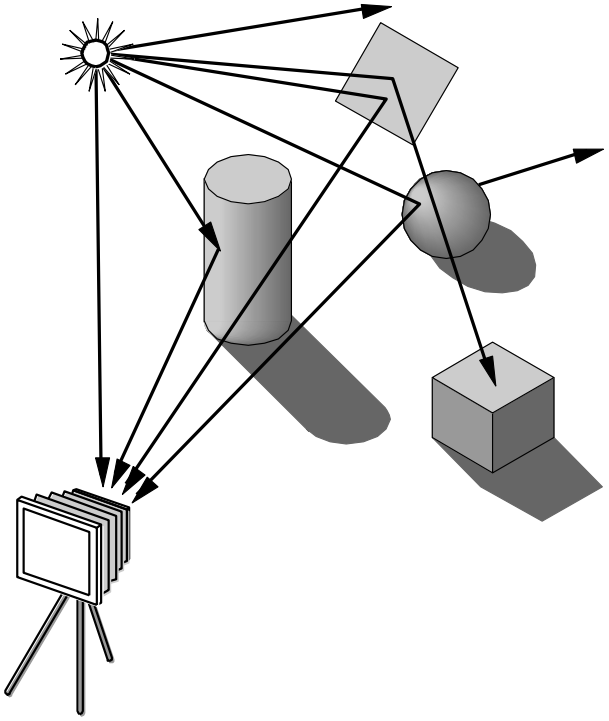
Advanced Graphics

Introduction

- OpenGL is based on a pipeline model in which primitives are rendered one at time
  - No shadows (except by tricks or multiple renderings)
  - No multiple reflections
- Global approaches
  - Rendering equation
  - Ray tracing
  - Radiosity
- Commercial Ray Tracing Libraries
  - Optix from Nvidia
  - OSPRay and Embree from Intel

Ray Tracing

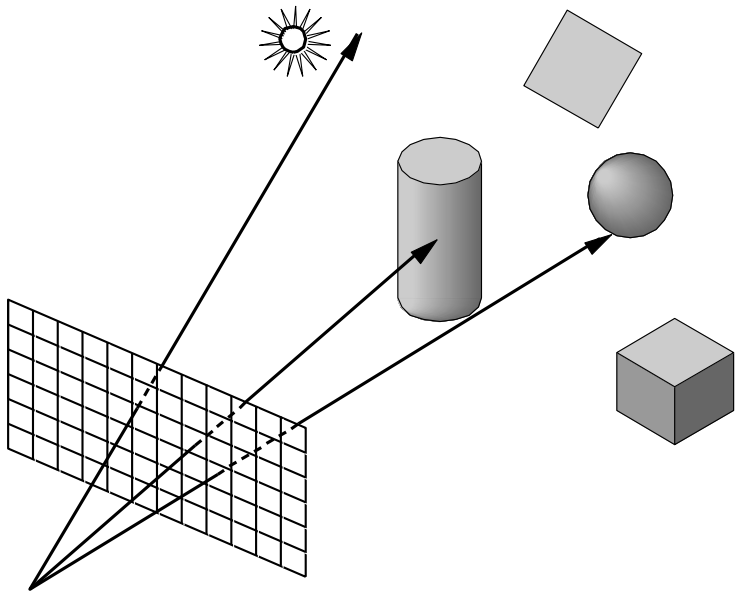
- Follow rays of light from a point source
- Can account for reflection and transmission



Ed Angel, Interactive Computer Graphics 5E, 2009: http://www.cs.utsa.edu/~jpq/Site/teaching/cg-s11/raytracing.ppt

Ray Casting

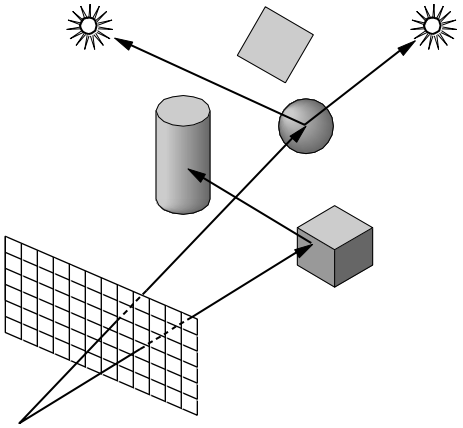
- Only rays that reach the eye matter
- Reverse direction and cast rays
- Need at least one ray per pixel



Ed Angel, Interactive Computer Graphics 5E, 2009: http://www.cs.utsa.edu/~jpq/Site/teaching/cg-s11/raytracing.ppt

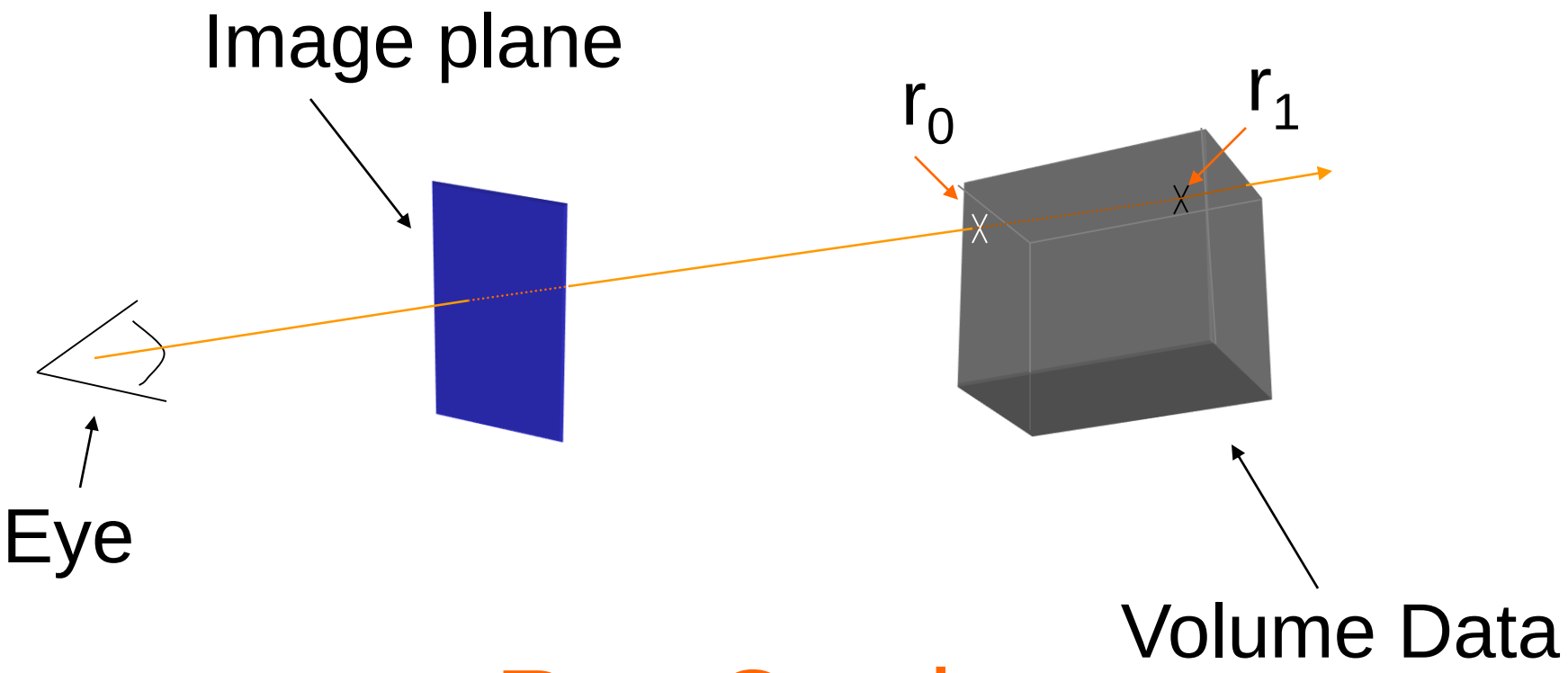
Shadow Rays

- Even if a point is visible, it will not be lit unless we can see a light source from that point
- Cast shadow or feeler rays



Ed Angel, Interactive Computer Graphics 5E, 2009: http://www.cs.utsa.edu/~jpq/Site/teaching/cg-s11/raytracing.ppt

Rendering



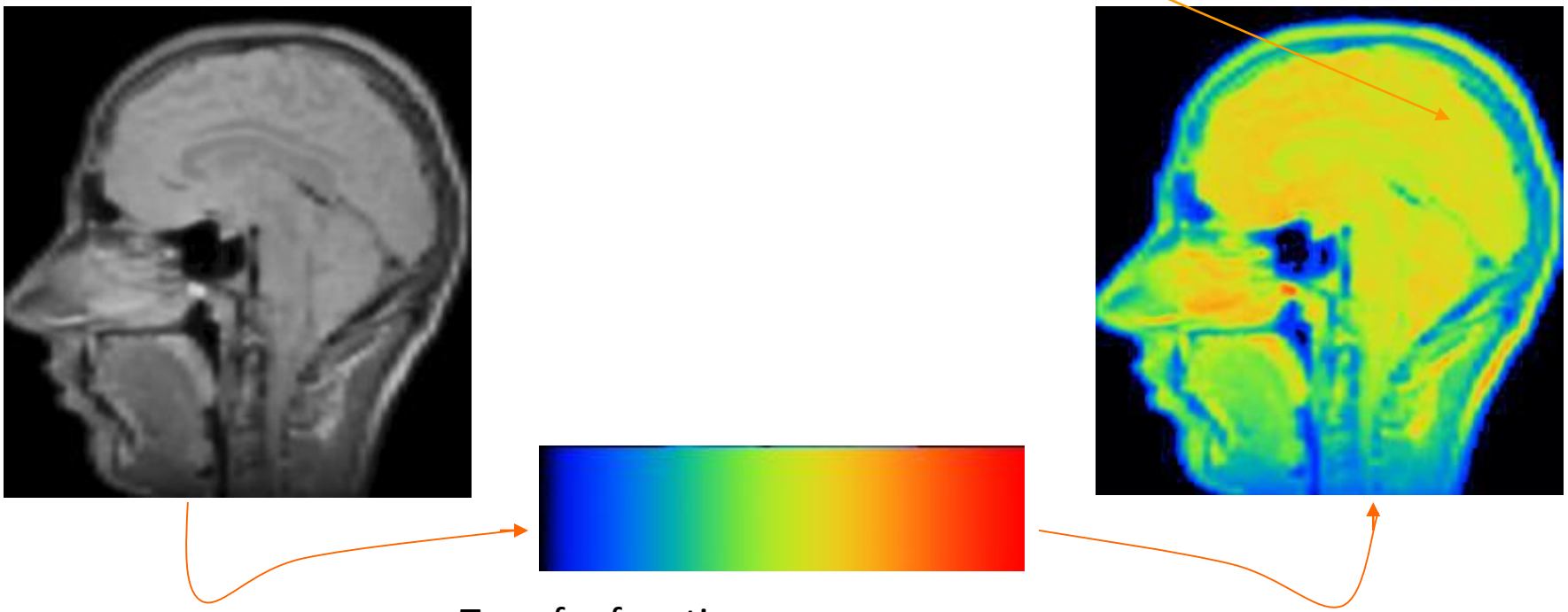
Ray Casting

But how do we get the final color??

Joe Michael Kniss, University of Utah:
http://www.cs.utah.edu/~jmk/papers/volumeRendering-g.ppt

Transfer Function

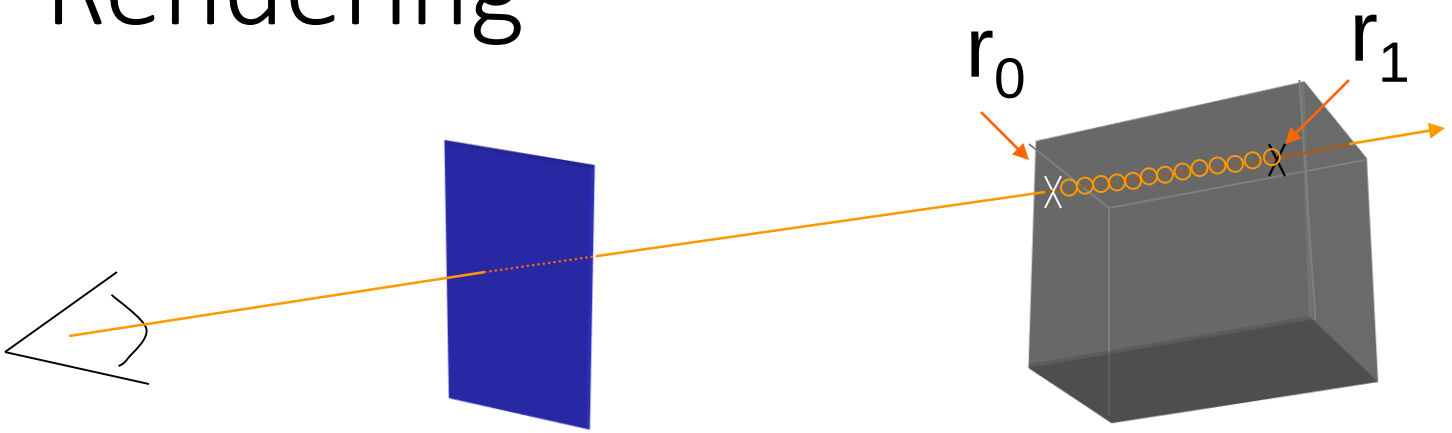
- Assign optical properties to data
  - Color
  - Opacity



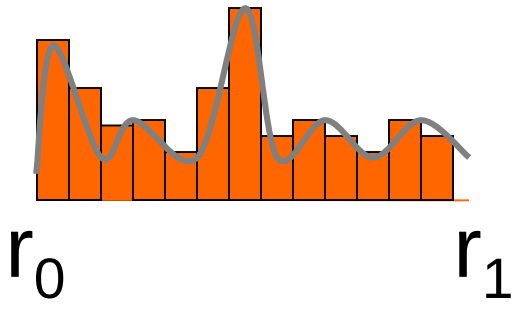
Transfer function

Joe Michael Kniss, University of Utah:
http://www.cs.utah.edu/~jmk/papers/volumeRendering-g.ppt

Rendering



Solution: Sum (Riemann)



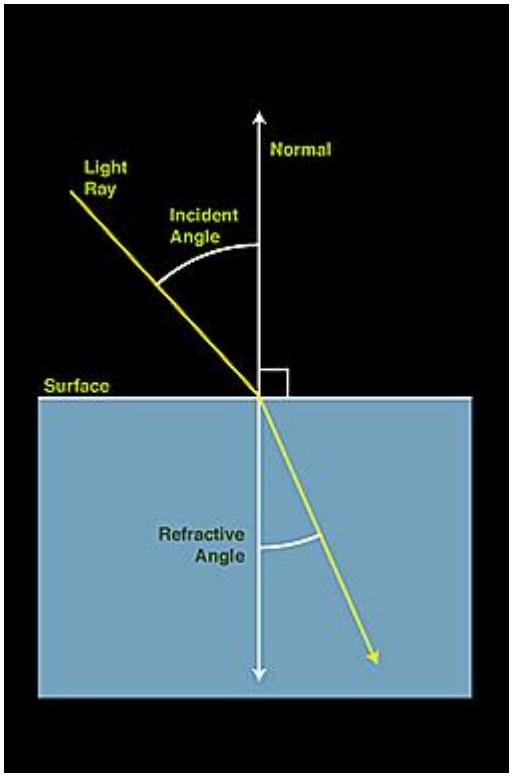
$$\int_{r_0}^{r_1} T(x)dx \rightarrow \sum_{i=0}^{n} T(x_i)\Delta x$$

Joe Michael Kniss, University of Utah:
http://www.cs.utah.edu/~jmk/papers/volumeRendering-g.ppt

Refraction

- Light bends as it moves through different materials
  - E.g., moving from air to water
  - Different speeds for different materials
    - See: Index of Refraction
- Snell's Law

  Ni * Sin(**Ai**) = **Nr** * Sin(**Ar**)



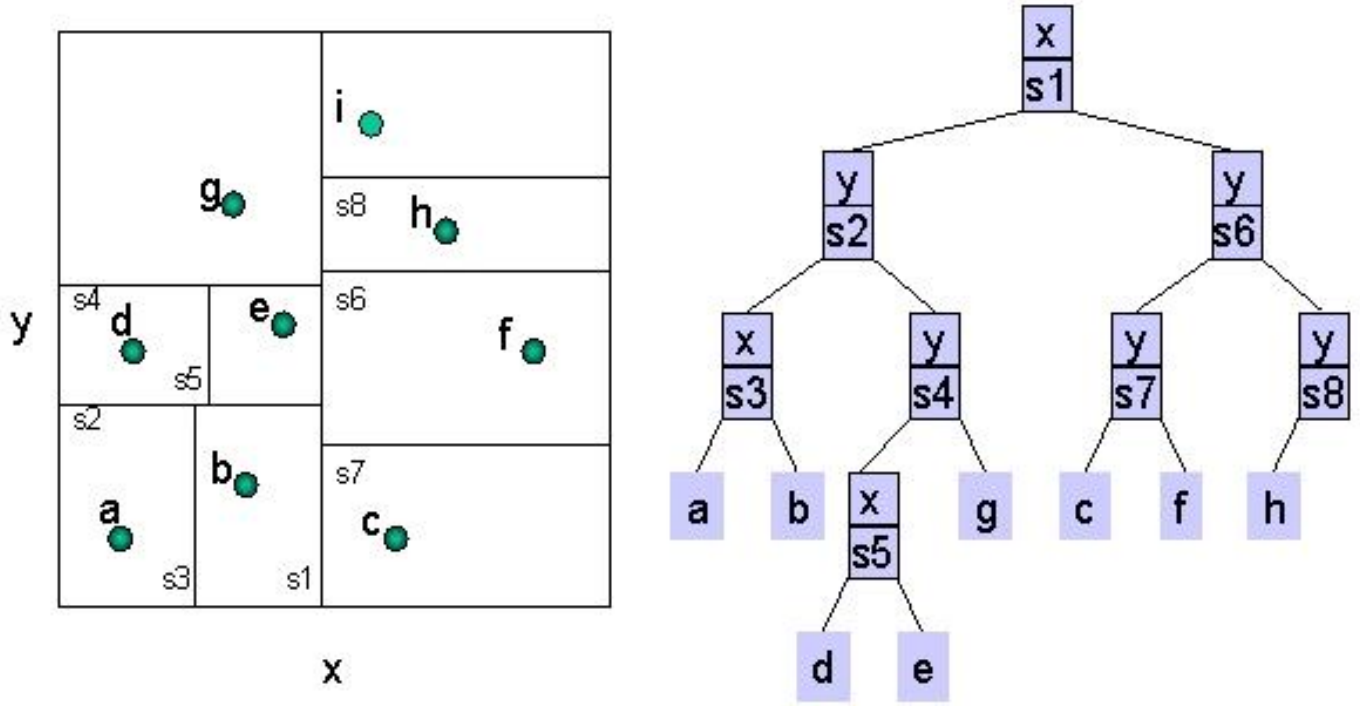
Ed Angel, Interactive Computer Graphics 5E, 2009: http://www.cs.utsa.edu/~jpq/Site/teaching/cg-s11/raytracing.ppt

Building a Ray Tracer

- Best expressed recursively
- Can remove recursion later
- Image based approach
  - For each ray …….
- Find intersection with closest surface
  - Need whole object database available
  - Complexity of calculation limits object types
- Compute lighting at surface
- Trace reflected and transmitted rays

Ed Angel, Interactive Computer Graphics 5E, 2009: http://www.cs.utsa.edu/~jpq/Site/teaching/cg-s11/raytracing.ppt

KD-Tree Search Structure



From https://courses.cs.washington.edu/courses/csep521/99sp/lectures/lecture15/sld031.htm

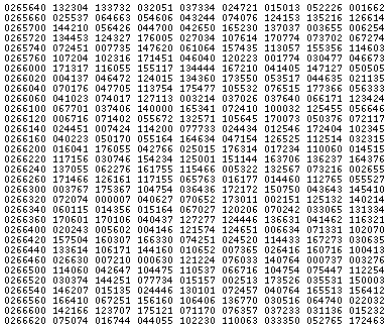
VTK

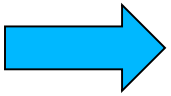
The Visualization Toolkit

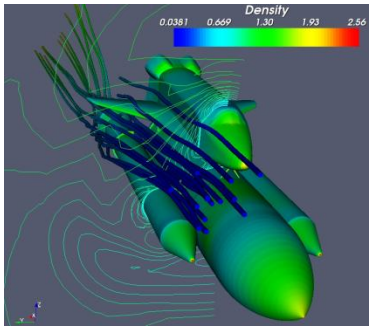
Introduction

• Visualization: converting raw data to a form that is viewable and understandable to humans.

• Scientific visualization: specifically concerned with data that has a well-defined representation in 2D or 3D space (e.g., from simulation mesh or scanner).







*Adapted from The ParaView Tutorial, Moreland

Robert Putnam, Class Presentation at Boston University, 2010
www.bu.edu/tech/files/2010/10/VTK-Fall-2010.ppt

VTK

Visualization Toolkit

Open source

Set of object-oriented class libraries for visualization and data analysis

Several language interfaces

- C++
- Tcl
- Java
- Python

Portable (MS Windows, Linux, OSX)

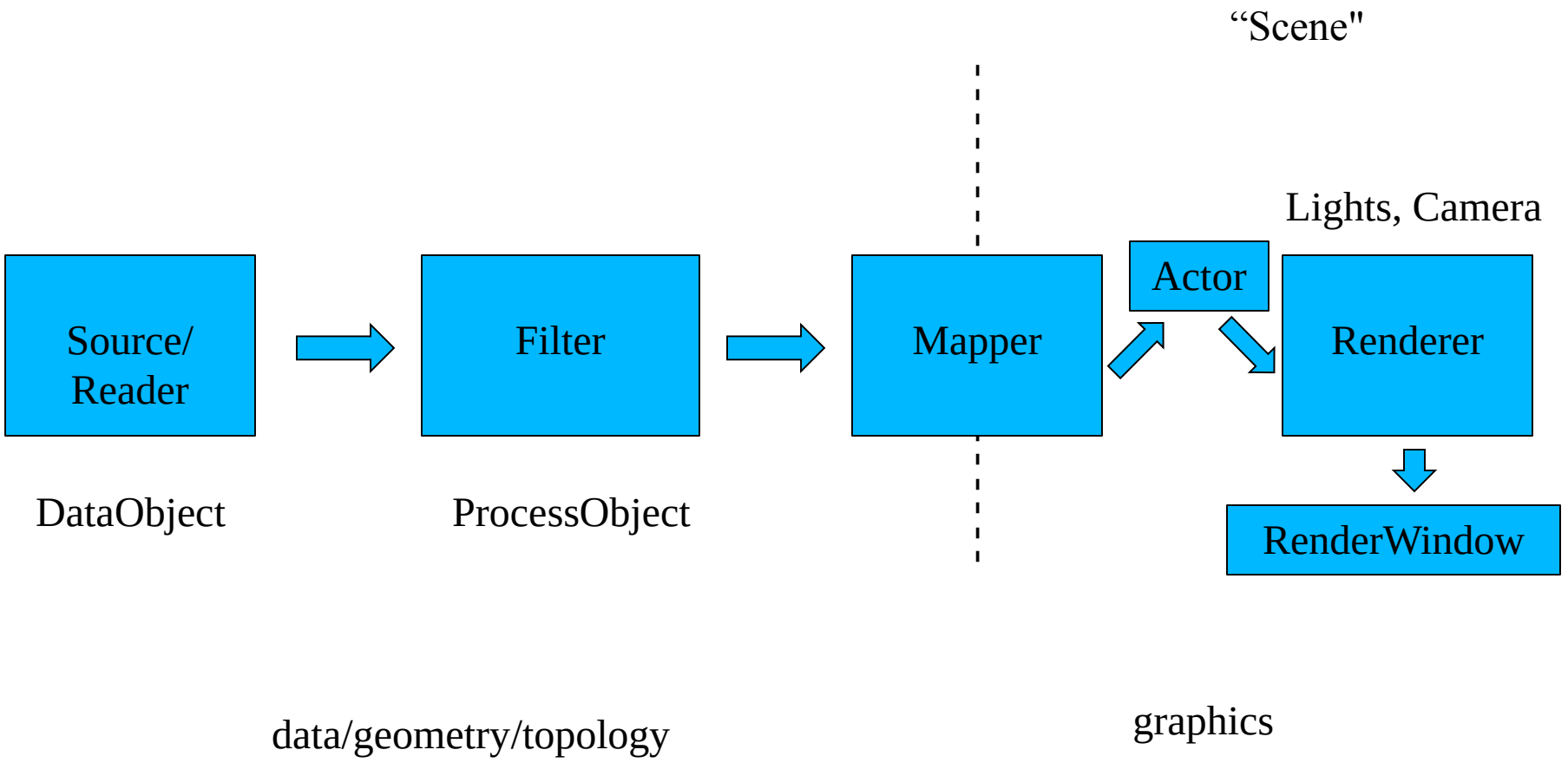
Active developer community

Good documentation available, free and otherwise

Professional support services available from Kitware

Robert Putnam, Class Presentation at Boston University, 2010
www.bu.edu/tech/files/2010/10/VTK-Fall-2010.ppt

VTK terminology/model



Robert Putnam, Class Presentation at Boston University, 2010
www.bu.edu/tech/files/2010/10/VTK-Fall-2010.ppt

Pipeline -> Sample Code

```
vtkStructuredGridReader reader
  reader SetFileName "density.vtk"
  reader Update
```

Reader

```
vtkContourFilter iso
  iso SetInputConnection [reader GetOutputPort]
  iso SetValue 0 .26
```

Filter

```
vtkPolyDataMapper isoMapper
  isoMapper SetInputConnection [iso GetOutputPort]
```

Mapper

```
vtkActor isoActor
  isoActor SetMapper isoMapper
```

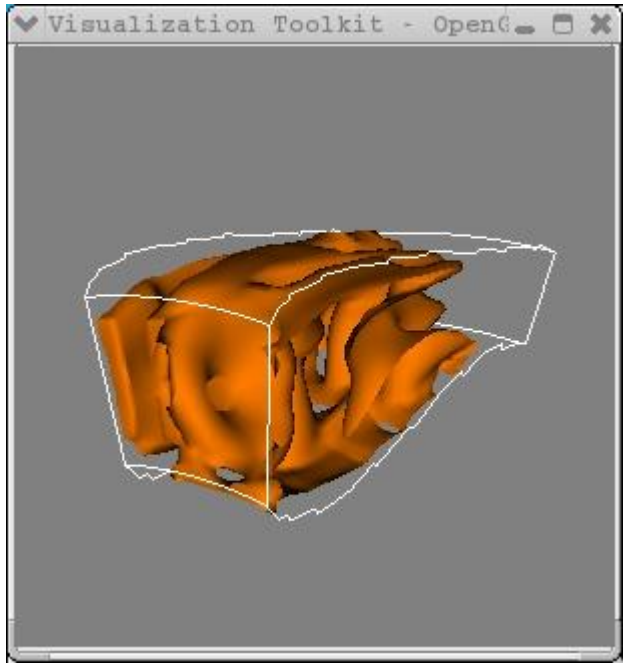
Actor

```
vtkRenderer ren1
  ren1 AddActor isoActor
```

Renderer

```
vtkRenderWindow renWin
  renWin AddRenderer ren1
  renWin SetSize 500 500
  renWin Render
```

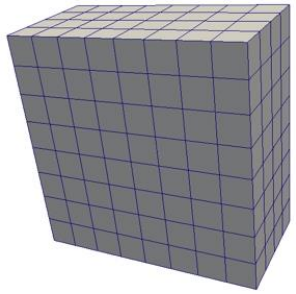
RenderWindow



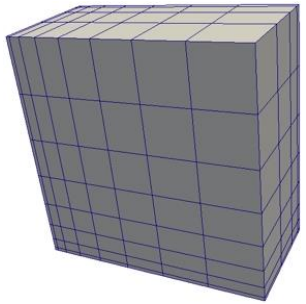
Robert Putnam, Class Presentation at Boston University, 2010
www.bu.edu/tech/files/2010/10/VTK-Fall-2010.ppt

Examples of Dataset Types

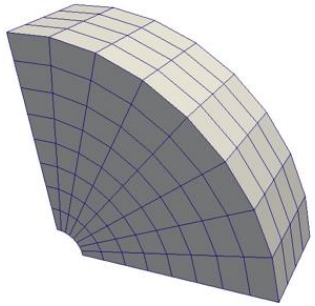
- **Structured Points (Image Data)**
  - regular in both topology and geometry
  - examples: lines, pixels, voxels
  - applications: imaging CT, MRI



- **Rectilinear Grid**
  - regular topology but geometry only partially regular
  - examples: pixels, voxels



- **Structured Grid (Curvilinear)**
  - regular topology and irregular geometry
  - examples: quadrilaterals, hexahedron
  - applications: fluid flow, heat transfer



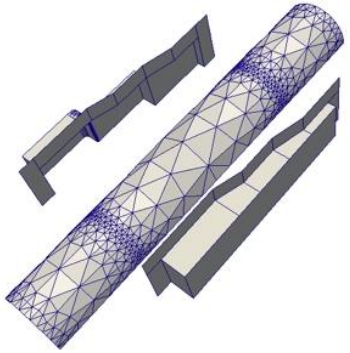
Robert Putnam, Class Presentation at Boston University, 2010
www.bu.edu/tech/files/2010/10/VTK-Fall-2010.ppt

Examples of Dataset Types (cont)

- **Polygonal Data**

  irregular in both topology and geometry

  examples: vertices, polyvertices, lines, polylines, polygons, triangle strips

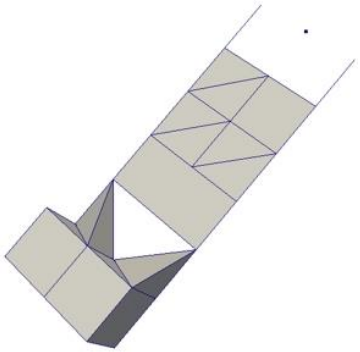


- **Unstructured Grid**

  irregular in both topology and geometry

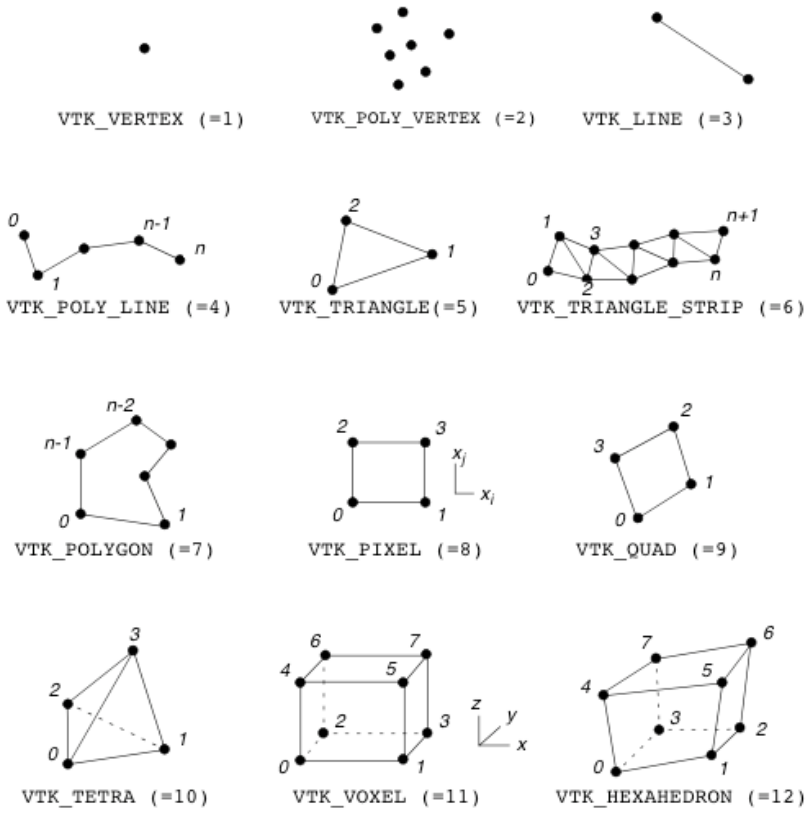
  examples: any combination of cells

  applications: finite element analysis, structural design, vibration



Robert Putnam, Class Presentation at Boston University, 2010
www.bu.edu/tech/files/2010/10/VTK-Fall-2010.ppt

Examples of Cell Types

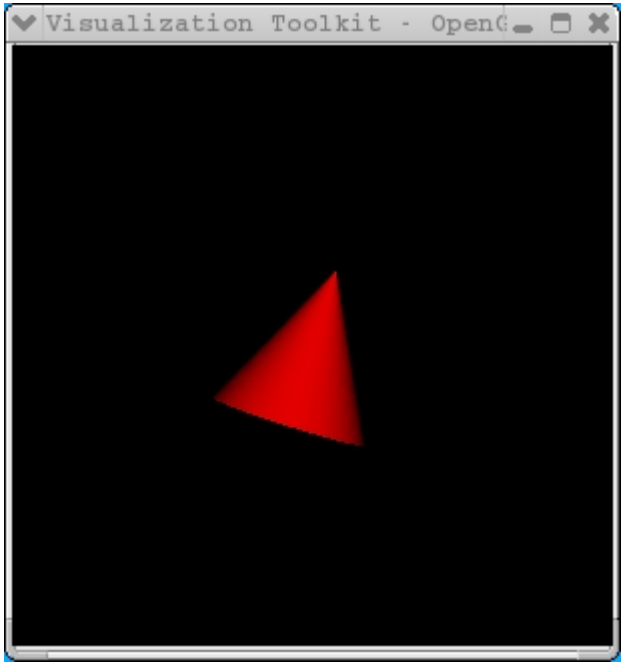


Robert Putnam, Class Presentation at Boston University, 2010
www.bu.edu/tech/files/2010/10/VTK-Fall-2010.ppt

Code – cone2.tcl

Editor cone2.tcl:

```
vtkConeSource cone
   cone SetResolution 100
vtkPolyDataMapper coneMapper
   coneMapper SetInput [cone GetOutput]
vtkActor coneActor
   coneActor SetMapper coneMapper
   [coneActor GetProperty] SetColor 1.0 0.0 0.0
vtkRenderer ren1
   ren1 SetBackground 0.0 0.0 0.0
   ren1 AddActor coneActor
vtkRenderWindow renWin
   renWin SetSize 500 500
   renWin AddRenderer ren1
vtkRenderWindowInteractor iren
   iren SetRenderWindow renWin
   iren Initialize
```



Robert Putnam, Class Presentation at Boston University, 2010
www.bu.edu/tech/files/2010/10/VTK-Fall-2010.ppt

VTK - Readers

- **Image and Volume Readers**
  - vtkStructuredPointsReader - read VTK structured points data files
  - vtkSLCReader - read SLC structured points files
  - vtkTIFFReader - read files in TIFF format
  - vtkVolumeReader - read image (volume) files
  - vtkVolume16Reader - read 16-bit image (volume) files
- **Structured Grid Readers**
  - vtkStructuredGridReader - read VTK structured grid data files
  - vtkPLOT3DReader - read structured grid PLOT3D files
- **Rectilinear Grid Readers**
  - vtkRectilinearGridReader - read VTK rectilinear grid data files
- **Unstructured Grid Readers**
  - vtkUnstructuredGridReader - read VTK unstructured grid data files

Robert Putnam, Class Presentation at Boston University, 2010
www.bu.edu/tech/files/2010/10/VTK-Fall-2010.ppt

VTK - Readers

- **Polygonal Data Readers**
  - vtkPolyDataReader - read VTK polygonal data files
  - vtkBYUReader - read MOVIE.BYU files
  - vtkMCubesReader - read binary marching cubes files
  - vtkOBJReader - read Wavefront (Maya) .obj files
  - vtkPLYReader - read Stanford University PLY polygonal data files
  - vtkSTLReader - read stereo-lithography files
  - vtkUGFacetReader - read EDS Unigraphic facet files

- **Image and Volume Readers (add'l)**
  - vtkBMPReader - read PC bitmap files
  - vtkDEMReader - read digital elevation model files
  - vtkJPEGReader - read JPEG files
  - vtkImageReader - read various image files
  - vtkPNMReader - read PNM (ppm, pgm, pbm) files
  - vtkPNGRReader - read Portable Network Graphic files

Robert Putnam, Class Presentation at Boston University, 2010
www.bu.edu/tech/files/2010/10/VTK-Fall-2010.ppt

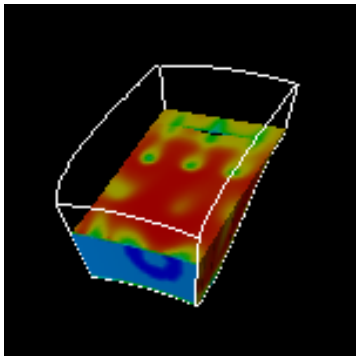
Clipping, Cutting, Subsampling

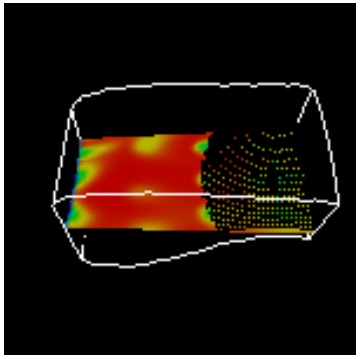
Selection Algorithms

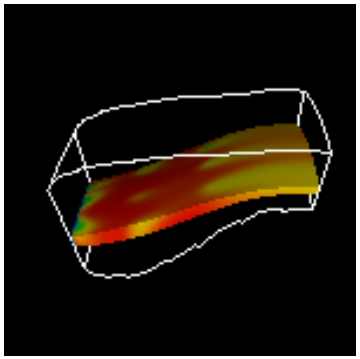
- Clipping
  - can reveal internal details of surface
  - VTK - vtkClipDataSet

- Cutting/Slicing
  - cutting through a dataset with a surface
  - VTK - vtkCutter

- Subsampling
  - reduces data size by selecting a subset of the original data
  - VTK - vtkExtractGrid







Robert Putnam, Class Presentation at Boston University, 2010
www.bu.edu/tech/files/2010/10/VTK-Fall-2010.ppt

Contouring

- **Scalar Algorithms (cont)**

  Contouring

  - construct a boundary between distinct regions, two steps:

    explore space to find points near contour

    connect points into contour (2D) or surface (3D)

  - 2D contour map (isoline):

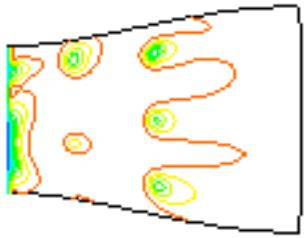
    applications: elevation contours from topography, pressure contours (weather maps) from meteorology3D isosurface:

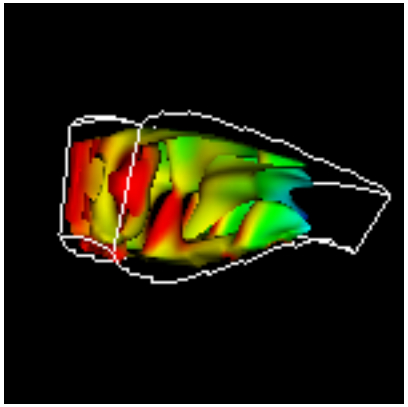
  - 3D isosurface:

    applications: tissue surfaces from tomography, constant pressure or temperature in fluid flow, implicit surfaces from math and CAD

  VTK

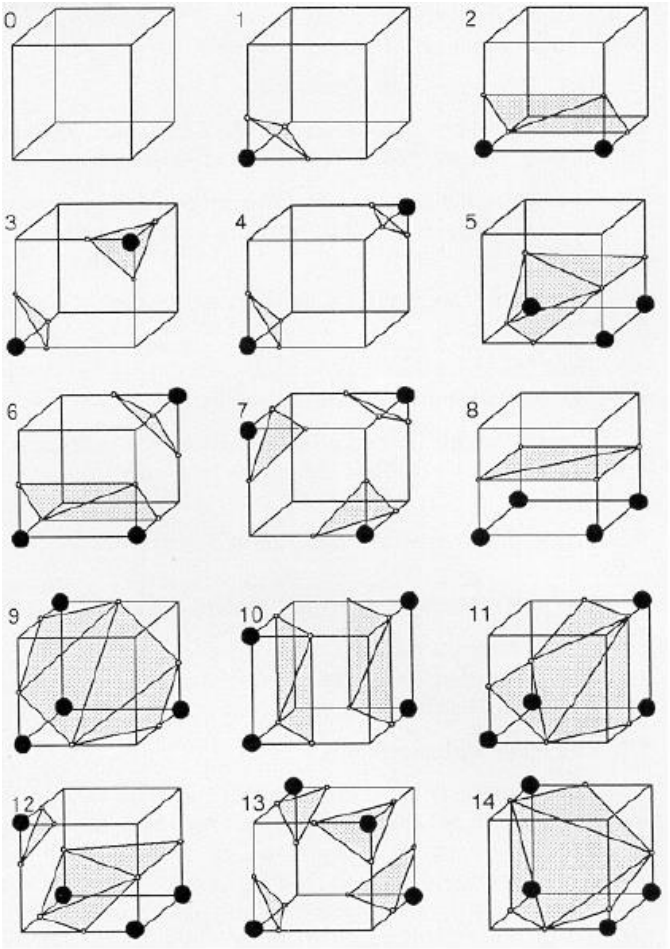
  vtkContourFilter

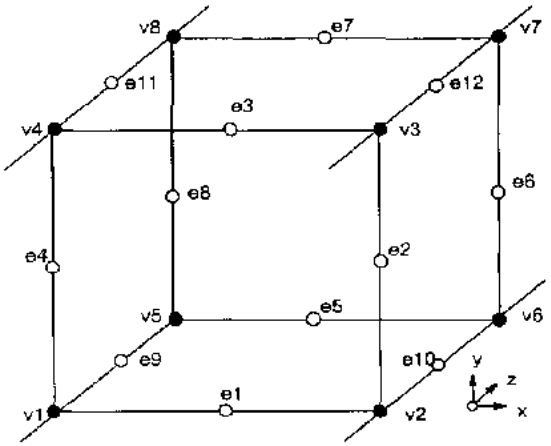




Robert Putnam, Class Presentation at Boston University, 2010
www.bu.edu/tech/files/2010/10/VTK-Fall-2010.ppt

Marching Cubes





Scott Schaefer, Class Presentation at Texas A&M:
http://faculty.cs.tamu.edu/schaefer/teaching/645_Fall2015/lectures/volumetric.ppt

Contour Trees

- Contour trees encode the topological changes that occur to the contour as the isovalue ranges between its minimum and maximum values
- Contour trees can be used to identify the most “important” isovalues in a data set according to various metrics (e.g., persistence / prominence)

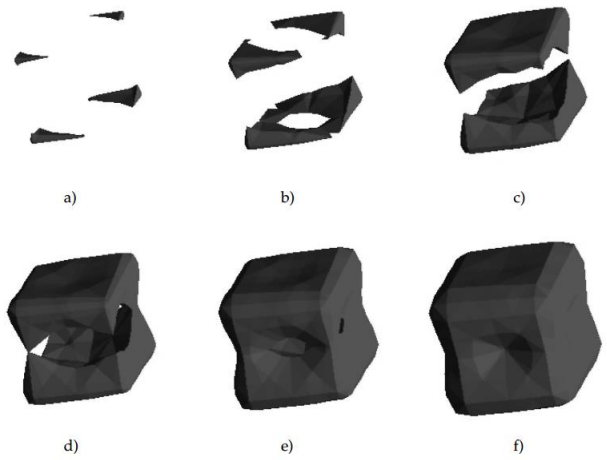


Figure 1: Level Sets of f as $f(x)$ decreases

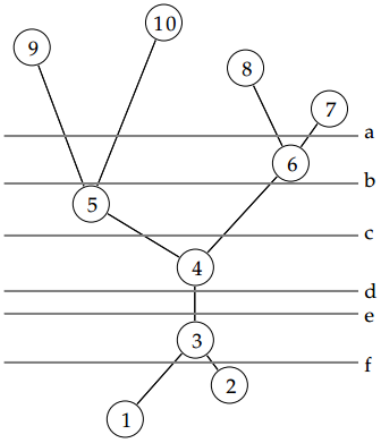


Figure 2: Contour tree for Figure 1

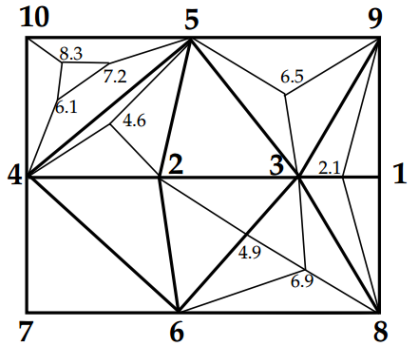
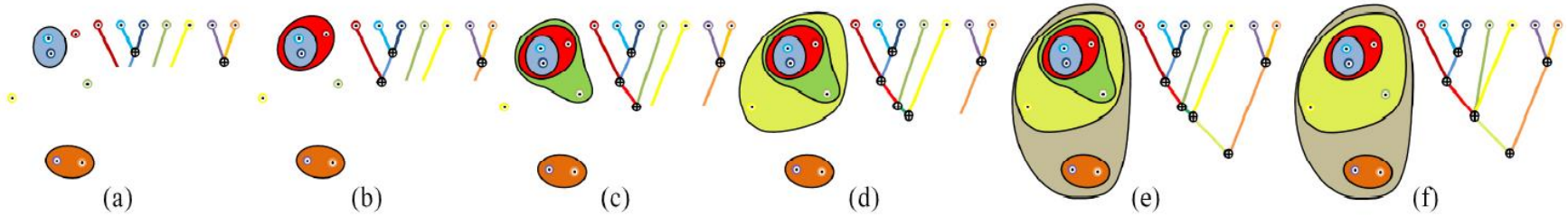


Figure 3: A small 2-D example, with the same contour tree as Fig. 1

From “Computing Contour Trees in All Dimensions” Hamish Carr, Jack Snoeyink, and Ulrike Axen. Computational Geometry: Theory and Applications, 2003.

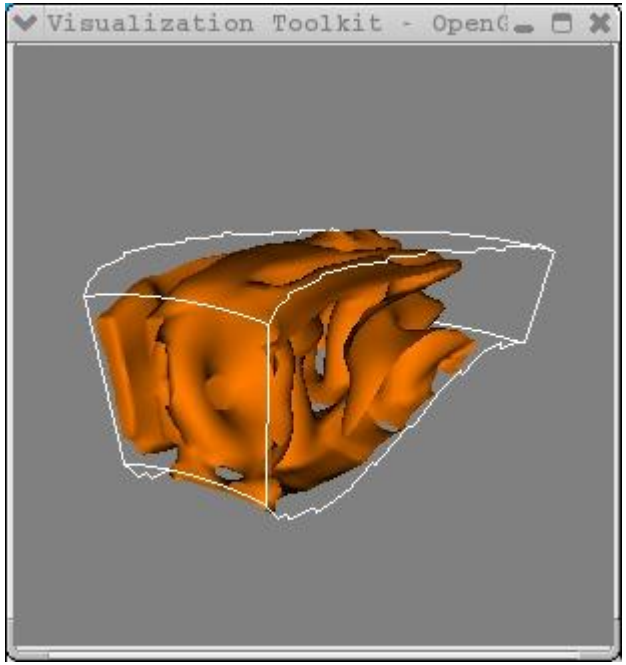


W. Widanagamaachchi, P.-T. Bremer, C. Sewell, L.-T. Lo, J. Ahrens, and V. Pascucci. Data-Parallel Halo Finding with Variable Linking Lengths. Proc. of the IEEE Symposium on Large-Scale Data Analysis and Visualization, Nov. 2014.

Code – Contour (isosurface)

Editor: isosurface.tcl

```
. . .
vtkStructuredGridReader reader
   reader SetFileName “density.vtk”
   reader Update

vtkContourFilter iso
   iso SetInputConnection [reader GetOutputPort]
   iso SetValue 0 0.26

vtkPolyDataMapper isoMapper
   isoMapper SetInputConnection [iso GetOutputPort]
   eval isoMapper SetScalarRange [[reader GetOutput] GetScalarRange]

vtkActor isoActor
   isoActor SetMapper isoMapper
. . .
```



Robert Putnam, Class Presentation at Boston University, 2010
www.bu.edu/tech/files/2010/10/VTK-Fall-2010.ppt

Oriented Glyphs

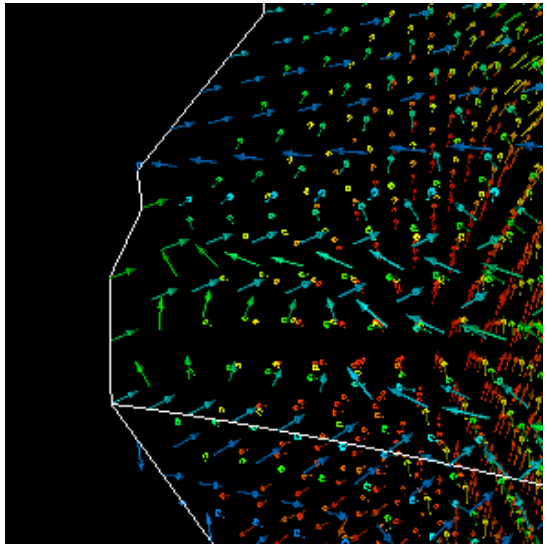
- **Vector Algorithms (cont)**

  Oriented Glyphs

  - orientation indicates direction
  - scale indicates magnitude
  - color indicates magnitude, pressure, temperature, or any variable

  VTK

  - vtkGlyph3D

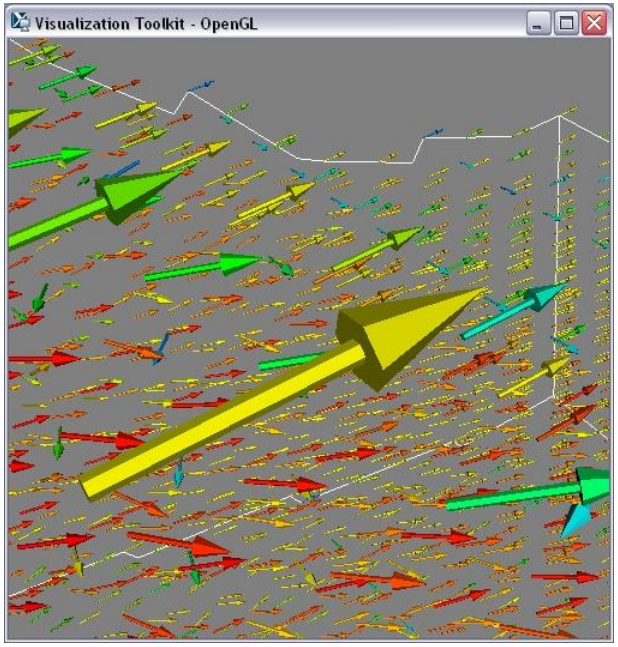


Robert Putnam, Class Presentation at Boston University, 2010
www.bu.edu/tech/files/2010/10/VTK-Fall-2010.ppt

Code – Oriented Glyphs

Editor: glyph.tcl

```
vtkArrowSource arrow
   arrow SetTipResolution 6
   arrow SetTipRadius 0.1
   arrow SetTipLength 0.35
   arrow SetShaftResolution 6
   arrow SetShaftRadius 0.03

vtkGlyph3D glyph
   glyph SetInput [reader GetOutputPort]
   glyph SetSource [arrow GetOutputPort]
   glyph SetVectorModeToUseVector
   glyph SetColorModeToColorByScalar
   glyph SetScaleModeToDataScalingOff
   glyph OrientOn
   glyph SetScaleFactor 0.2

vtkPolyDataMapper glyphMapper
   glyphMapper SetInput [glyph GetOutput]
   glyphMapper SetLookupTable lut
   glyphMapper ScalarVisibilityOn
   eval glyphMapper SetScalarRange [[reader GetOutput] GetScalarRange]
```



Robert Putnam, Class Presentation at Boston University, 2010
www.bu.edu/tech/files/2010/10/VTK-Fall-2010.ppt

Field Lines

- **Vector Algorithms (cont)**

  Field Lines

  - Fluid flow is described by a vector field in three dimensions for steady (fixed time) flows or four dimensions for unsteady (time varying) flows
  - Three techniques for determining flow

    Pathline (Trace)

    tracks particle through unsteady (time-varying) flow

    shows particle trajectories over time

    rake releases particles from multiple positions at the same time instant

    reveals compression, vorticity

    Streamline

    - tracks particle through steady (fixed-time) flow
    - holds flow steady at a fixed time
    - snapshot of flow at a given time instant

    Streakline

    - particles released from the same position over a time interval (time-varying)
    - snapshot of the variation of flow over time
    - example: dye steadily injected into fluid at a fixed point

Robert Putnam, Class Presentation at Boston University, 2010
www.bu.edu/tech/files/2010/10/VTK-Fall-2010.ppt

Field Lines

Streamlines

- Lines show particle flow
- VTK – vtkStreamTracer

Streamlets

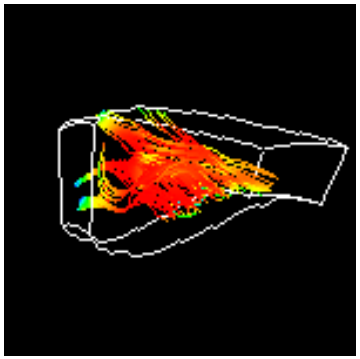
- half way between streamlines and glyphs
- VTK - vtkStreamTracer, vtkGlyph3D

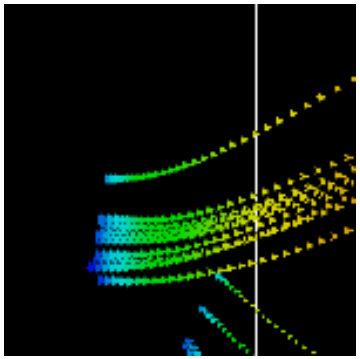
Streamribbon

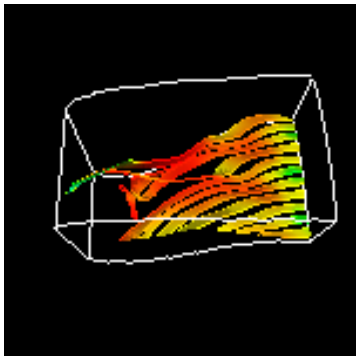
- rake of two particles to create a ribbon
- VTK - vtkStreamTracer, vtkRuledSurfaceFilter

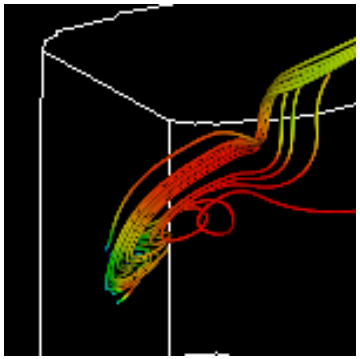
Streamtube

- circular rake of particles to create a tube
- VTK - vtkStreamTracer, vtkTubeFilter







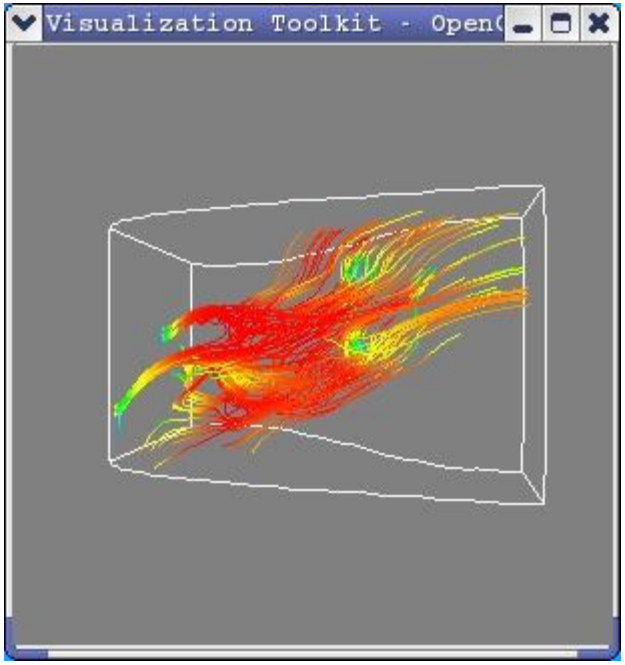


Robert Putnam, Class Presentation at Boston University, 2010
www.bu.edu/tech/files/2010/10/VTK-Fall-2010.ppt

Code – Streamlines

Editor: streamLines.tcl

```
vtkPointSource seeds
  seeds SetRadius 3.0
  eval seeds SetCenter [[reader GetOutput] GetCenter]
  seeds SetNumberOfPoints 100

vtkRungeKutta4 integ

vtkStreamTracer streamer
  streamer SetInputConnection  [reader GetOutputPort]
  streamer SetSourceConnection [seeds GetOutputPort]
  streamer SetMaximumPropagation 100
  streamer SetMaximumPropagationUnitToTimeUnit
  streamer SetInitialIntegrationStepUnitToCellLengthUnit
  streamer SetInitialIntegrationStep 0.1
  streamer SetIntegrationDirectionToBoth
  streamer SetIntegrator integ

vtkPolyDataMapper mapStreamLines
  mapStreamLines SetInputConnection [streamer GetOutputPort]
  eval mapStreamLines SetScalarRange [[reader GetOutput] GetScalarRange]

vtkActor streamLineActor
  streamLineActor SetMapper mapStreamLines
```

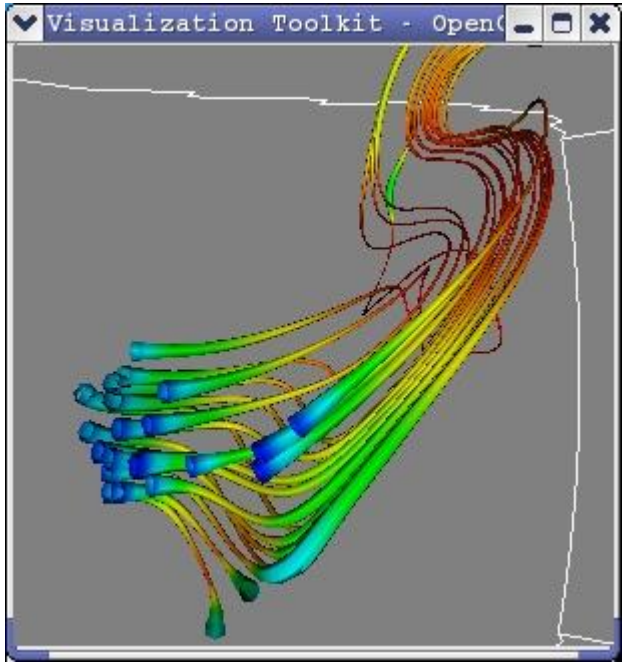


Robert Putnam, Class Presentation at Boston University, 2010
www.bu.edu/tech/files/2010/10/VTK-Fall-2010.ppt

Code – Streamtubes

Editor: streamTubes.varyRadius.tcl

```
vtkPointSource seeds
    seeds SetRadius 1.0
    seeds SetNumberOfPoints 50

vtkRungeKutta4 integ
vtkStreamTracer streamer
   streamer SetInputConnection  [reader GetOutputPort]
   streamer SetSourceConnection [seeds GetOutputPort]
. . .
vtkTubeFilter streamTube
    streamTube SetInputConnection [streamer GetOutputPort]
    streamTube SetRadius 0.01
    streamTube SetNumberOfSides 6
#    streamTube SetVaryRadiusToVaryRadiusOff
    streamTube SetVaryRadiusToVaryRadiusByScalar

vtkPolyDataMapper mapStreamTube
    mapStreamTube SetInputConnection [streamTube GetOutputPort]
    mapStreamTube SetLookupTable lut
    eval mapStreamTube SetScalarRange  [[[[reader GetOutput] GetPointData] GetScalars] GetRange]

vtkActor streamTubeActor
    streamTubeActor SetMapper mapStreamTube
    [streamTubeActor GetProperty] BackfaceCullingOn
```



Robert Putnam, Class Presentation at Boston University, 2010
www.bu.edu/tech/files/2010/10/VTK-Fall-2010.ppt

VTK - Writers

- **Polygonal Data Writers**
  - vtkBYUWriter - write MOVIE.BYU files
  - vtkCGMWriter - write 2D polygonal data as a CGM file
  - vtkIVWriter - write Inventor files
  - vtkMCubesWriter - write triangles in marching cubes format
  - vtkPolyDataWriter - write VTK polygonal data files
  - vtkPLYWriter - write Standford University PLY polygonal data files
  - vtkSTLWriter - write stereo-lithography files

- **Image and Volume writers**
  - vtkBMPwriter - write PC bitmap files
  - vtkJPEGwriter - write images in JPEG format
  - vtkPostscriptWriter – write image files in Postscript format
  - vtkPNMwriter - write PNM (ppm, pgm, pbm) image files
  - vtkPNGwriter - write image file in Portable Network Graphic format
  - vtkTIFFWriter – write image files in TIFF format
  - vtkStructuredPointsWriter – write a vtkStructuredPoints file

Robert Putnam, Class Presentation at Boston University, 2010
www.bu.edu/tech/files/2010/10/VTK-Fall-2010.ppt

ParaView

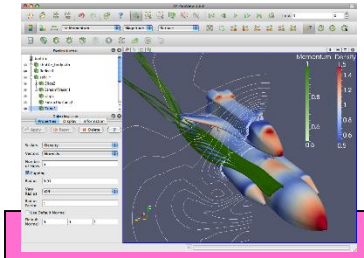
Large-scale Scientific Visualization

What is ParaView?

- An open-source, scalable, multi-platform visualization application.
- Support for distributed computation models to process large data sets.
- An open, flexible, and intuitive user interface.
- An extensible, modular architecture based on open standards.
- A flexible BSD 3 Clause license
- Commercial maintenance and support.

Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

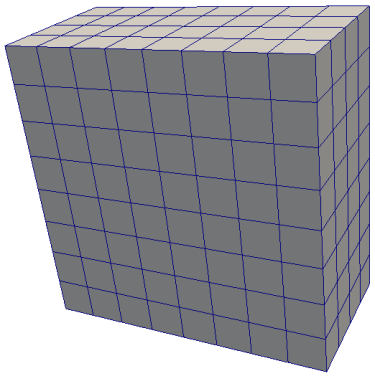
ParaView Application Architecture



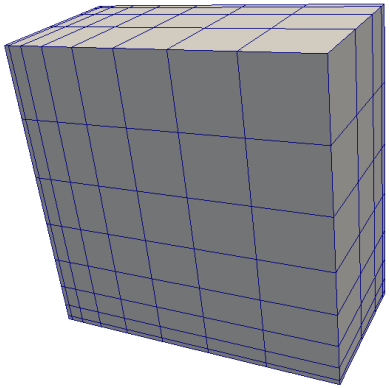
ParaView Client	pvpython	ParaWeb	Catalyst	Custom App
UI (Qt Widgets, Python Wrappings)				
ParaView Server				
VTK				
OpenGL	MPI	IceT	Etc.	

Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

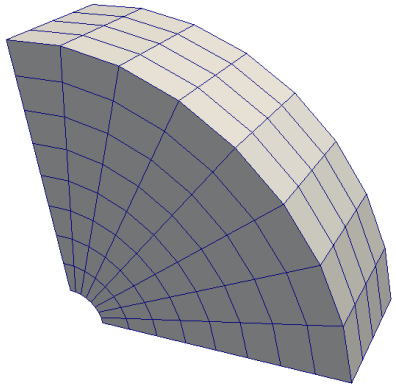
Data Types



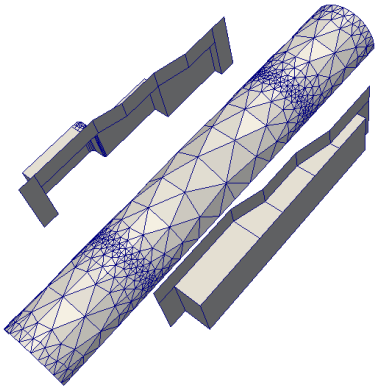
Uniform Rectilinear
(vtkImageData)



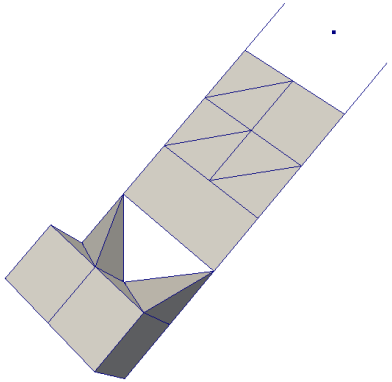
Non-Uniform Rectilinear
(vtkRectilinearData)



Curvilinear
(vtkStructuredData)



Polygonal
(vtkPolyData)



Unstructured Grid
(vtkUnstructuredGrid)

Multi-block

Hierarchical Adaptive Mesh Refinement (AMR)

Hierarchical Uniform AMR

Octree

Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

User Interface

Menu Bar

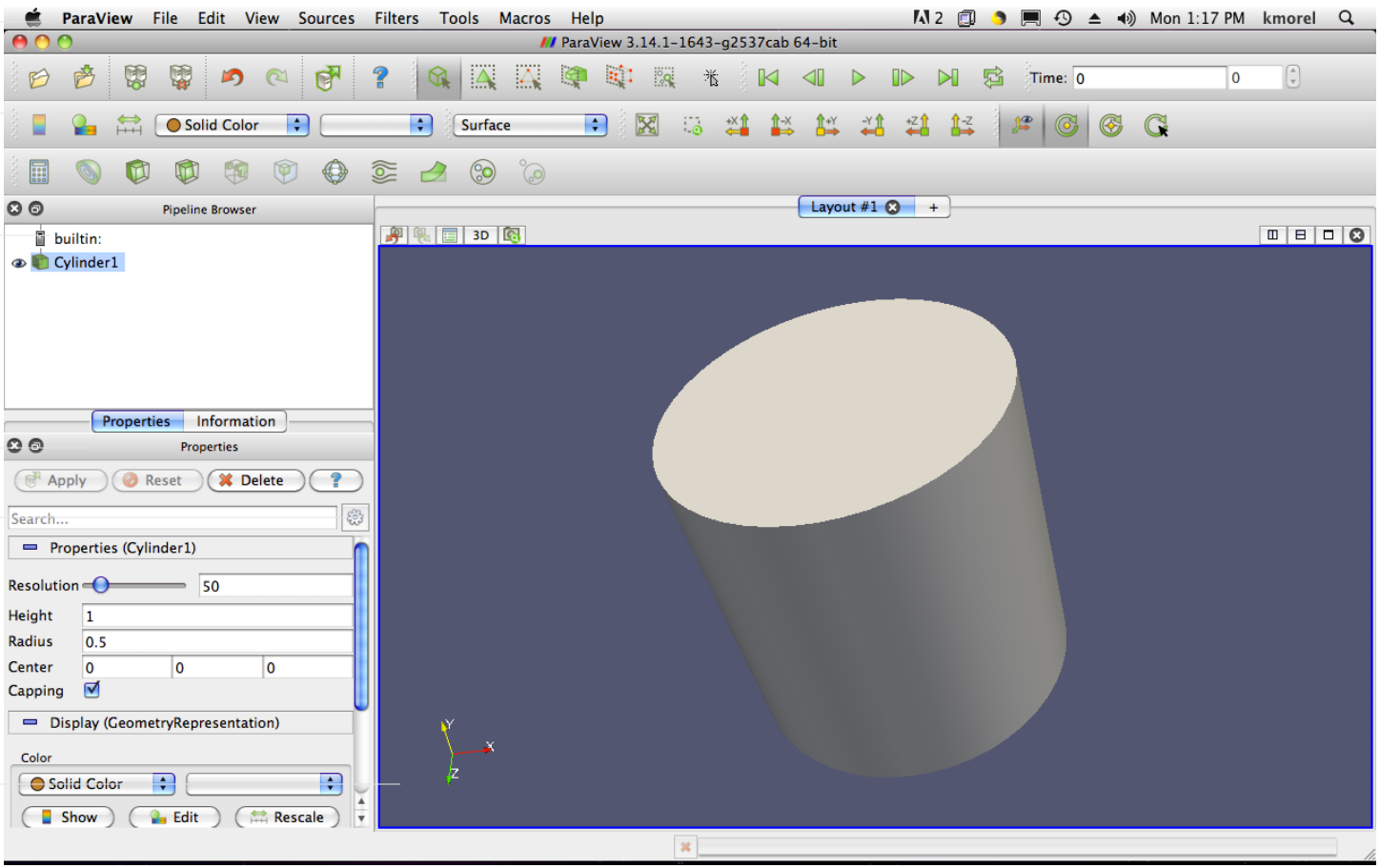
Toolbars

Pipeline Browser

Properties Panel

Advanced Toggle

3D View

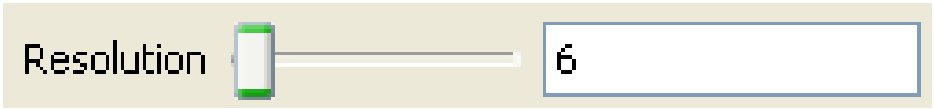


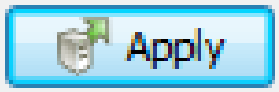
Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

Creating a Cylinder Source

1. Go to the **Source** menu and select **Cylinder**.
2. Click the **Apply** button to accept the default parameters. 

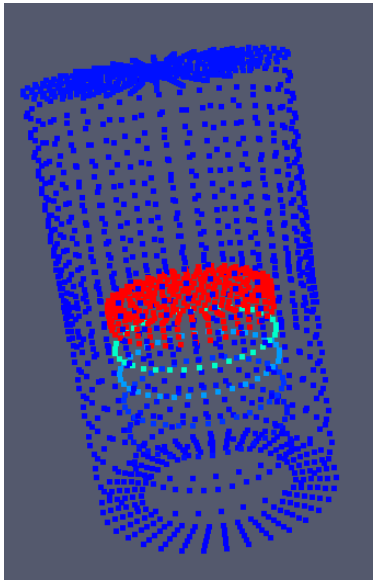
3. Increase the **Resolution** parameter.



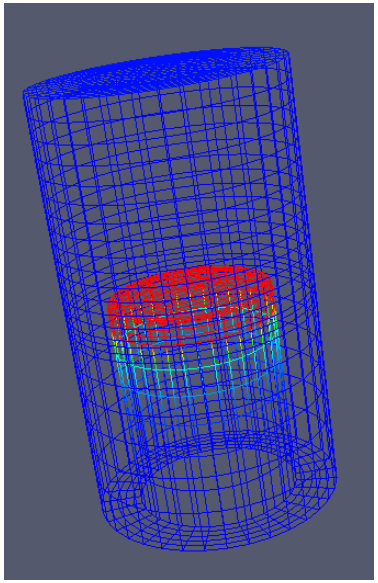
4. Click the 

button again.
5. Delete the Cylinder. 

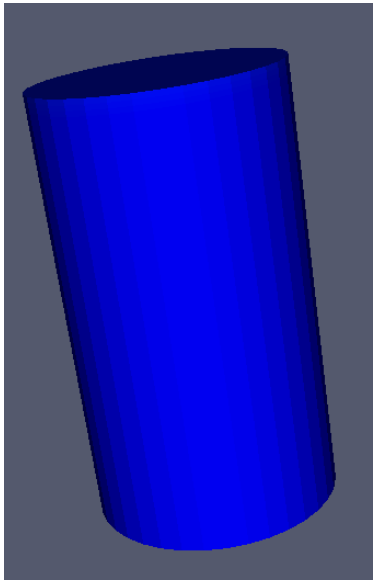
Geometry Representations



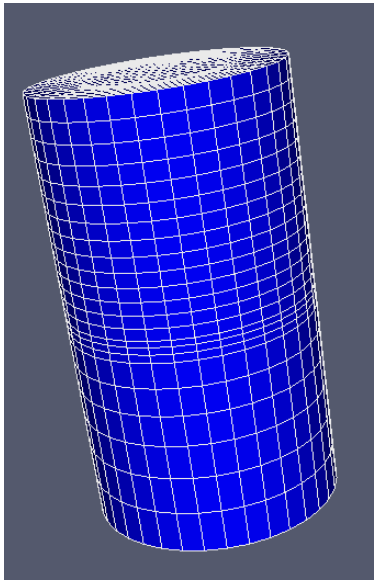
Points



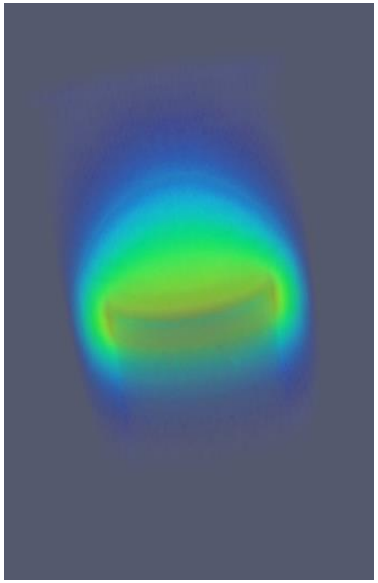
Wireframe



Surface



Surface
with Edges



Volume

Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

Common Filters


Calculator


Contour


Clip


Slice


Threshold


Extract Subset


Glyph


Stream Tracer


Warp (vector)

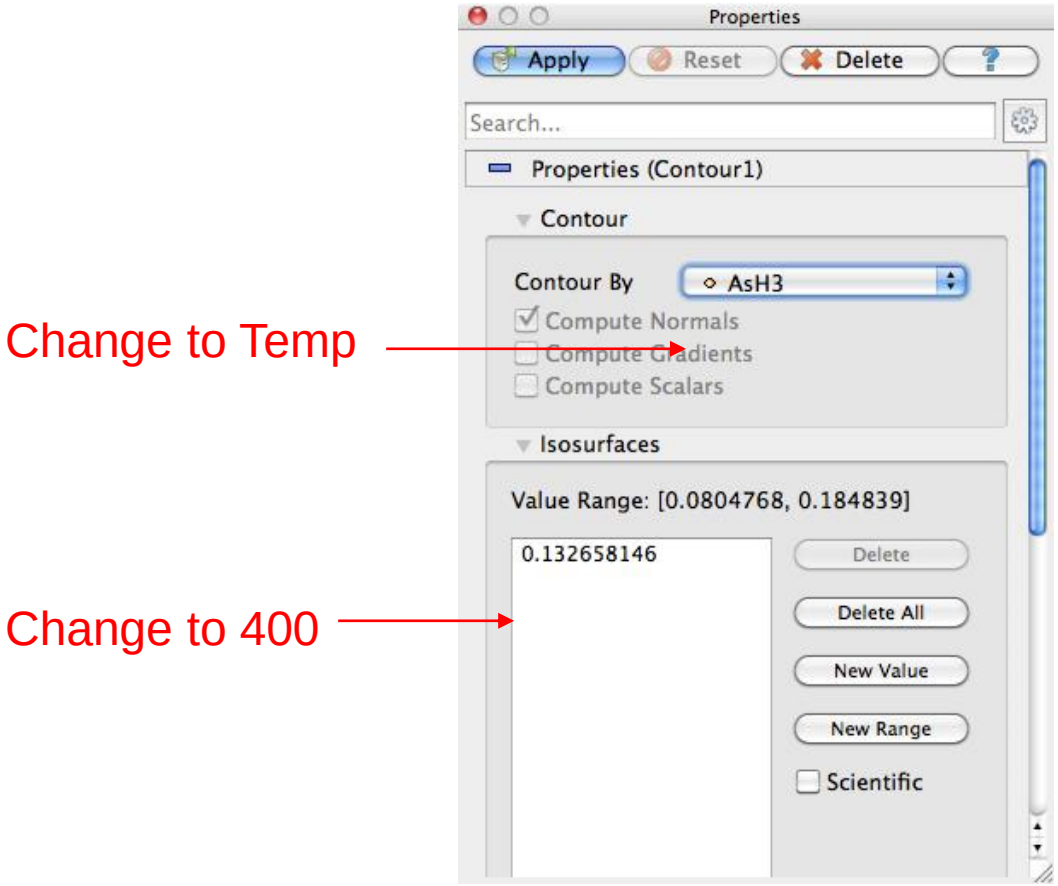

Group Datasets


Extract Level

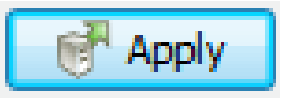
Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

Apply a Filter

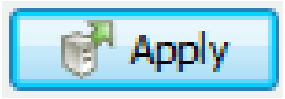
Change parameters to create an isosurface at Temp = 400K.



Histogram / Bar Chart

1. Select disk_out_ref.ex2.
2. Filters → Data Analysis → Histogram 
3. Change Input Array to Temp.
4. 

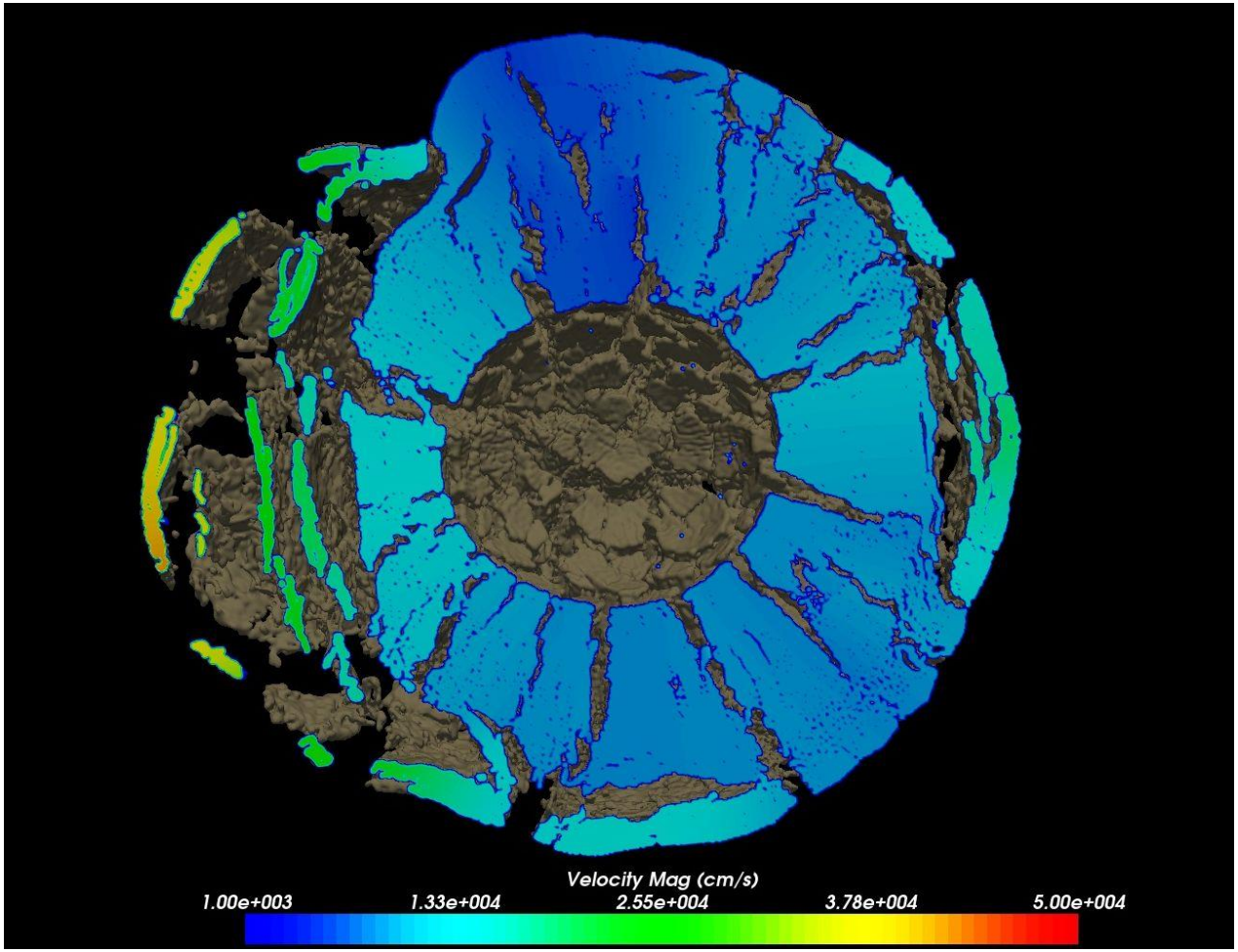
Make an Animation

1. Sources → Sphere, 

2. Make animation view visible.
3. Change No. Frames to 50.
4. Select Sphere1, Start Theta, press 
5. Double-click Sphere1 – Start Theta
6. Make New keyframe.
7. First keyframe value→360, second keyframe time→0.5 value→0.
8. Click OK.

Golevka Asteroid vs. 10 Megaton Explosion

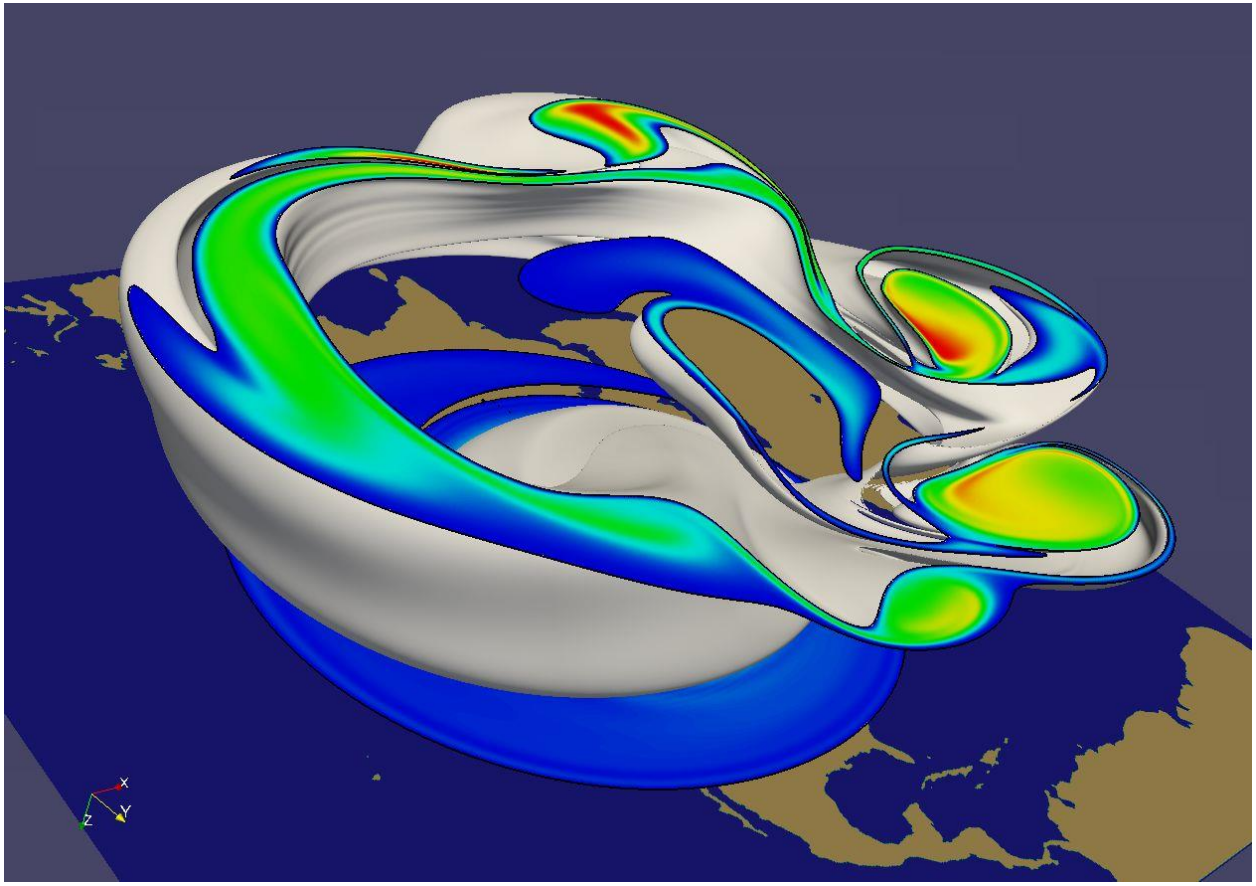
- CTH shock physics, over 1 billion cells



Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

Polar Vortex Breakdown

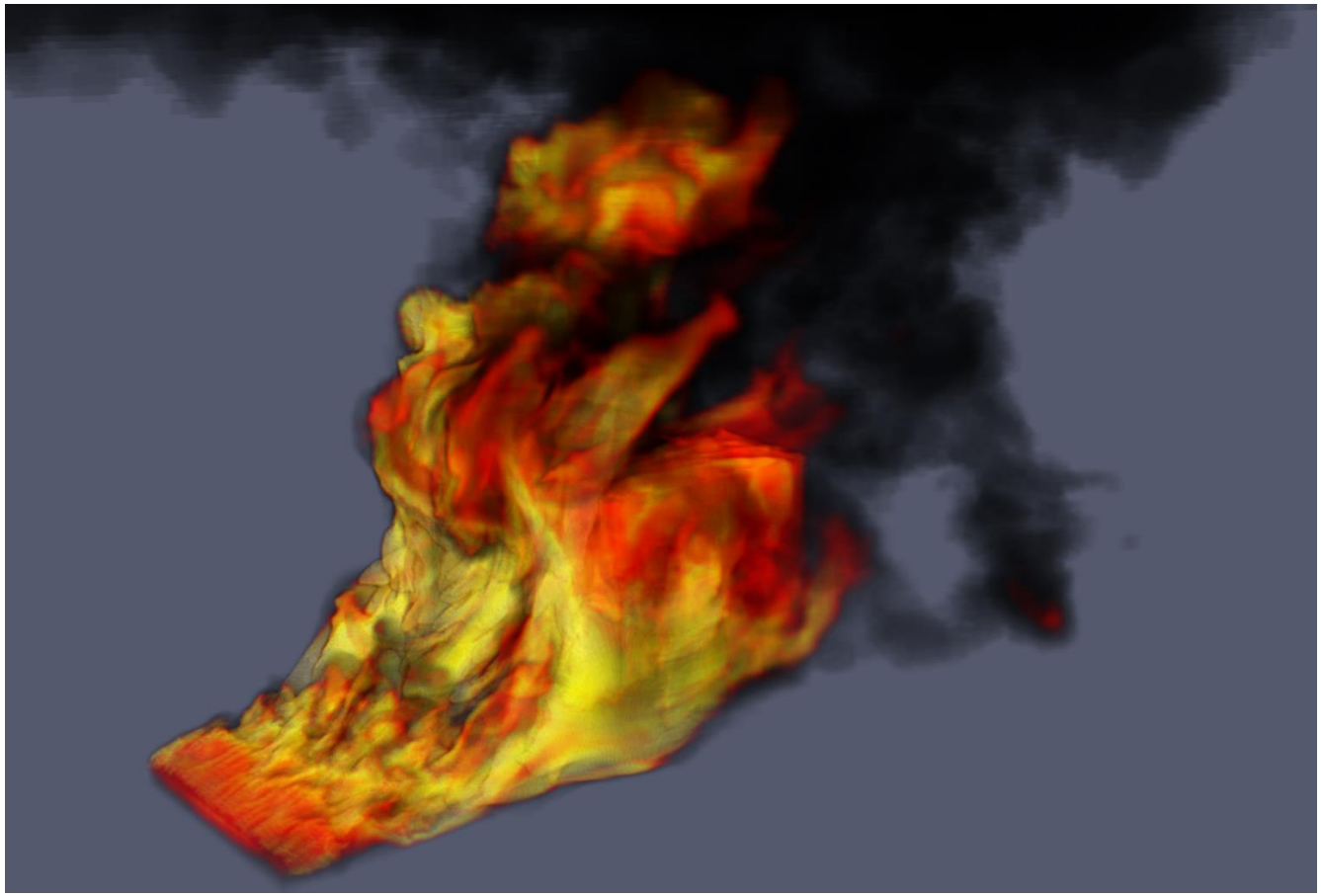
- SEAM Climate Modeling, 1 billion cells (500 million cells visualized).



Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

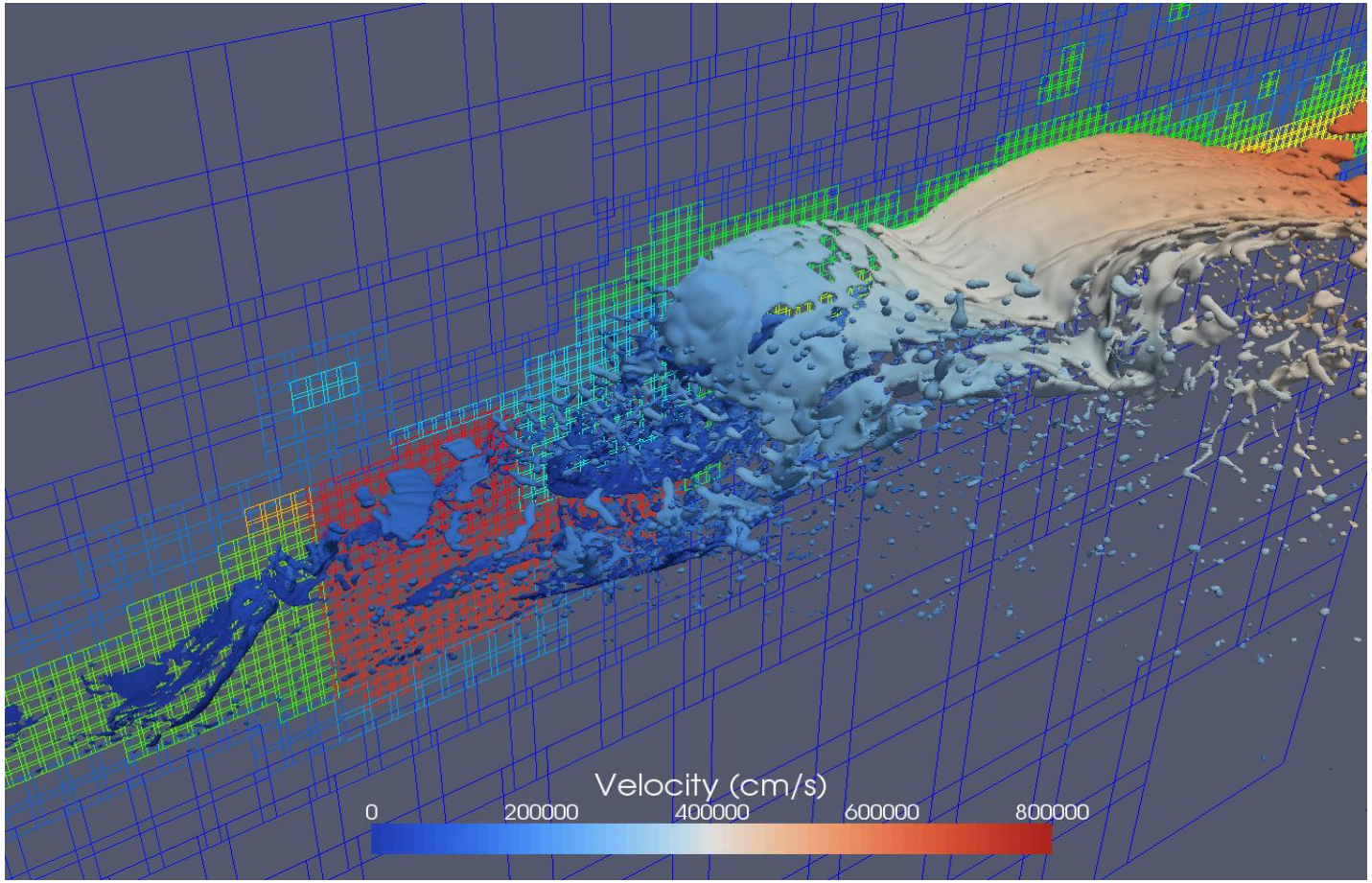
Objects-in-Crosswind Fire

- Coupled SIERRA/Fuego/Syrinx/Calore, 10 million unstructured hexahedra



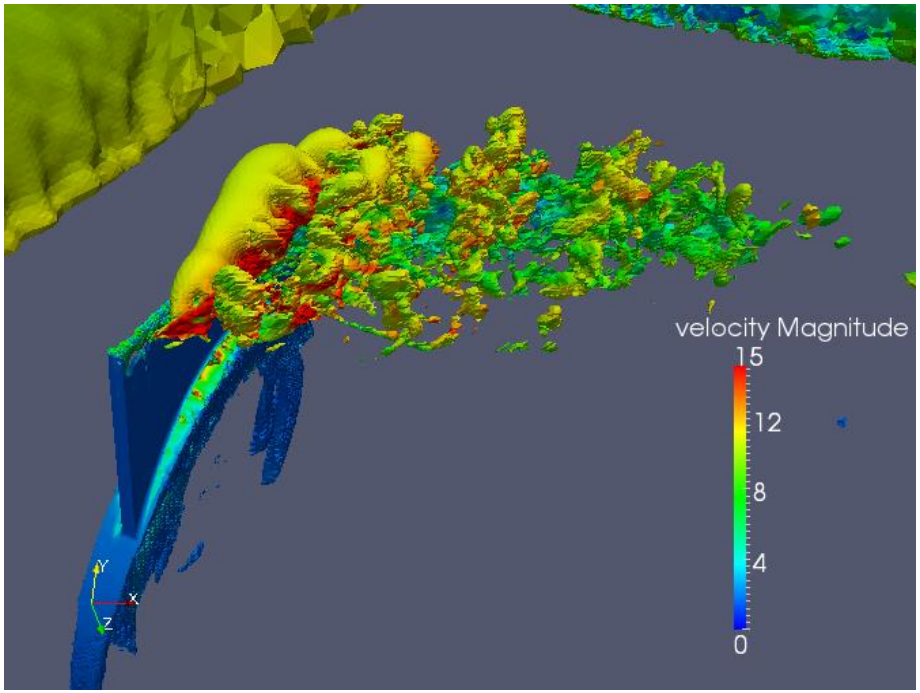
Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

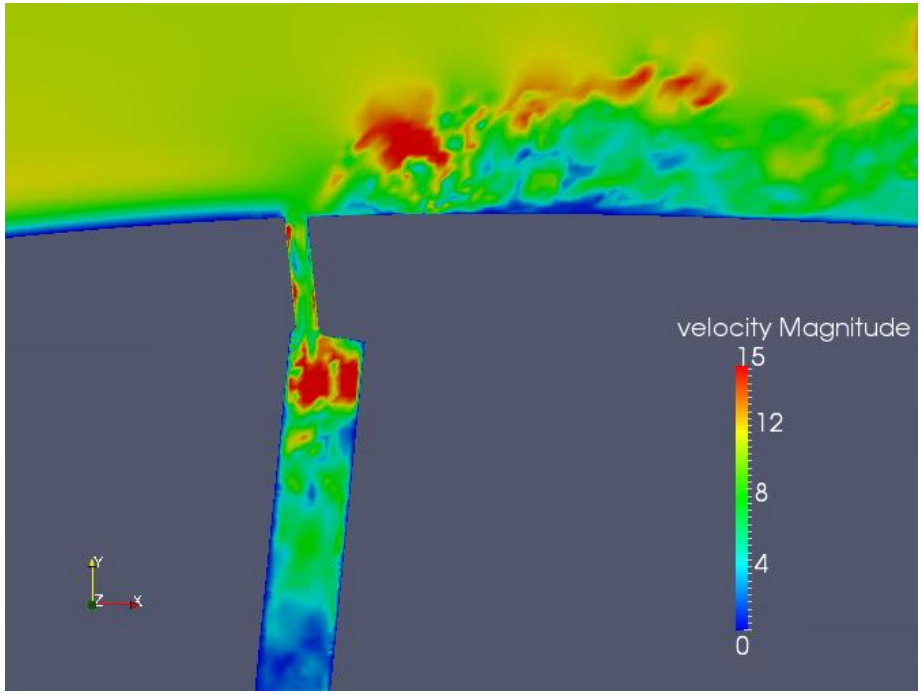
Large Scale AMR

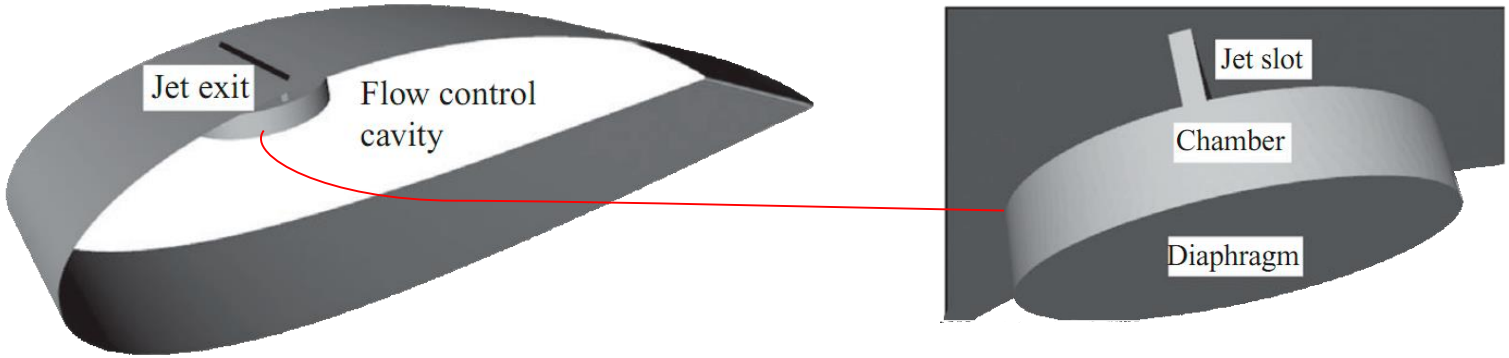


Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

Wing with Unsteady Crossflow from Synthetic Jet (3.3B tet)

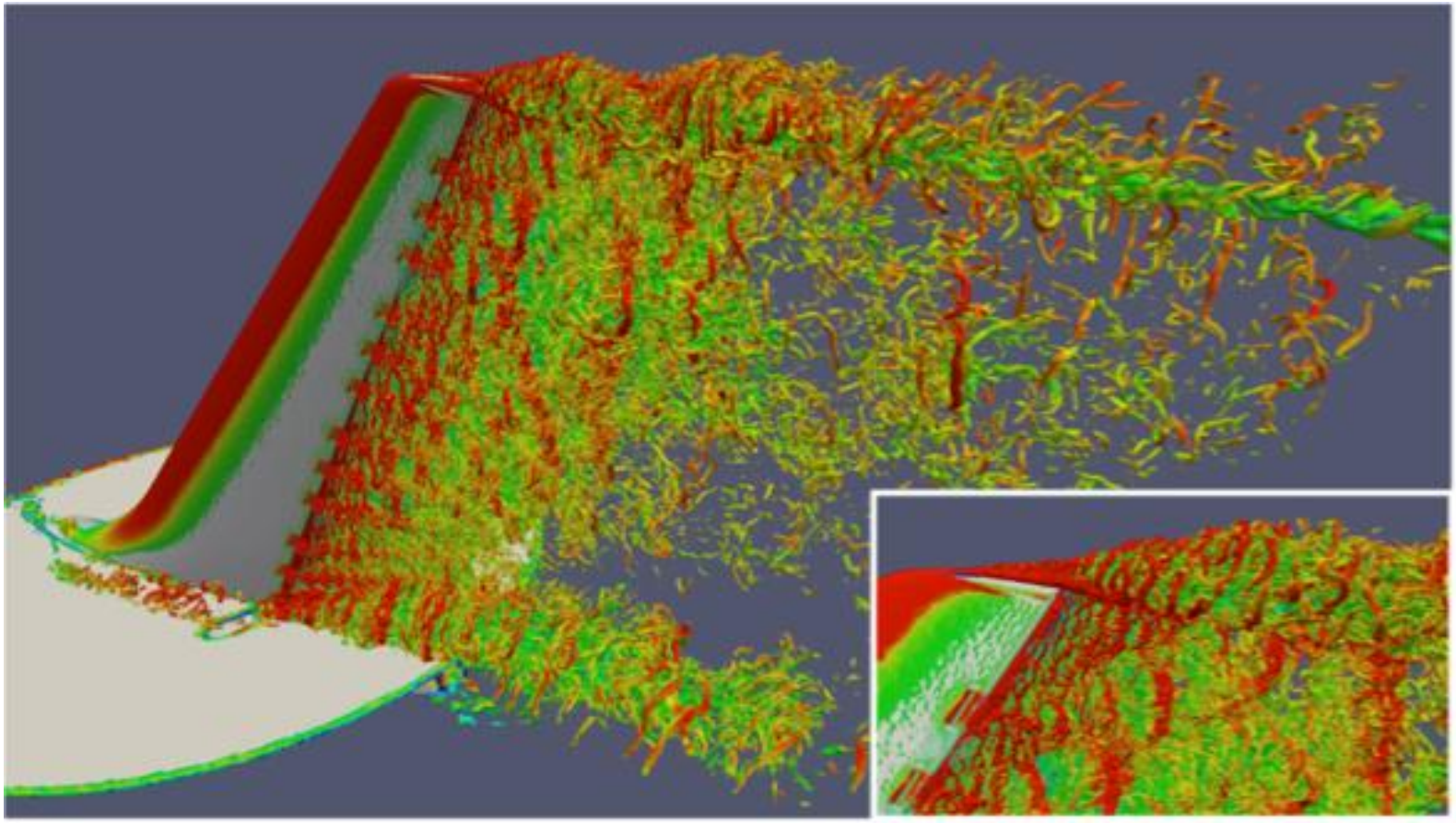






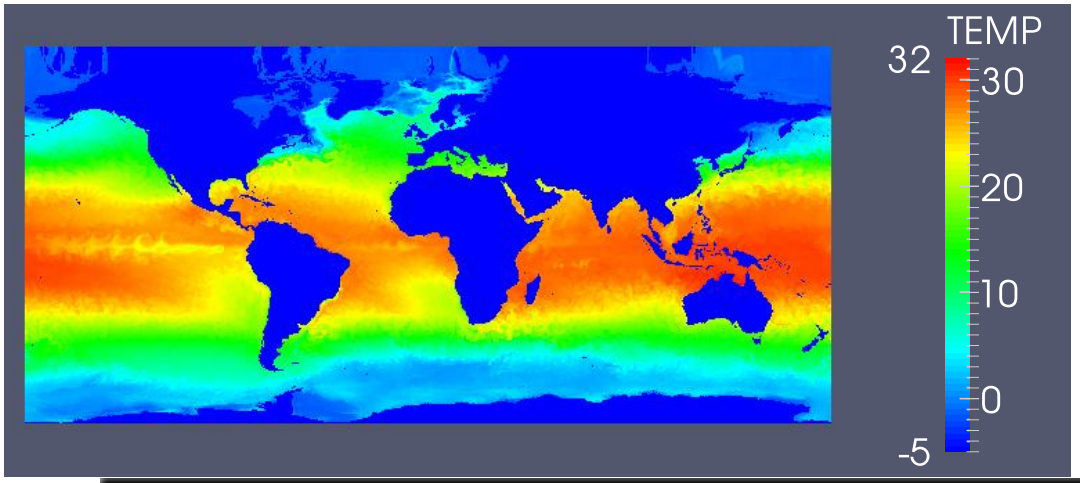
Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

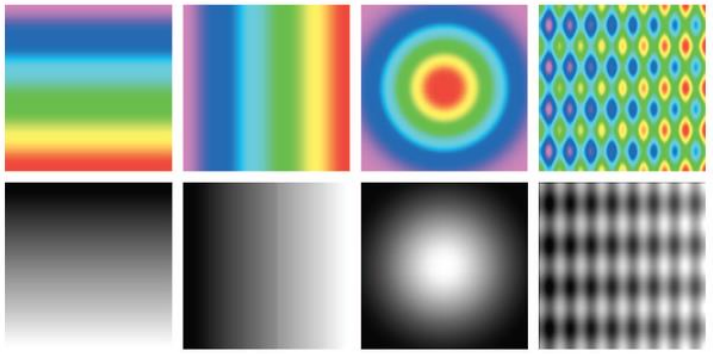
Wake of Deflected Wing Flap



Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

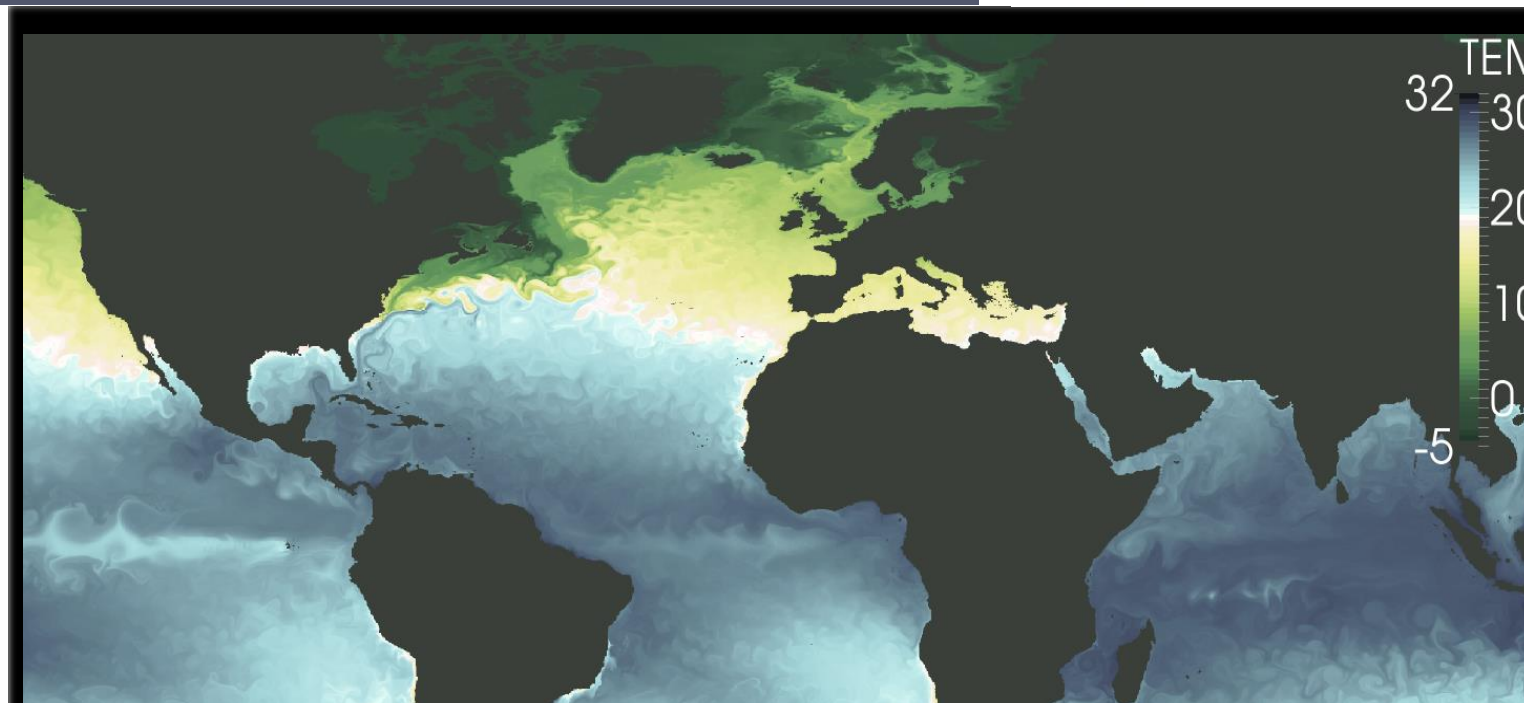
Choosing Color Maps





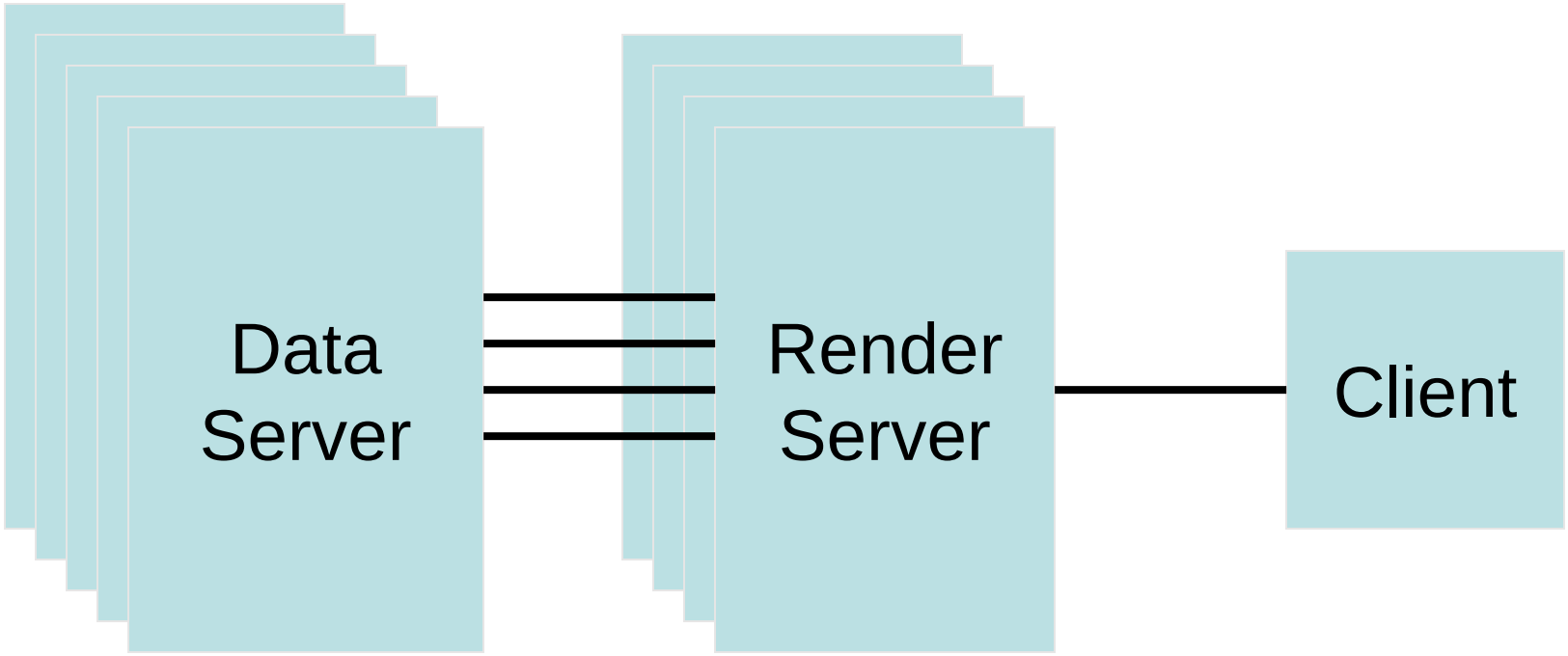
Apparent sharp gradients in the data are revealed as rainbow color map artifacts, not data features

Borland D, Taylor III RM (2007) Rainbow color map (still) considered harmful. IEEE Computer Graphics and Applications March/April:14-17



Ocean images from “Understanding via Color”, 2015, by Francesca Samsel

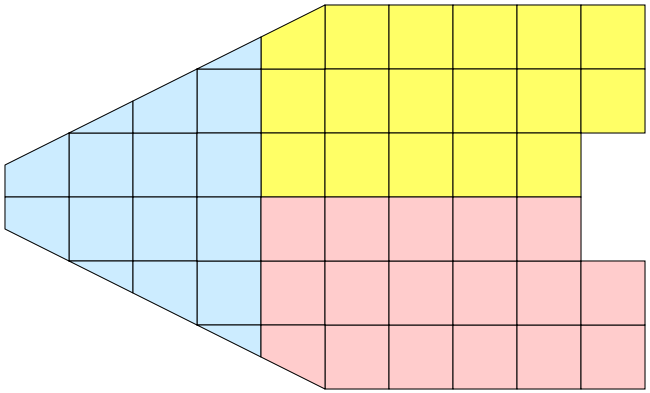
Client-Render Server-Data Server



Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

Data Parallel Pipelines

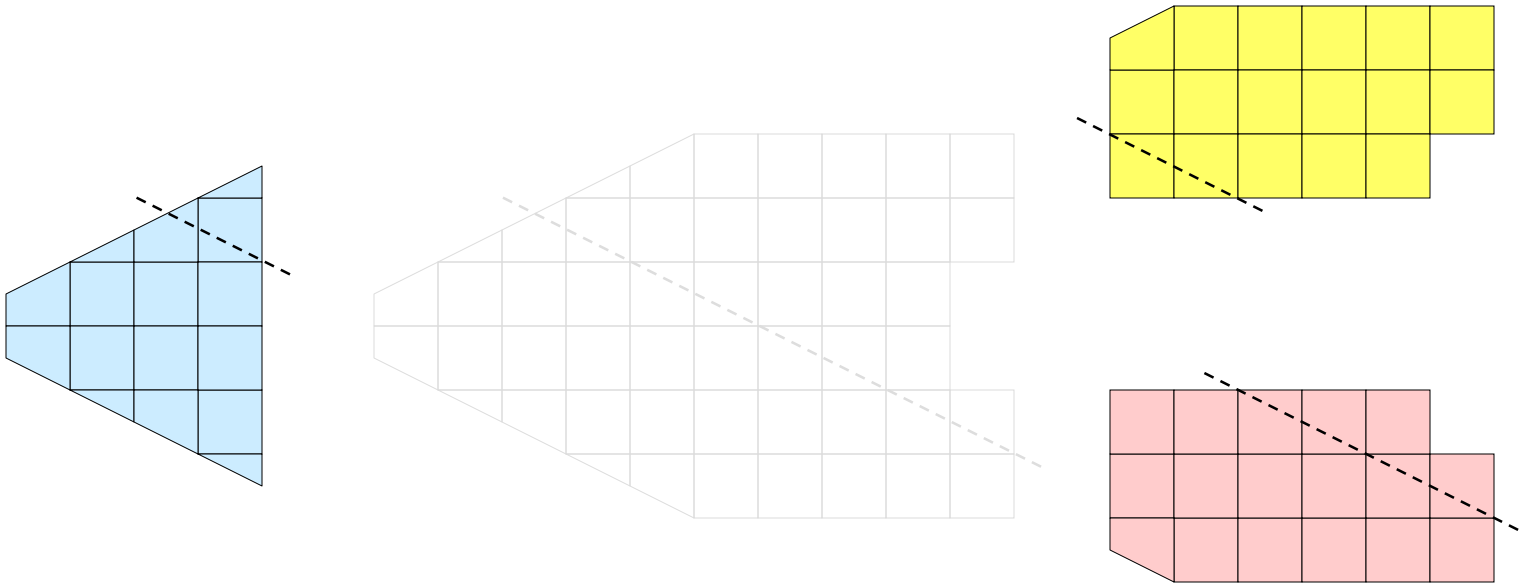
- Duplicate pipelines run independently on different partitions of data.



Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

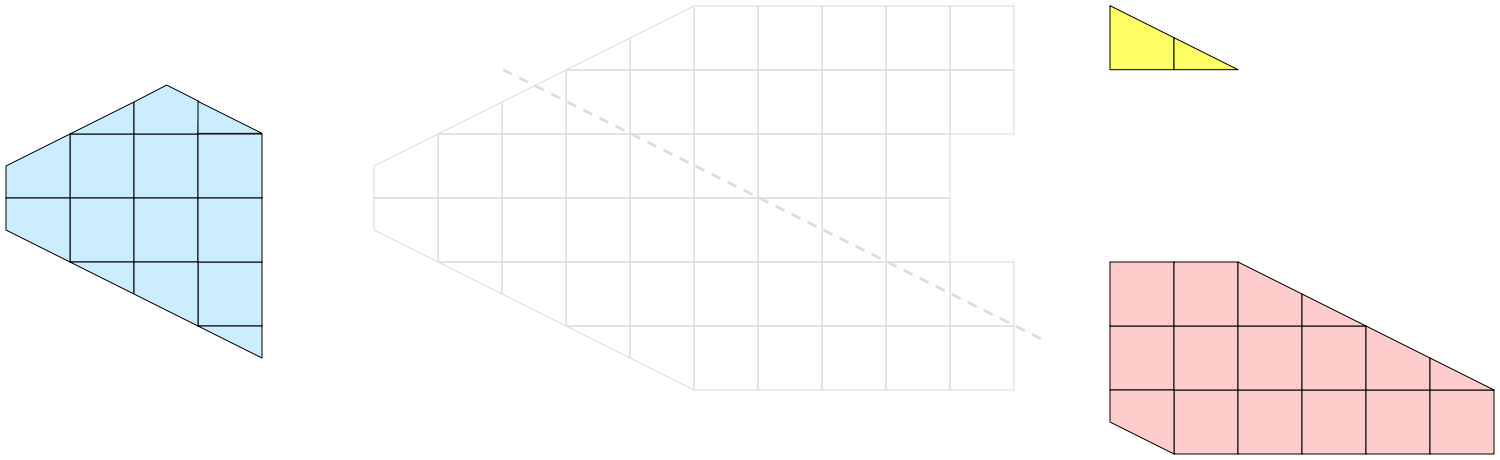
Data Parallel Pipelines

- Some operations will work regardless.
  - Example: Clipping.



Data Parallel Pipelines

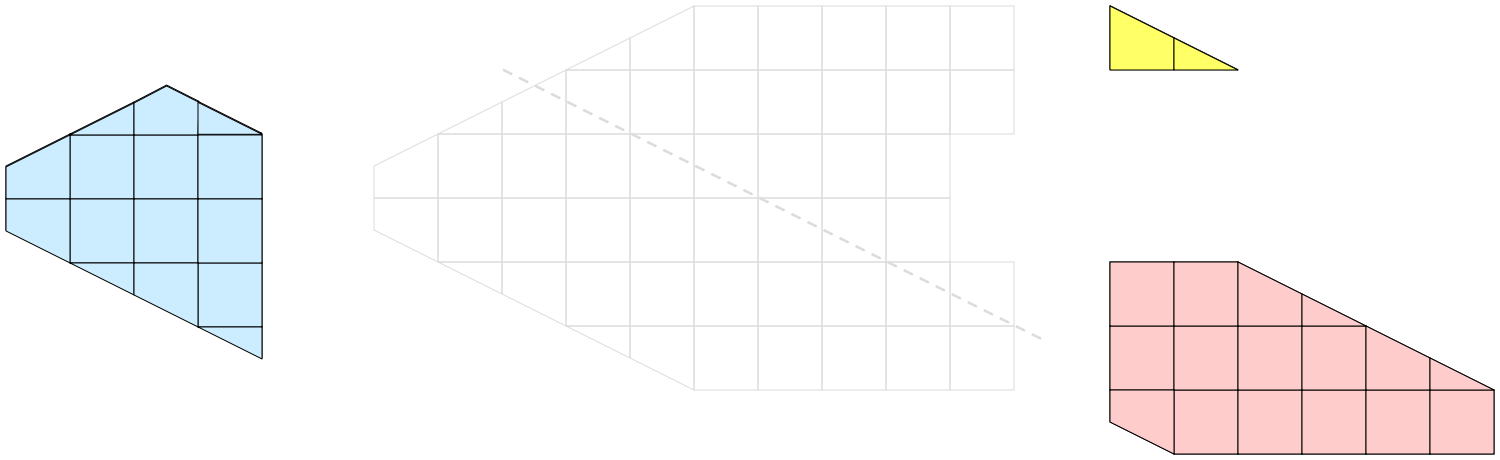
- Some operations will work regardless.
  - Example: Clipping.



Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

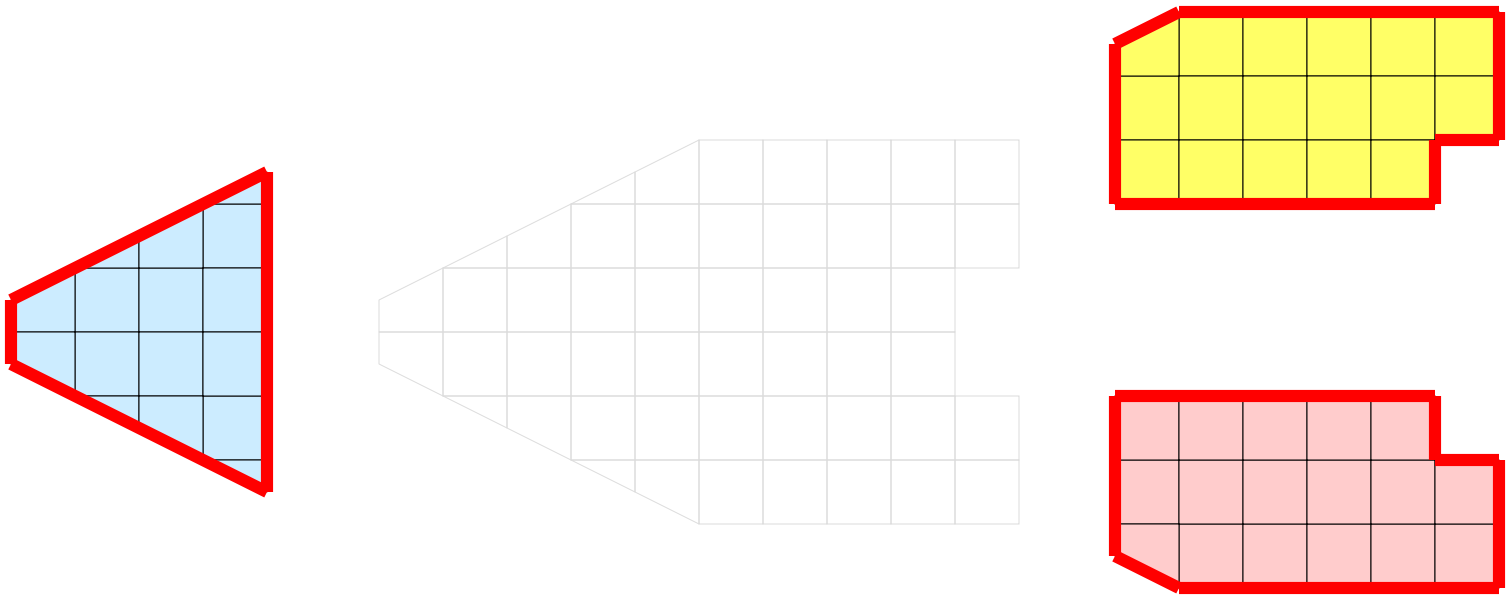
Data Parallel Pipelines

- Some operations will work regardless.
  - Example: Clipping.



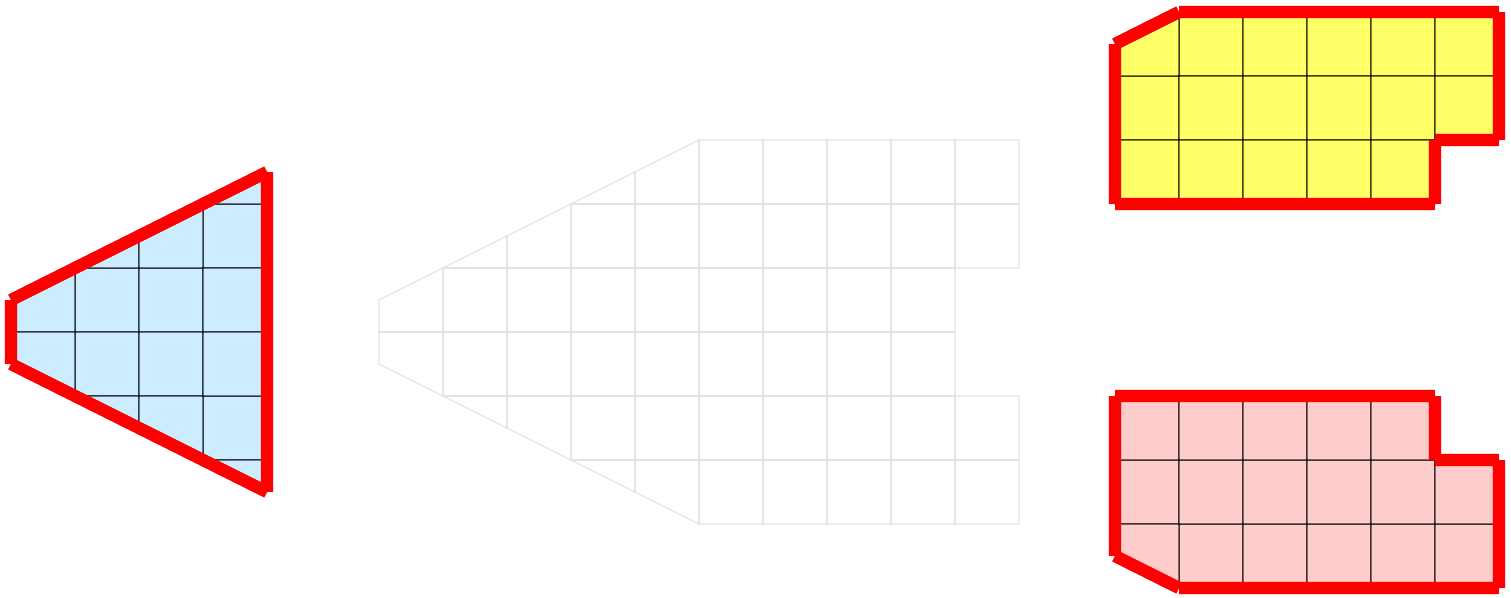
Data Parallel Pipelines

- Some operations will have problems.
  - Example: External Faces



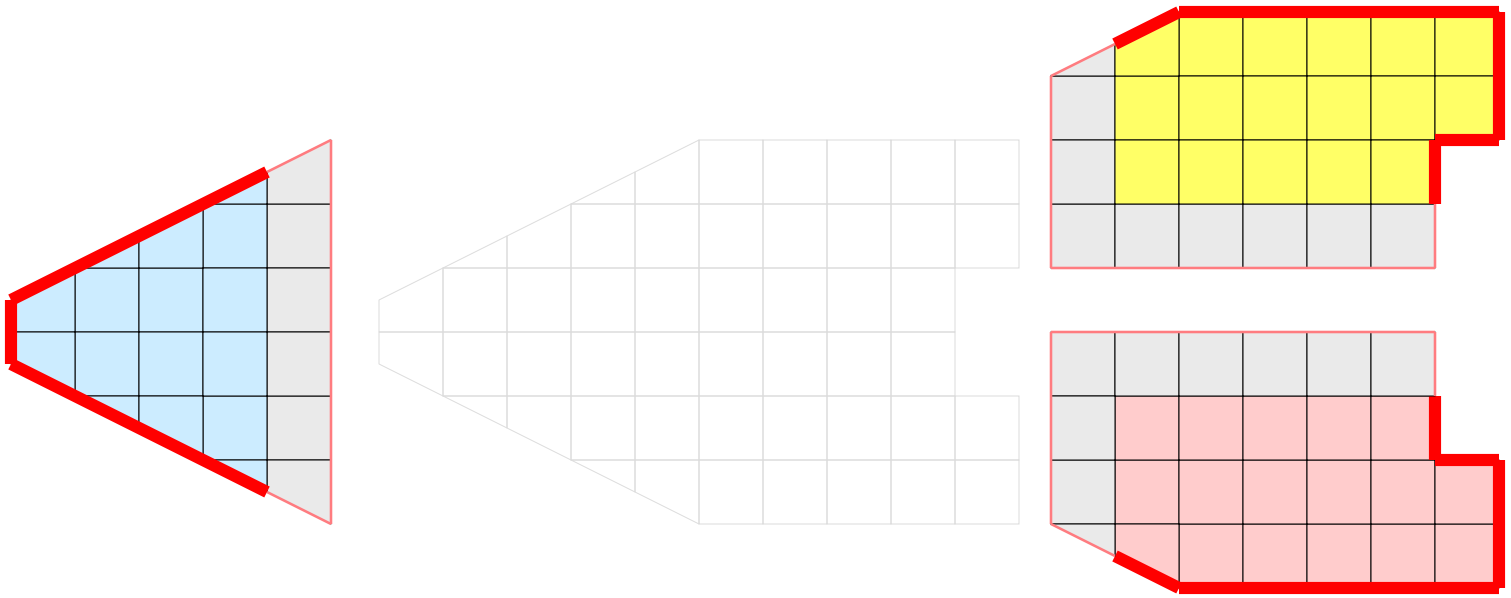
Data Parallel Pipelines

- Some operations will have problems.
  - Example: External Faces



Data Parallel Pipelines

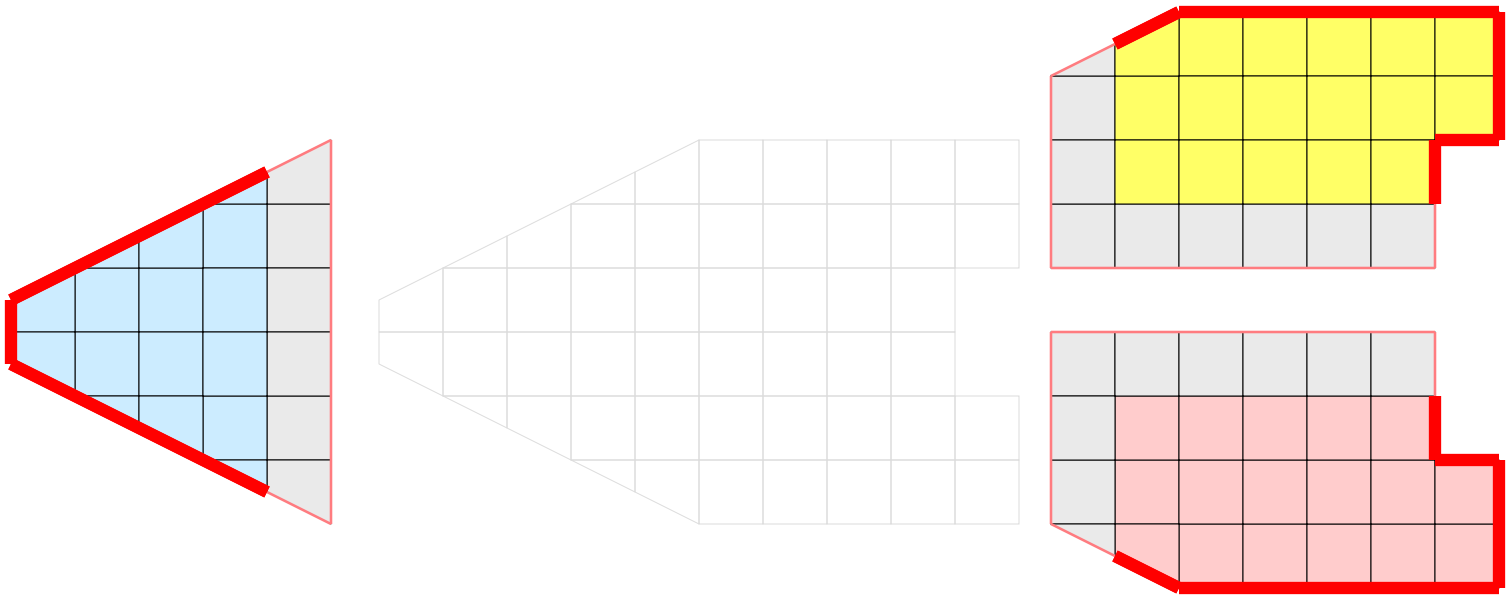
- Ghost cells can solve most of these problems.



Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

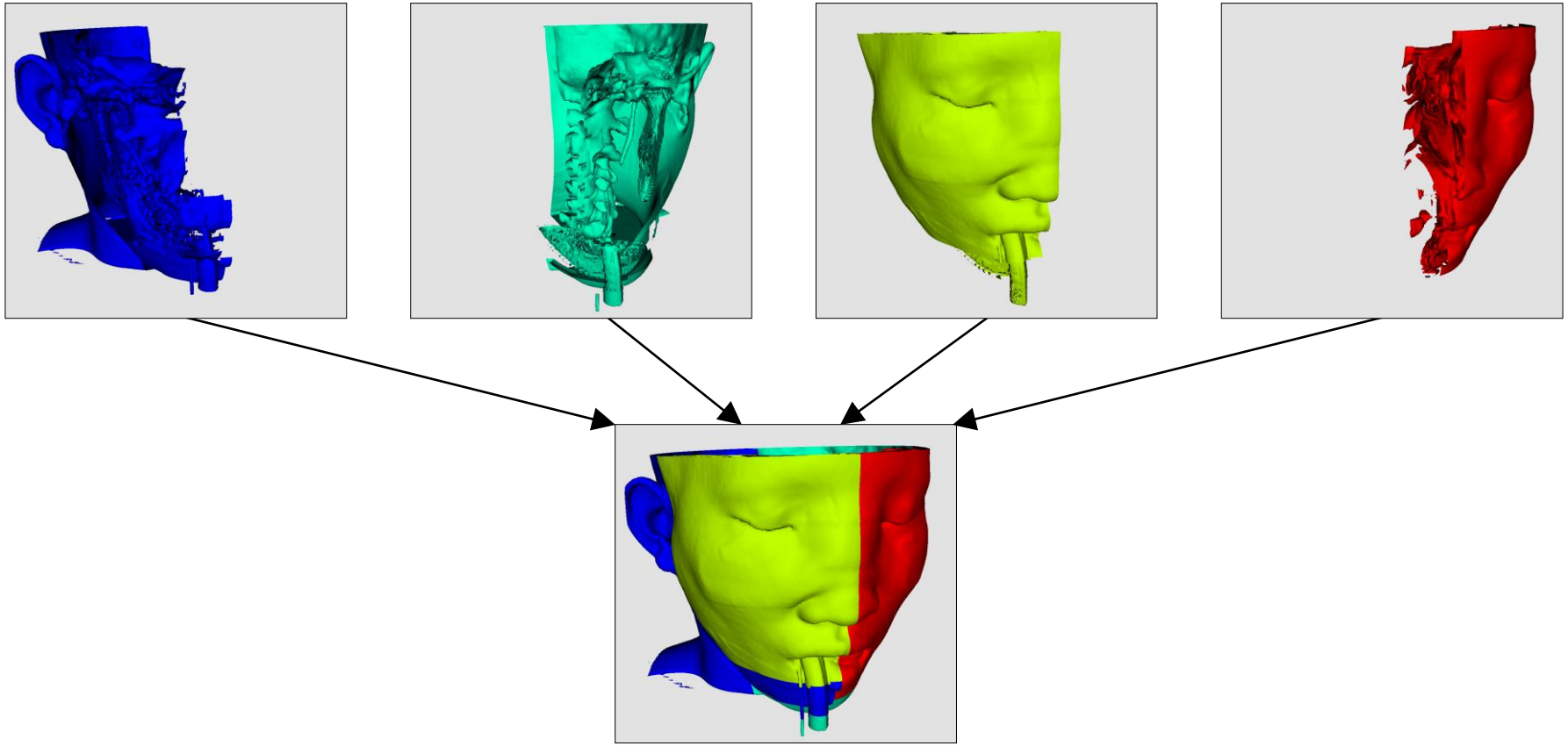
Data Parallel Pipelines

- Ghost cells can solve most of these problems.



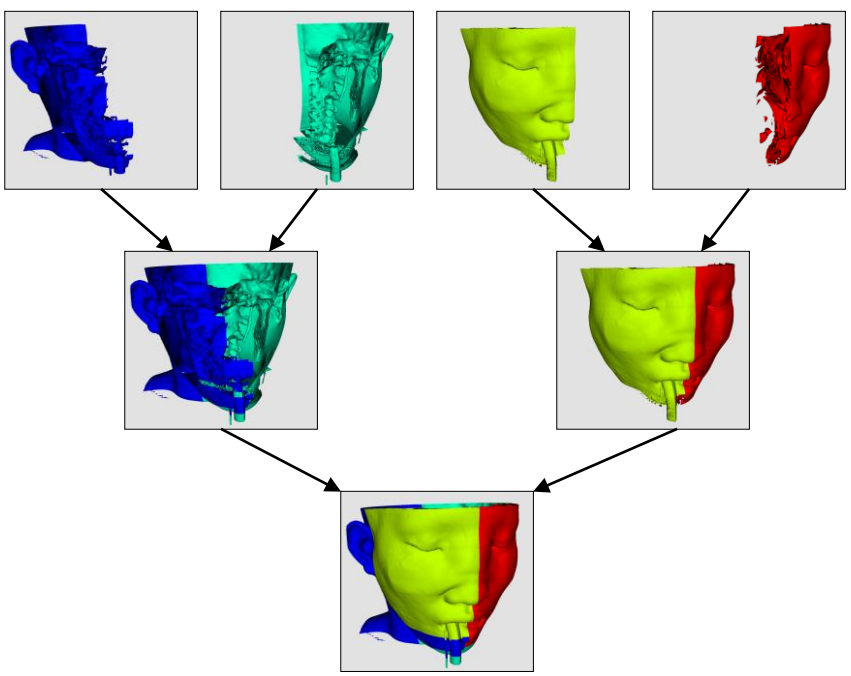
Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

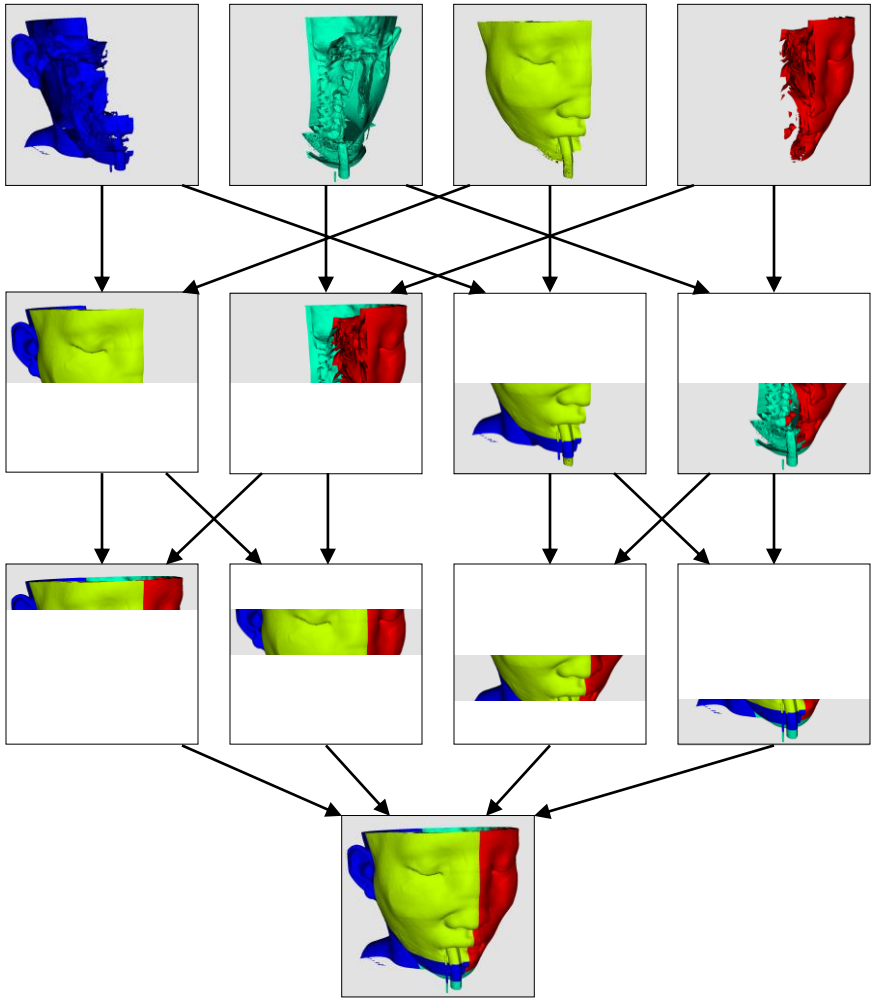
Parallel Rendering

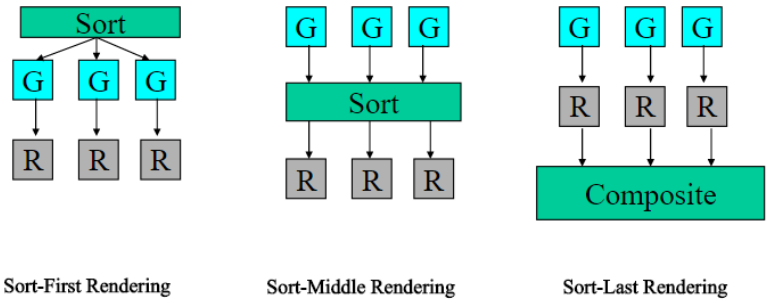


Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

Parallel Rendering





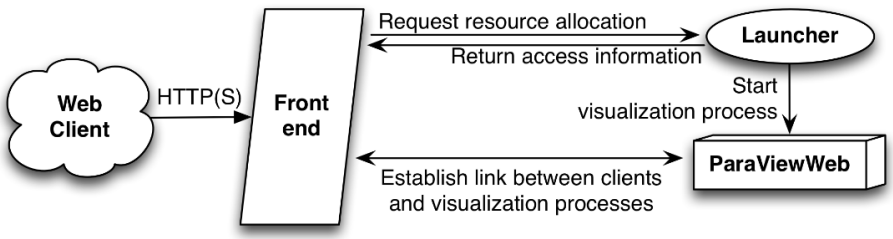


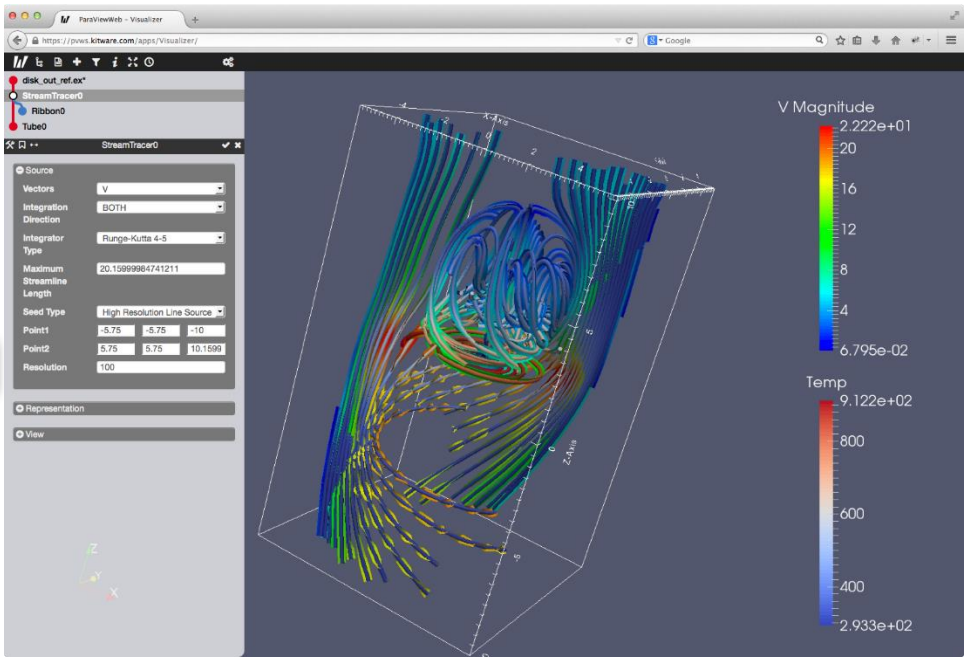
Ed Angel, http://slideplayer.com/slide/5186942/

Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

ParaView Web

- Web remote visualization framework.
- ParaView client on the web.

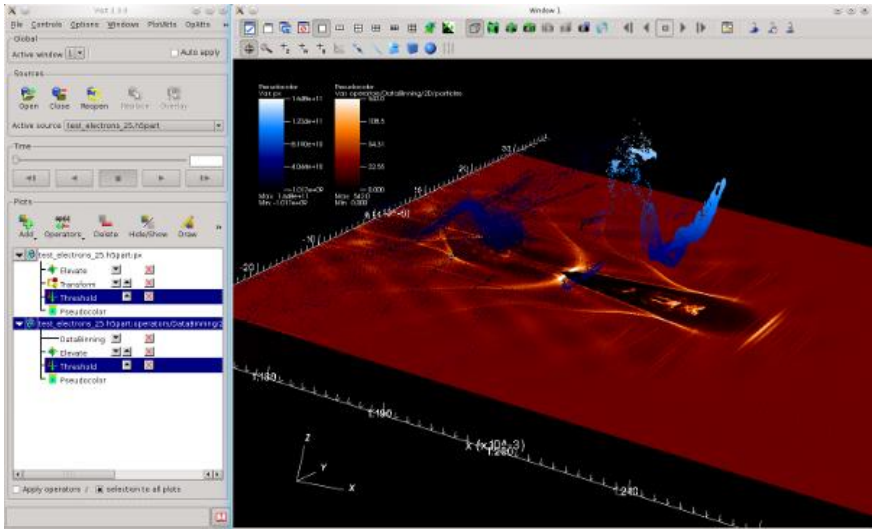


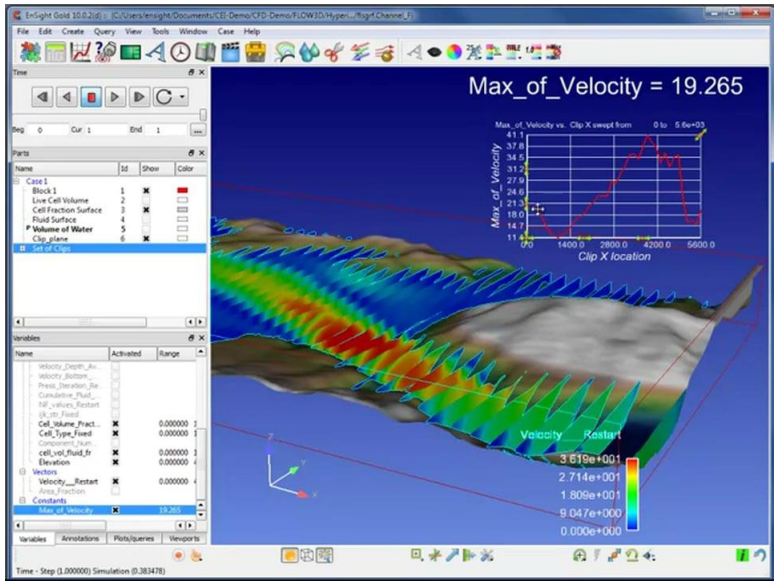


Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

ParaView Alternatives

- VisIt – also open-source and built on VTK
- EnSight - commercial





Images from http://www.nersc.gov/users/data-analytics/data-visualization/visit-2/
and https://www.ensight.com/flow-3d-post-processing/

In-Situ Visualization

Catalyst

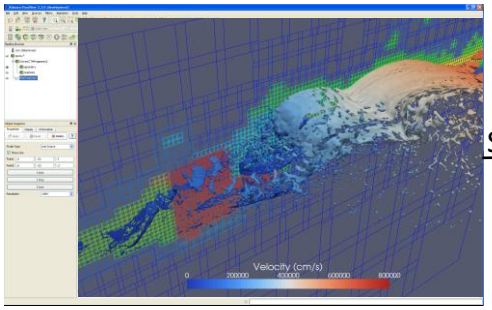
Why *In Situ*?

System Parameter	2011	"2018"		Factor Change
System peak	2 PF	1 EF		500
Power	6 MW	≤20 MW		3
System Memory	0.3 PB	32-64 PB		33
Node Performance	0.125 Tf/s	1 TF	10 TF	8-80
Node Concurrency	12	1,000	10,000	83-830
Network BW	1.5 GB/s	100 GB/s	1,000 GB/s	66-660
System Size (nodes)	18,700	1M	100k	50
Total Concurrency	225 K	10 B	100 B	40k-400k
Storage Capacity	15 PB	300-1,000 PB		20-67
I/O BW	0.2 TB/s	20-60 TB/s		100-300

Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

Source: "Scientific Discovery at the Exascale: Report from the DOE ASCR 2011 Workshop on Exascale Data Management, Analysis and Visualization."

In Situ Analysis and Visualization



Script Export

```
# Create the reader and set the filename.
reader = servermanager.sources.Reader(FileNames=path)
view = servermanager.CreateRenderView()
repr = servermanager.CreateRepresentation(reader, view)
reader.UpdatePipeline()
dataInfo = reader.GetDataInformation()
pDinfo = dataInfo.GetPointDataInformation()
arrayInfo = pDInfo.GetArrayInformation("displacement9")
if arrayInfo:
 # get the range for the magnitude of displacement9
 range = arrayInfo.GetComponentRange(-1)
 lut = servermanager.rendering.PVLookupTable()
 lut.RGBPoints  = [range[0], 0.0, 0.0, 1.0,
                   range[1], 1.0, 0.0, 0.0]
 lut.VectorMode = "Magnitude"
 repr.LookupTable = lut
 repr.ColorArrayName = "displacement9"
 repr.ColorAttributeType = "POINT_DATA"
```

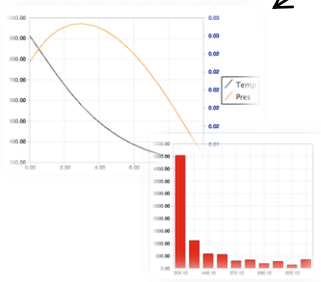
Augmented script in input deck.

Simulation

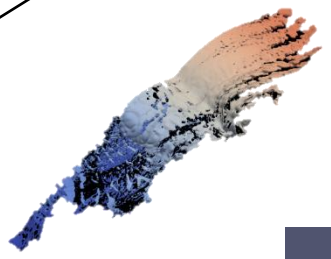
ParaView Catalyst

Output Processed Data

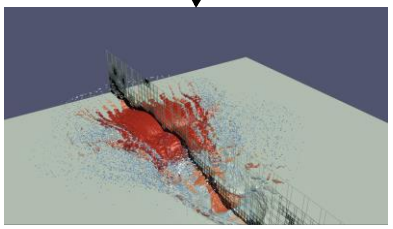
Statistics



Series Data



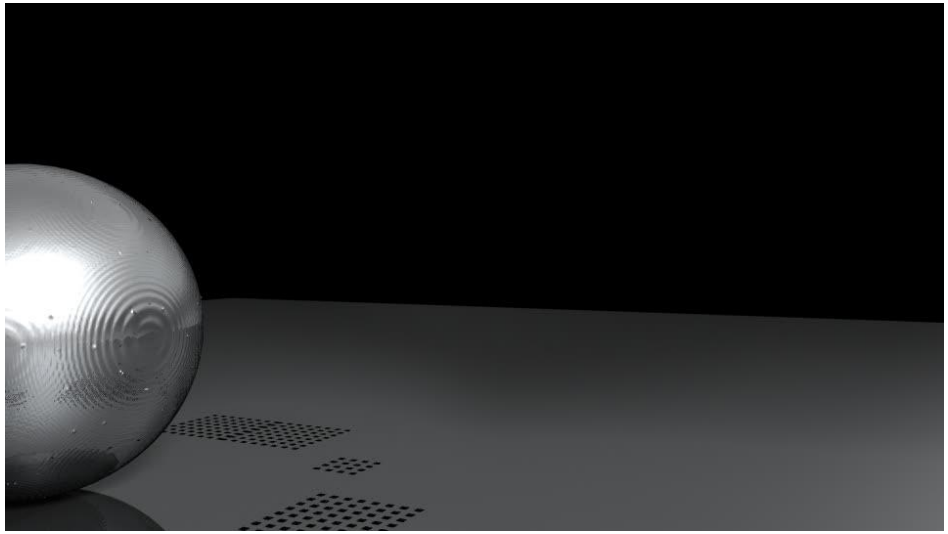
Polygonal Output with Field Data

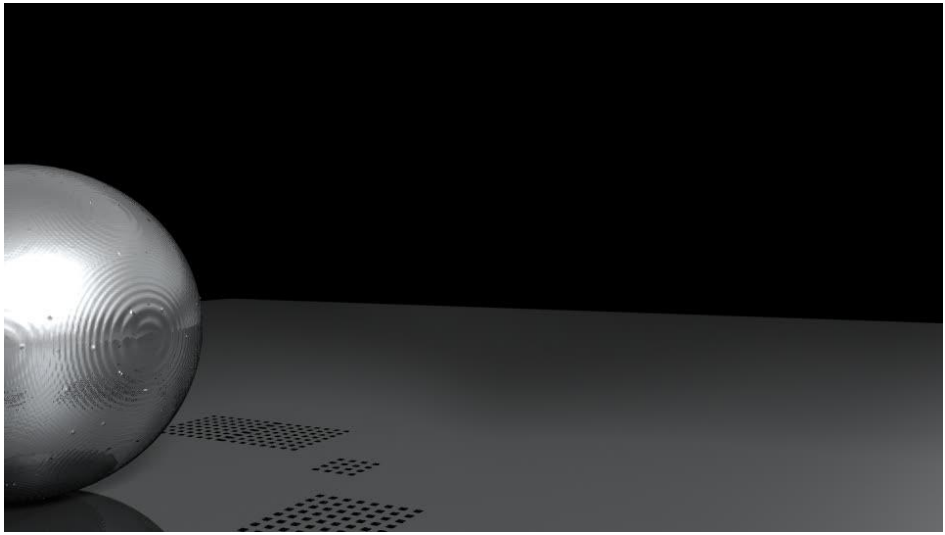


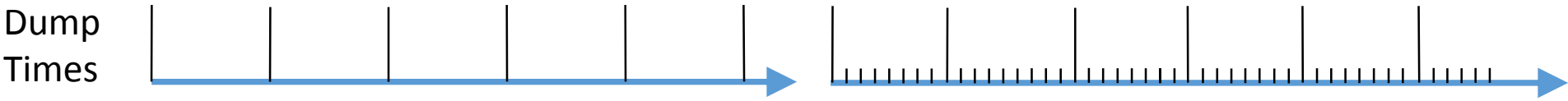
Rendered Images

Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

Access to More Data







Post-processing

In situ processing

CTH (Sandia) simulation with roughly equal data stored at simulation time

Reflections and shadows added in post-processing for both examples

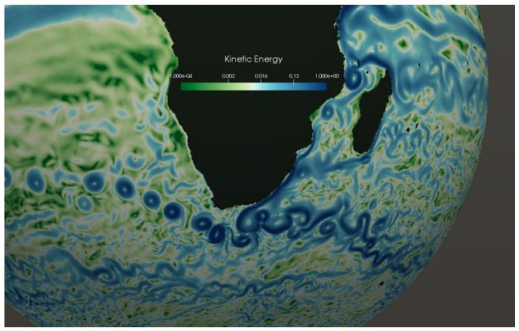
Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

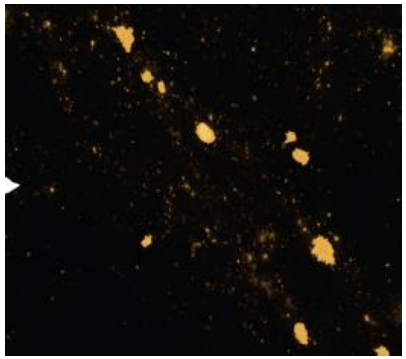
ParaView GUI Plugin

- Creates Catalyst scripts to use with simulation runs
- Similar to using ParaView interactively
  - Setup desired pipelines
  - Ideally, start with a representative data set from the simulation
- Extra pipeline information to tell what to output during simulation run
  - Add in data extract writers
  - Create screenshots to output
  - Both require file name and write frequency

Feature Extraction

- Identify “features” of interest in the data
- May be able to save only the features rather than all the raw data
- Examples
  - Ocean eddies (circular currents of water) in ocean climate simulations
  - Halos (regions of high density) in cosmology dark matter particle simulations





Eddies from “Visualization of Ocean Currents and Eddies in a High-Resolution Global Ocean-Climate Model” by Samsel et. Al.
Halos from “Large Scale Simulations of Sky Surveys” by Heitmann et. al.

Image Databases

Cinema

Image Database

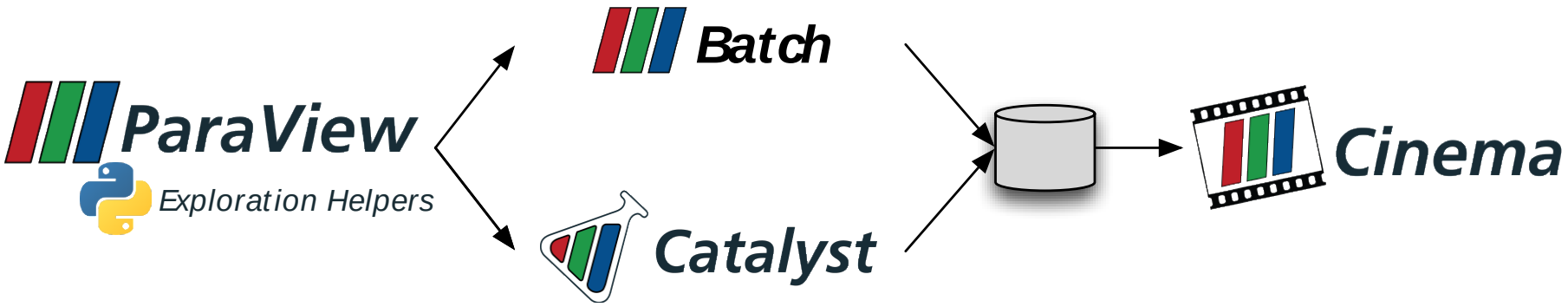
In situ visualization and analysis completed with interactive exploration using an image database.

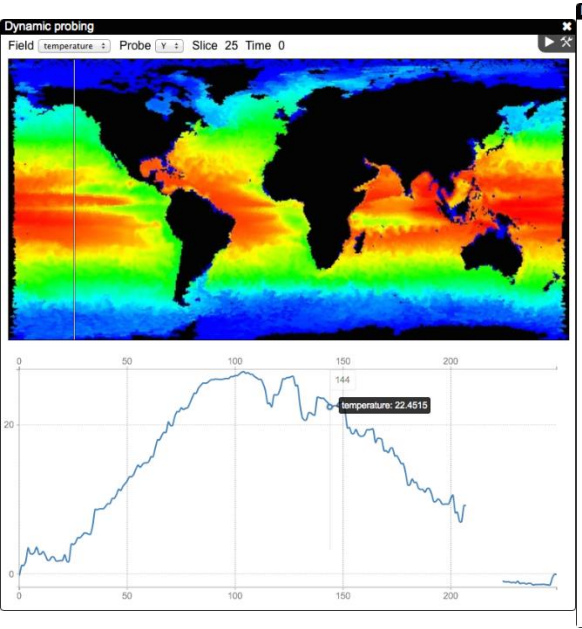
Viable solution for extreme scale visualization and analysis. Our framework:

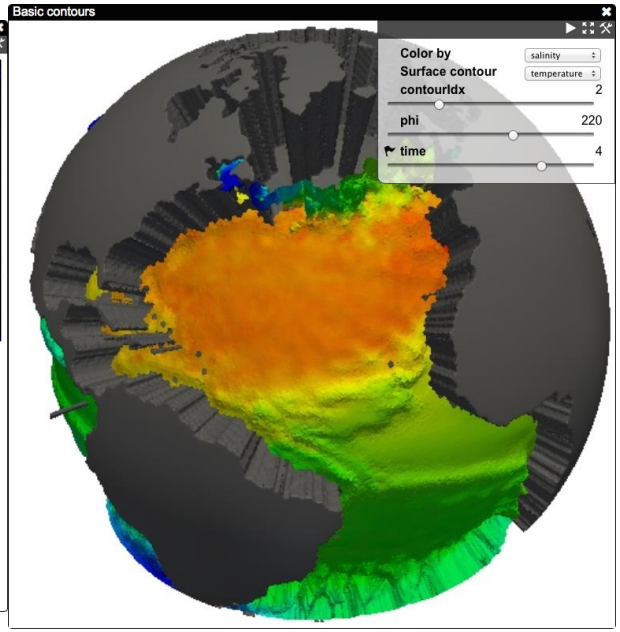
- Enables different interaction modes
  - (animation and selection for objects (isosurface value),
  - camera (rotation),
  - time) then we imagined possible with a set (database) of pre-generated analysis results.
- Responsive interactive solution
  - rivals modern post-processing approaches
  - Produces constant time retrieval (and assembly) of visualization objects from the image database.
- Encourages the use of typically avoided analysis in post-processing approaches
  - computationally intensive analysis
  - temporal exploration

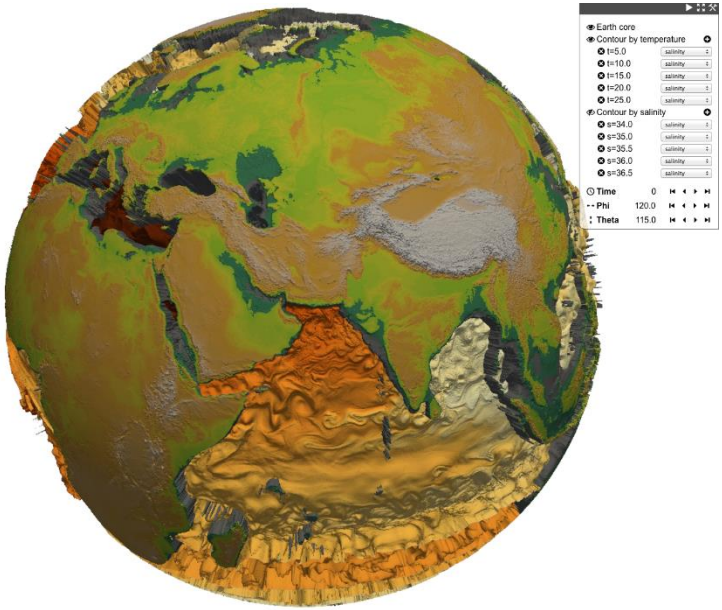
John Patchett, Presentation at Kaiserlautern University, 2014
https://datascience.lanl.gov/data/KaiserslauternInSituPatchett.pptx

What Can Be Done?



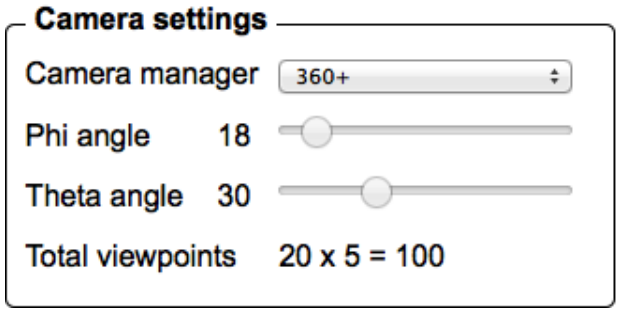


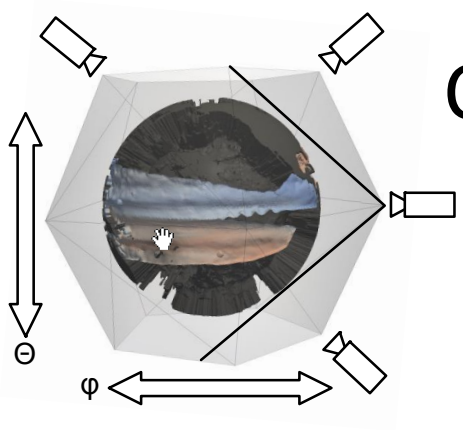




Ken Moreland, Alan Scott, David DeMarle, Li-Ta Lo, Joseph Insley, and Rich Cook, Tutorial at Supercomputing 2015
http://www.paraview.org/Wiki/images/8/8e/ParaView_Tutorial_Slides.pptx

Interactive on What?

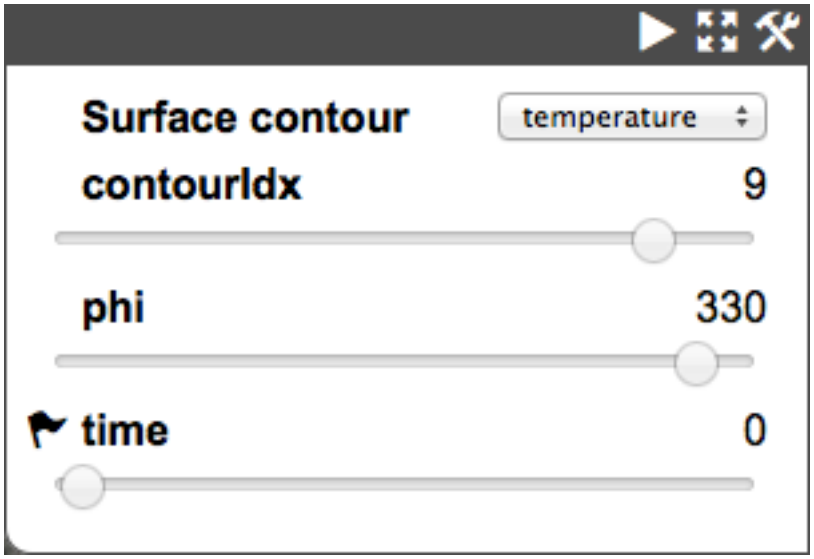


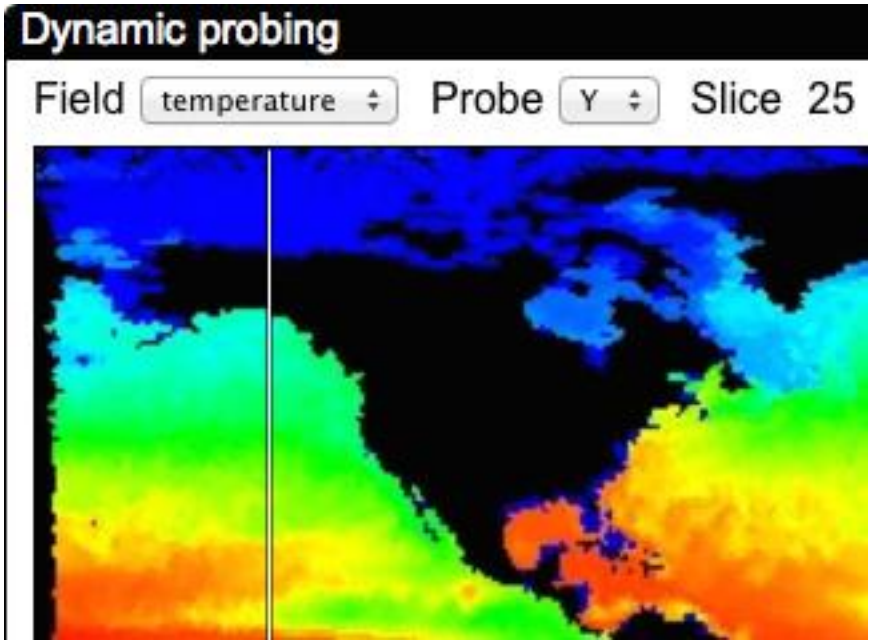


Camera

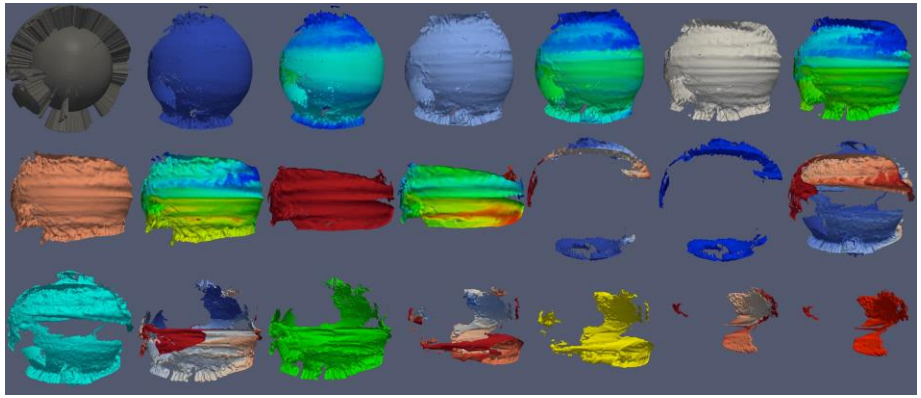
Time

Filter Parameter



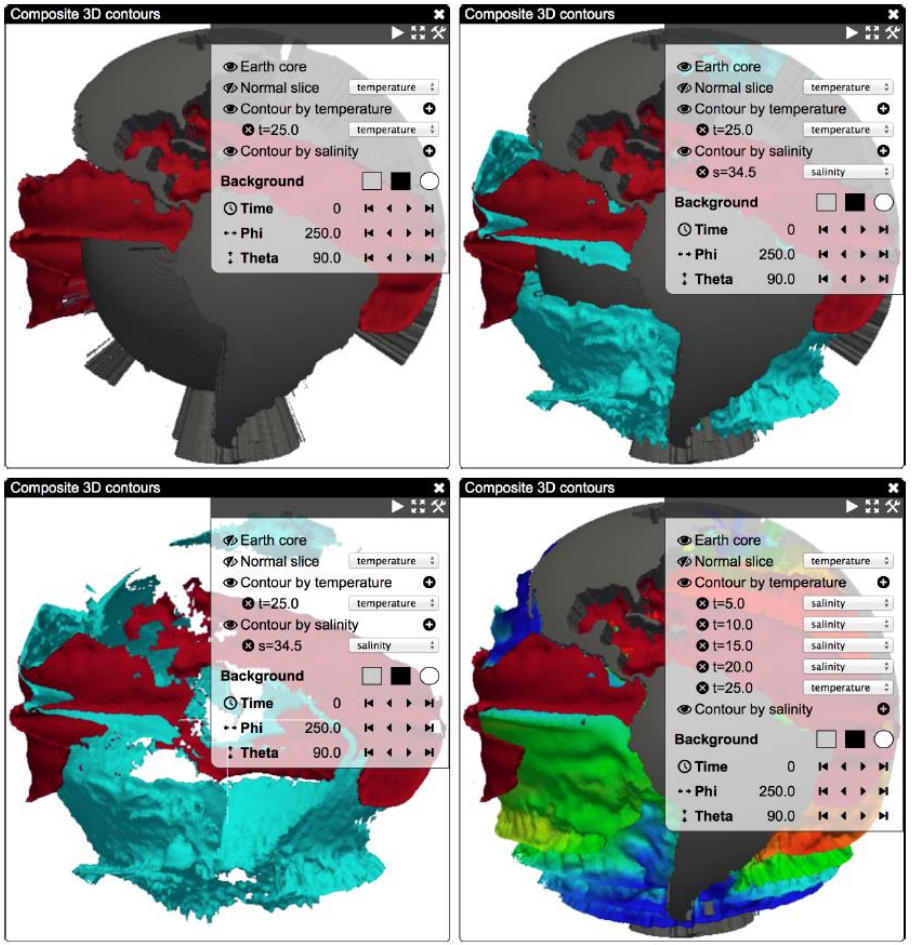


Composable Imagery



For visualization object compositing, the image is replaced by an image set consisting of an image sprite file, a composite file, and a query file.

Images can be extracted from image sprite file and properly composited in a combinatorial number of ways, like those to the right.

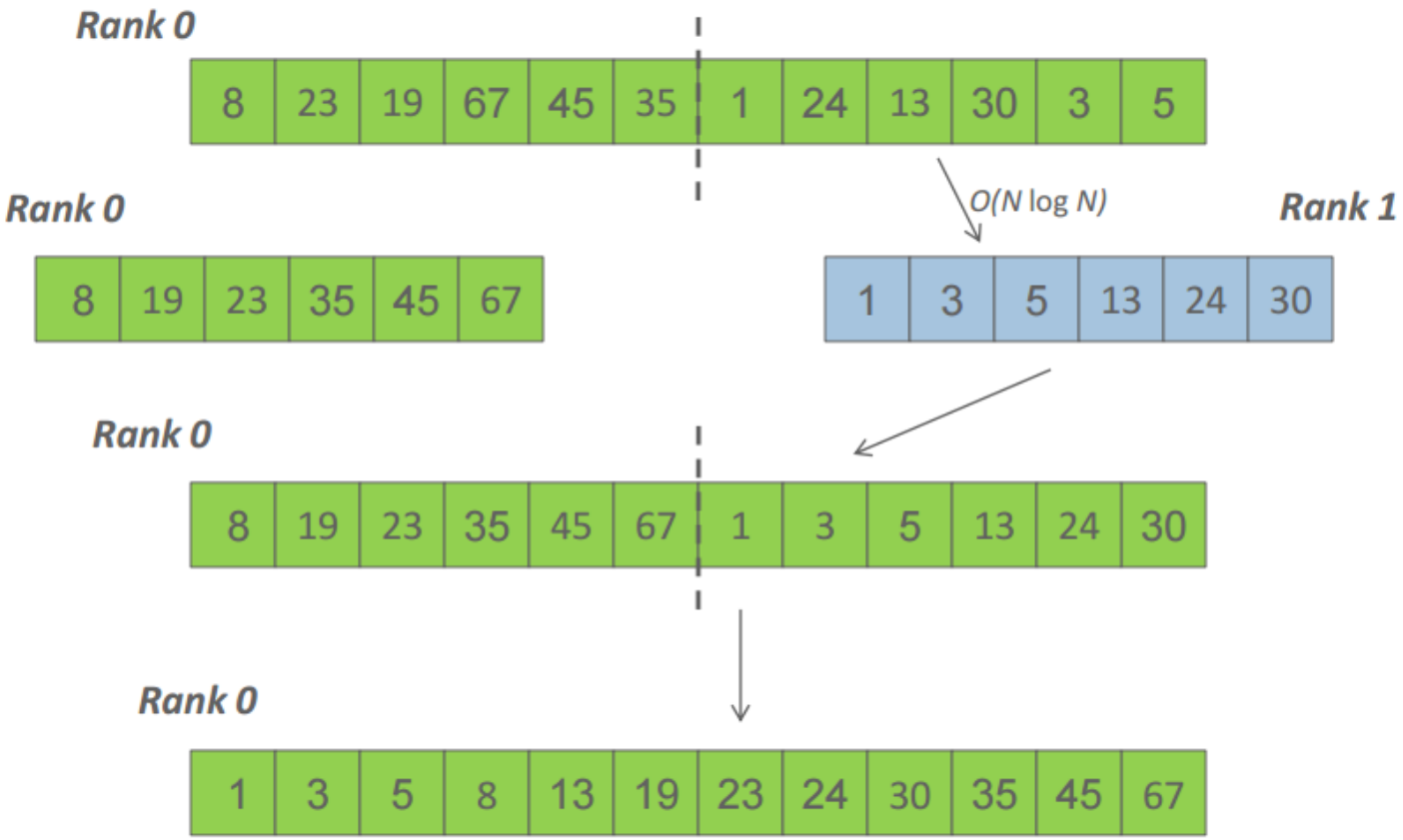


John Patchett, Presentation at Kaiserlautern University, 2014
https://datascience.lanl.gov/data/KaiserslauternInSituPatchett.pptx

Distributed Memory Parallelism

MPI, Legion

Parallel Sort using MPI Send/Recv



Pavan Balajii and Torsten Hoefler, Presentation at PPoPP 2013
https://htor.inf.ethz.ch/teaching/mpi_tutorials/ppopp13/2013-02-24-ppopp-mpi-basic.pdf

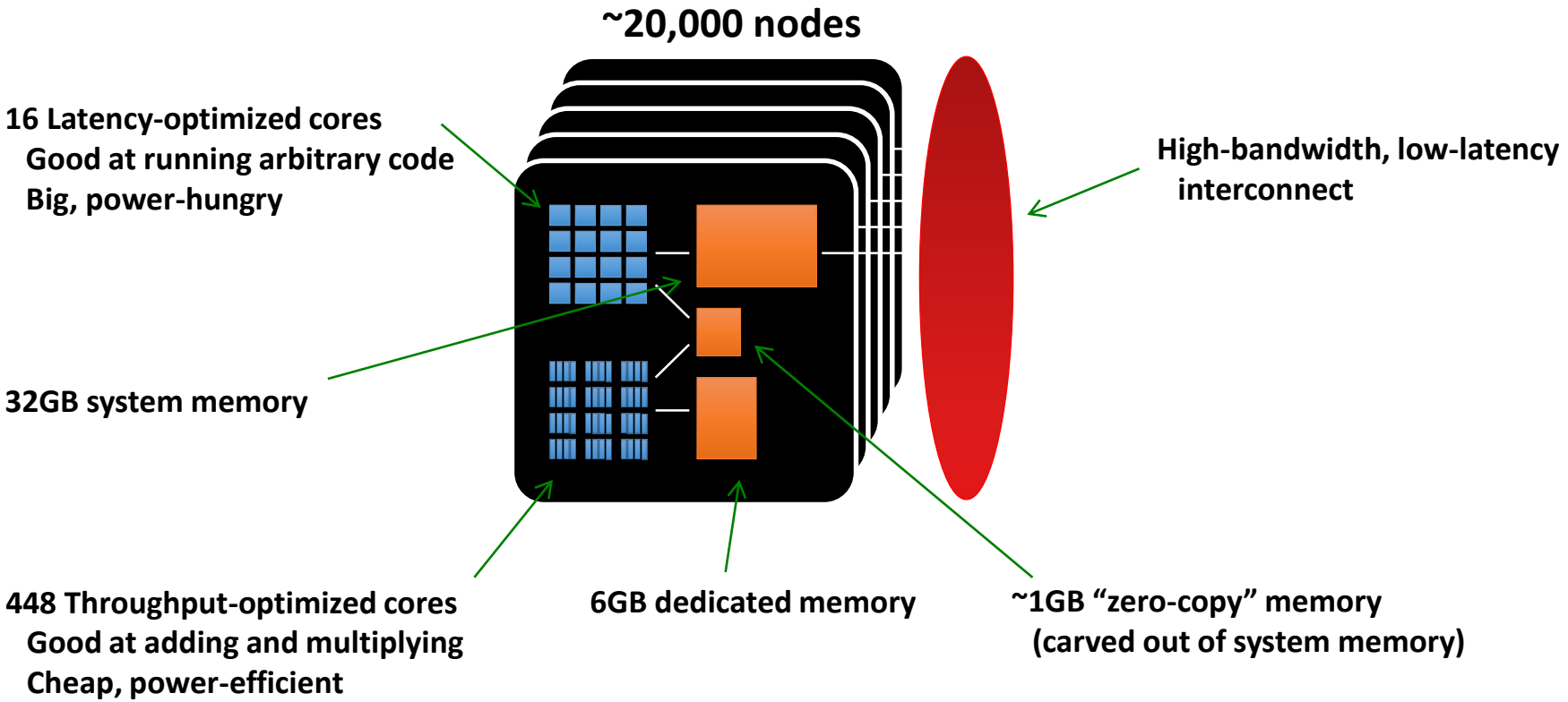
Parallel Sort using MPI Send/Recv (contd.)

```
#include <mpi.h>
#include <stdio.h>
int main(int argc, char ** argv)
{
    int rank;
    int a[1000], b[500];

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    if (rank == 0) {
        MPI_Send(&a[500], 500, MPI_INT, 1, 0, MPI_COMM_WORLD);
        sort(a, 500);
        MPI_Recv(b, 500, MPI_INT, 1, 0, MPI_COMM_WORLD, &status);

        /* Serial: Merge array b and sorted part of array a */
    }
    else if (rank == 1) {
        MPI_Recv(b, 500, MPI_INT, 0, 0, MPI_COMM_WORLD, &status);
        sort(b, 500);
        MPI_Send(b, 500, MPI_INT, 0, 0, MPI_COMM_WORLD);
    }

    MPI_Finalize(); return 0;
}
```


Pavan Balajii and Torsten Hoefler, Presentation at PPoPP 2013
https://htor.inf.ethz.ch/teaching/mpi_tutorials/ppopp13/2013-02-24-ppopp-mpi-basic.pdf

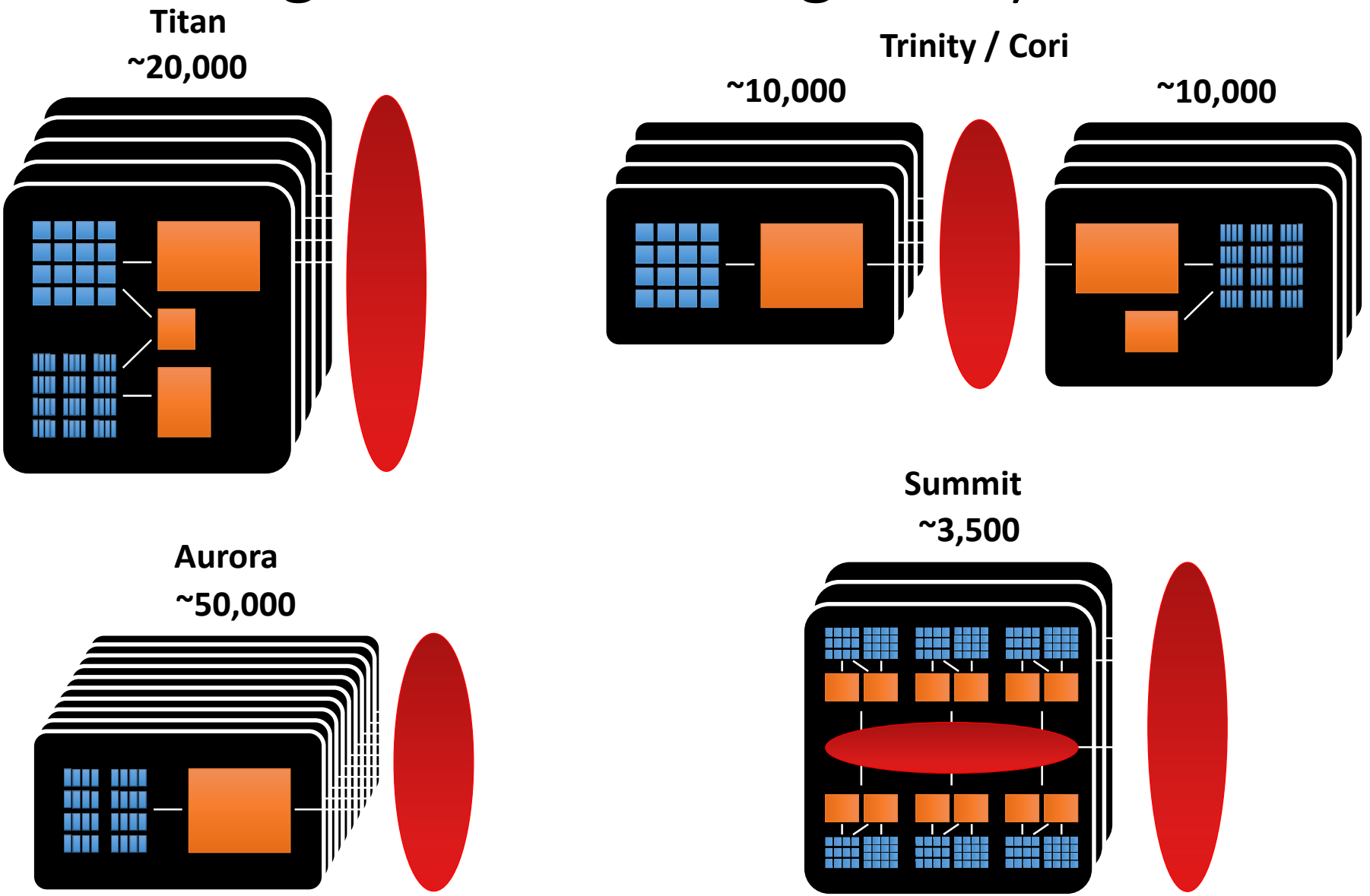
Heterogeneity

- System Architecture for Titan (#2 on Top500)



Elliott Slaughter, Presentation at Oak Ridge National Lab, 2016
https://www.olcf.ornl.gov/wp-content/uploads/2016/01/Legion_Elliott_Slaughter.pptx

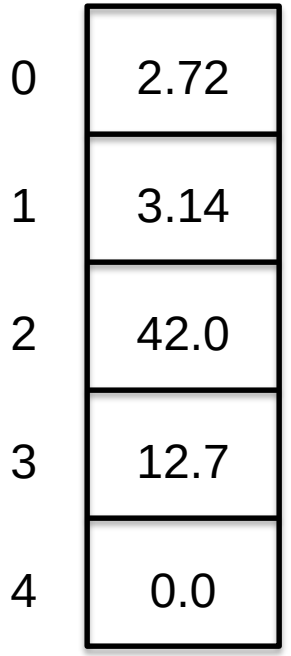
Heterogeneous Heterogeneity



Elliott Slaughter, Presentation at Oak Ridge National Lab, 2016
https://www.olcf.ornl.gov/wp-content/uploads/2016/01/Legion_Elliott_Slaughter.pptx

Legion: Tasks & Regions

- A *task* is the unit of parallel execution
  - I.e. a function
- Task arguments are *regions*
  - Collections
  - Rows are an *index space*
  - Columns are *fields*
- Tasks declare how they use their regions




```
task saxpy(is : ispace(int1d), x,y: region(is, float), a: float )
where reads(x, y), writes(y)
```

Elliott Slaughter, Presentation at Oak Ridge National Lab, 2016
https://www.olcf.ornl.gov/wp-content/uploads/2016/01/Legion_Elliott_Slaughter.pptx

Example Task

```
task saxpy(is: ispace(int1d), x: region(is, float),
                      y: region(is, float), a: float)
where
  reads(x, y), writes(y)
do
  for i in is do
    y[i] += a*x[i]
  end
end
```

Elliott Slaughter, Presentation at Oak Ridge National Lab, 2016
https://www.olcf.ornl.gov/wp-content/uploads/2016/01/Legion_Elliott_Slaughter.pptx

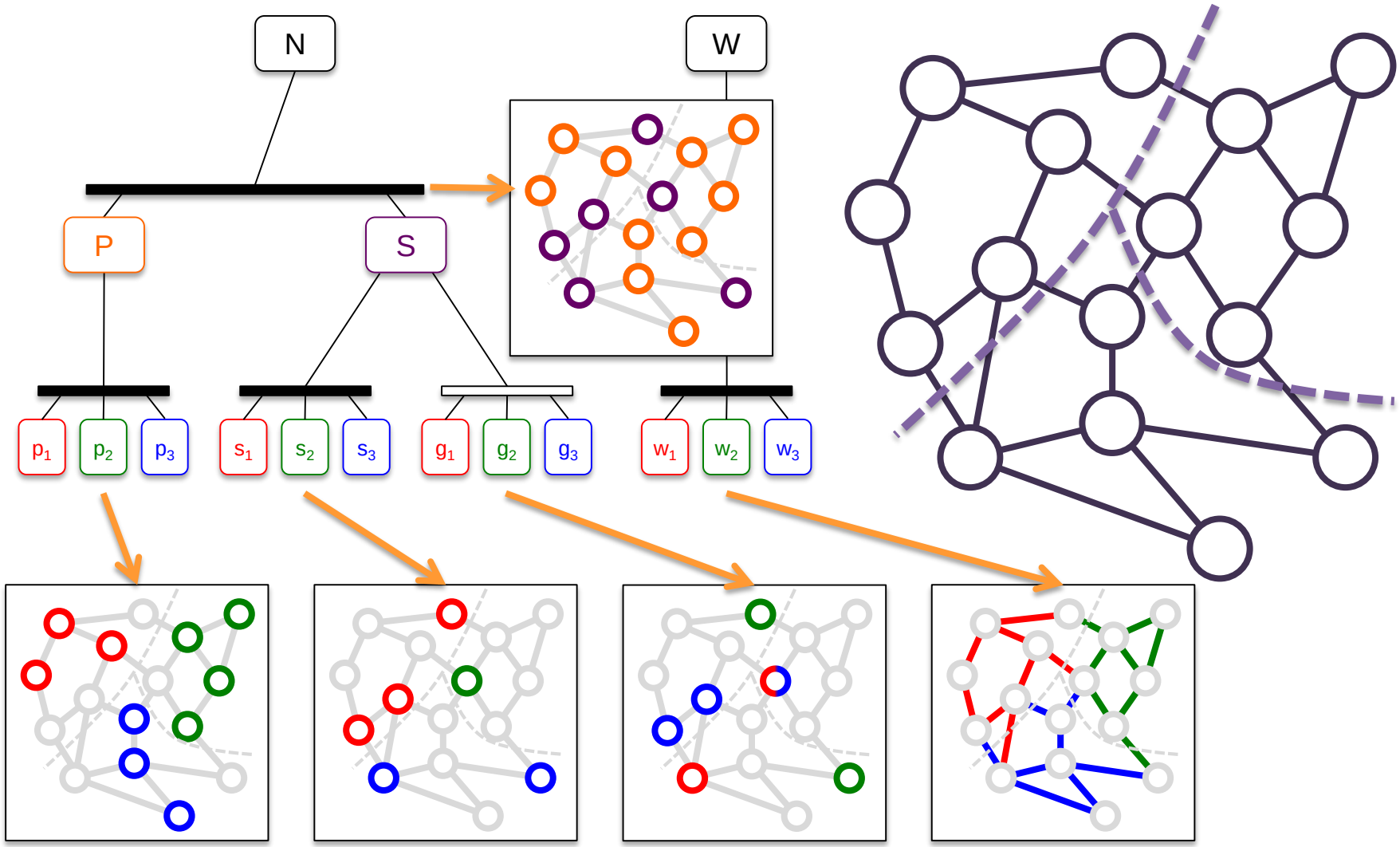
Regions

- Regions can be *partitioned* into *subregions*

- Partitioning is a primitive operation
  - Supports describing arbitrary subsets of a region

Elliott Slaughter, Presentation at Oak Ridge National Lab, 2016
https://www.olcf.ornl.gov/wp-content/uploads/2016/01/Legion_Elliott_Slaughter.pptx

Partitioning



Elliott Slaughter, Presentation at Oak Ridge National Lab, 2016
https://www.olcf.ornl.gov/wp-content/uploads/2016/01/Legion_Elliott_Slaughter.pptx

Tasks

- Tasks can call *subtasks*
  - Sequential semantics, implicit parallelism
  - If tasks do not *interfere,* can be executed in parallel

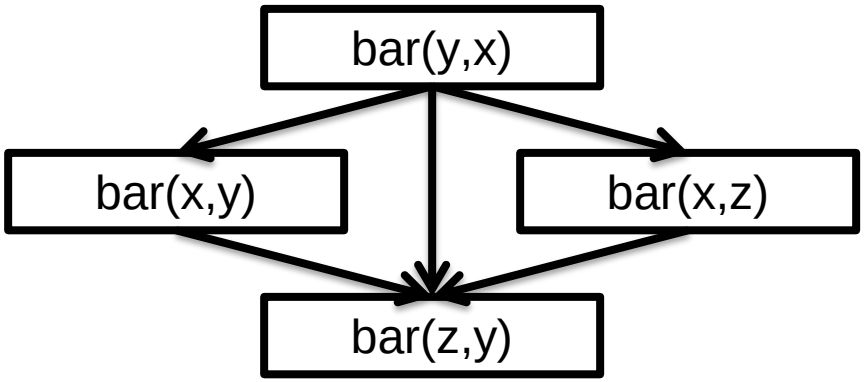
```
task foo(x,y,z: region(…))
where reads writes(x,y,z) do
        bar(y,x)
        bar(x,y)
        bar(x,z)
        bar(z,y)
end
task bar(r,s: region(…)) where reads(r), writes(s)
```

Elliott Slaughter, Presentation at Oak Ridge National Lab, 2016
https://www.olcf.ornl.gov/wp-content/uploads/2016/01/Legion_Elliott_Slaughter.pptx

Deferred Execution

```
task foo(x,y,z: region(…))
where reads writes(x,y,z) do
    bar(y,x)
    bar(x,y)
    bar(x,z)
    bar(z,y)
end
task bar(r,s: region(…)) where reads(r), writes(s)
```

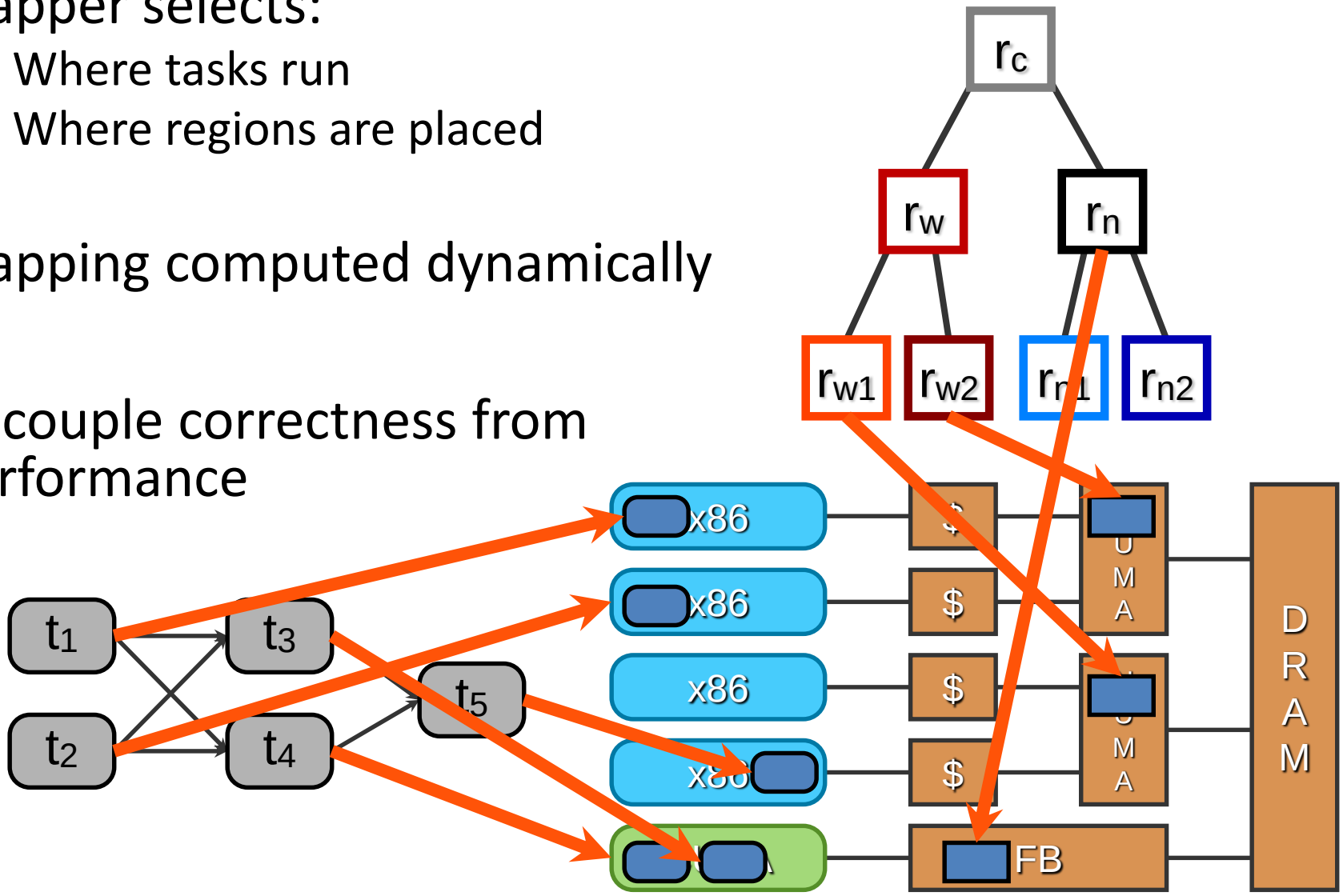
Legion Runtime



Elliott Slaughter, Presentation at Oak Ridge National Lab, 2016
https://www.olcf.ornl.gov/wp-content/uploads/2016/01/Legion_Elliott_Slaughter.pptx

Mapping Interface

- Mapper selects:
  - Where tasks run
  - Where regions are placed
- Mapping computed dynamically
- Decouple correctness from performance



Elliott Slaughter, Presentation at Oak Ridge National Lab, 2016
https://www.olcf.ornl.gov/wp-content/uploads/2016/01/Legion_Elliott_Slaughter.pptx

Shared Memory Parallelism

Data Parallel Programming

Extreme Scale: Threads, Threads Threads!

- A clear trend in supercomputing is ever increasing parallelism
- Clock increases are long gone
  - "The Free Lunch Is Over" (Herb Sutter)

	Jaguar – XT5	Titan – XK7	Exascale*
Cores	224,256	299,008 cpu and 18,688 gpu	1 billion
Concurrency	224,256 way	70 – 500 million way	10 – 100 billion way
Memory	300 Terabytes	700 Terabytes	128 Petabytes

*Source: Scientific Discovery at the Exascale, Ahern, Shoshani, Ma, et al.

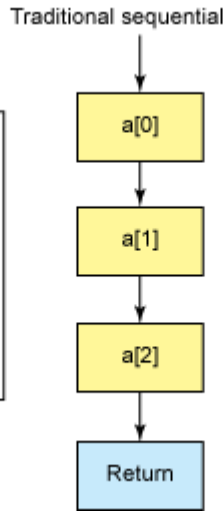
Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

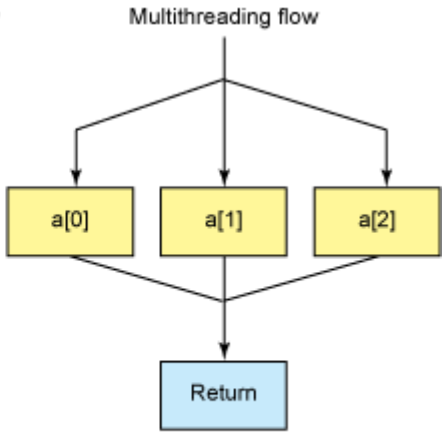
Shared Memory Programming Models

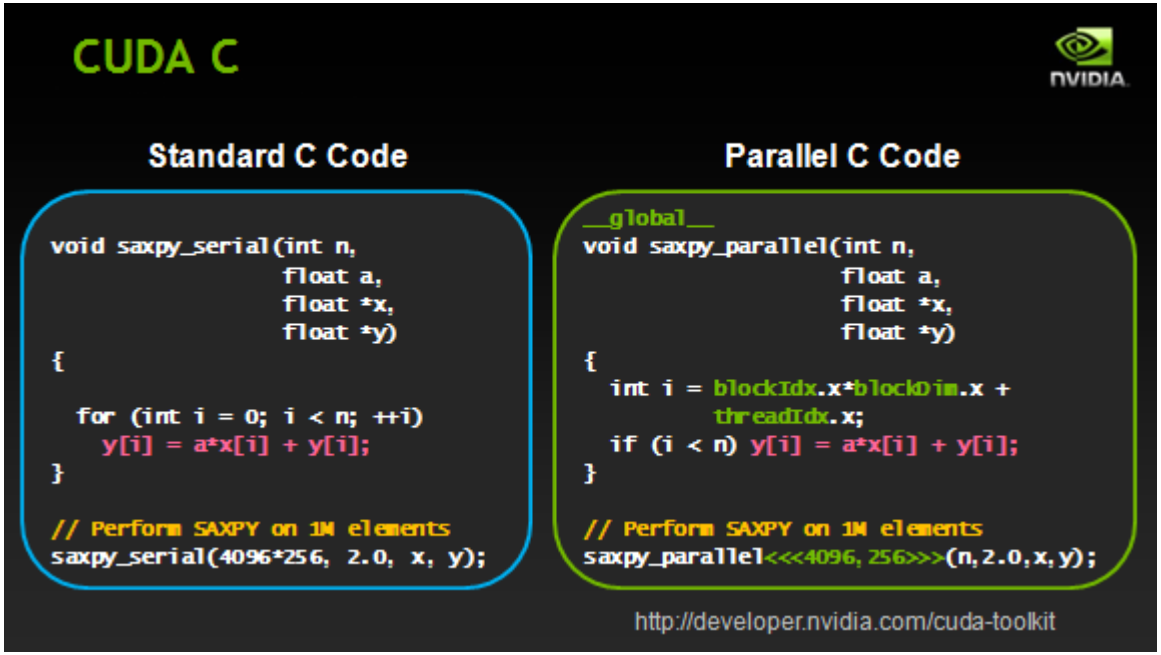
- Nvidia CUDA
- Intel Thread Building Blocks
- Intel Cilk
- OpenMP
- OpenCL
- OpenACC

Multithreading C edample with OpenMP

```
int main( int argc, char **argv)
(
        int i, a[3]
        #pragma omp parallel for
        for (i - o ; i < 3 ; i++) {
                a[i] = i*i;
        )
        return 0;
}
```



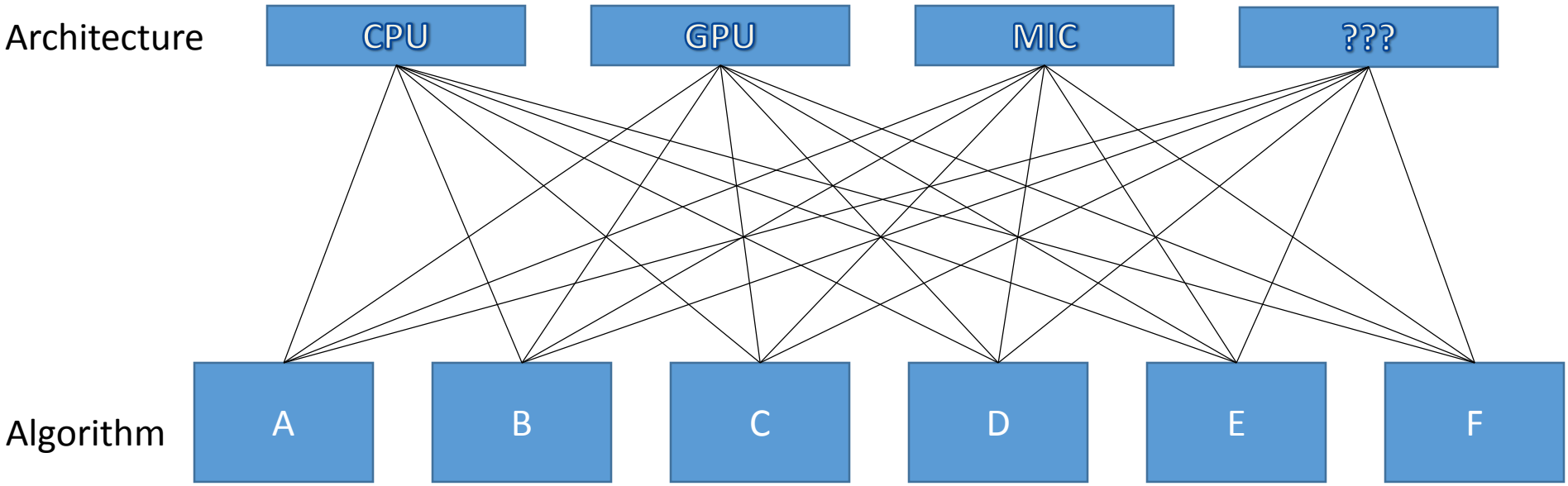




CUDA image from:
http://www.nvidia.co.uk/object/cuda-parallel-computing-uk.html

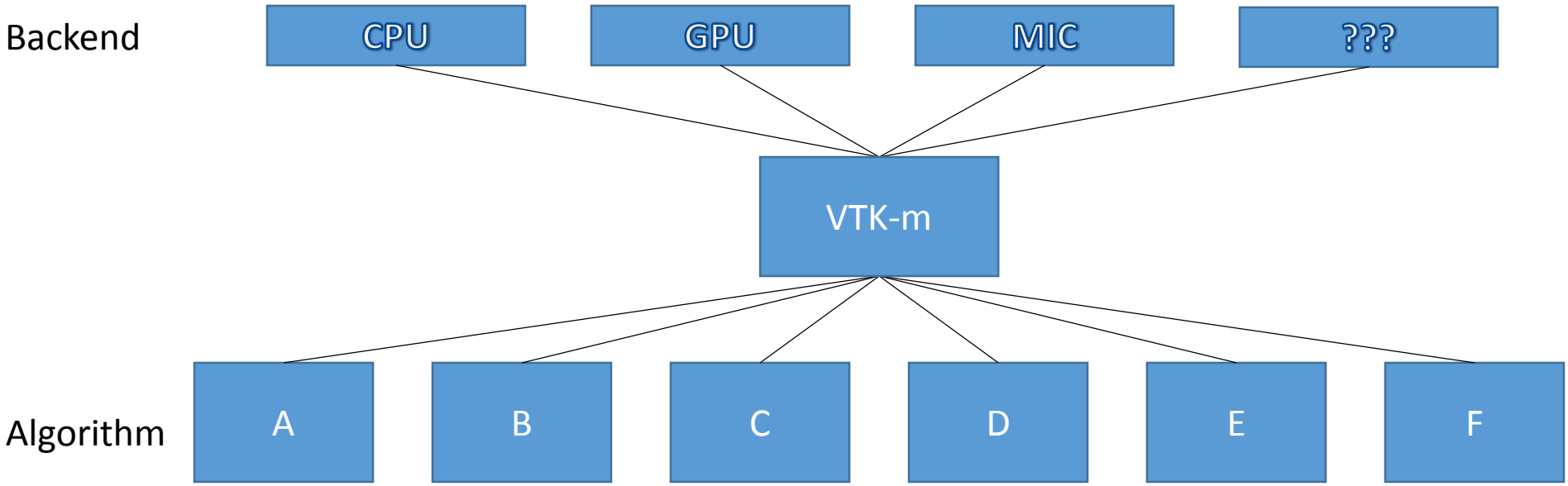
OpenMP image from:
http://www.ibm.com/developerworks/library/l-gcc4/

Performance Portability



Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

Performance Portability



Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

NVIDIA's Thrust Library

- Thrust is an open-source C++ template library developed by NVIDIA
- It allows the user to write CUDA programs using an STL-like interface, without having to know CUDA-specific syntax or functions
- In addition to CUDA, it has backends for OpenMP and Intel TBB, and can be extended to support additional backends
- It implements many data-parallel primitives, with user-defined functors
- It provides thrust::host_vector and thrust::device_vector, simplifying memory management and data transfer between the host and device

```
#include <thrust/transform_reduce.h>
#include <thrust/functional.h>
#include <thrust/device_vector.h>
#include <thrust/host_vector.h>
#include <cmath>

// square<T> computes the square of a number f(x) -> x*x
template <typename T>
struct square
{
    __host__ __device__
        T operator()(const T& x) const {
            return x * x;
        }
};

int main(void)
{
    // initialize host array
    float x[4] = {1.0, 2.0, 3.0, 4.0};

    // transfer to device
    thrust::device_vector<float> d_x(x, x + 4);

    // setup arguments
    square<float>        unary_op;
    thrust::plus<float> binary_op;
    float init = 0;

    // compute norm
    float norm = std::sqrt(thrust::transform_reduce(d_x.begin(),
                           d_x.end(), unary_op, init, binary_op));

    std::cout << norm << std::endl;

    return 0;
}
```

Sample Thrust code to compute vector norm

Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

Brief Introduction to Data-Parallel Programming and Thrust

What algorithms does Thrust provide?

- Sorts
- Transforms
- Reductions
- Scans
- Binary searches
- Stream compactions
- Scatters / gathers

Challenge: Write operators in terms of these primitives only

Reward: Efficient, portable code

input	4	5	2	1	3
transform(+1)	5	6	3	2	4
inclusive_scan(+)	4	9	11	12	15
exclusive_scan(+)	0	4	9	11	12
exclusive_scan(max)	0	4	5	5	5
transform_inscan(*2,+)	8	18	22	24	30
for_each(-1)	3	4	1	0	2
sort	1	2	3	4	5
copy_if(n % 2 == 1)	5	1	3		
reduce(+)					15
input1	0	0	2	4	8
input2	3	4	1	0	2
upper_bound	3	4	2	2	3
permutation_iterator	4	8	0	0	2

Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

Five Operations You Can Do with a Lot of Data in Parallel

- Generate/Create
  - Automatically fill with programmatically defined data
- Transform
  - Apply some “operation” to each element of the data
  - Also called “Map” in many contexts
- Compact
  - Take only the elements in which you are interested
  - Also called “Filter” in many contexts
- Expand
  - The opposite of Compact
  - Create a larger data set from a smaller data set
- Aggregate
  - Calculate a “summary” of your data (e.g., sum or average)
  - Also called “Reduce” or “Fold”
  - “Scan” also provides all intermediate values

Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

Simple Examples with Thrust Pseudocode

- Generate

```
thrust::sequence(0,4)   0  1  2  3  4
```

- Transform

```
input                   4  5  2  1  3
thrust::transform(+1)   5  6  3  2  4
```

- Compact

```
input                   4  5  2  1  3
thrust::copy_if(even)   4  2
```

- Expand

```
input                   4  5  2  1  3
thrust::for_each(x,2x)  4  8  5 10  2  4  1  2  3  6
```

- Aggregate

```
input                   4  5  2  1  3
thrust::reduce(+)      15
```

Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

Generate Data in Parallel

- Many copies of a certain constant value

```
// Ten elements with initial value of integer 1
thrust::device_vector<int> x(10, 1);
```

- A sequence of increasing or decreasing values

```
// Allocate space for ten integers, uninitialized
thrust::device_vector<int> y(10);
// Fill the space with integers
thrust::sequence(y.begin(), y.end(), 5, 2);
```

- Random Values
  - Multiple copies of a random number generator
  - Give each one a different seed

Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

Transform: Vector Addition

- Apply a binary operator “plus” to each element in x and y

```
thrust::transform(x.begin(), x.end(), // begin and end of the
                                      // first input vector
y.begin(),                            // begin of the second
                                      // input vector
result.begin(),                       // begin of the result
                                      // vector
thrust::plus<int>());                 // predefined integer
                                      // addition
```

```
     x: 1  1  1  1  1  1  1  1  1  1
                   +
     y: 5  7  9 11 13 15 17 19 21 23
                   =
result: 6  8 10 12 14 16 18 20 22 24
```

Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

Transform: Uniform Sampling of a Mathematical Function

- Q: How are we going to generate something more complicated?
  A: Start from some sequence and apply some transformation
- Sampling the function $f(x) = x^2$ in the interval of [0, 1]

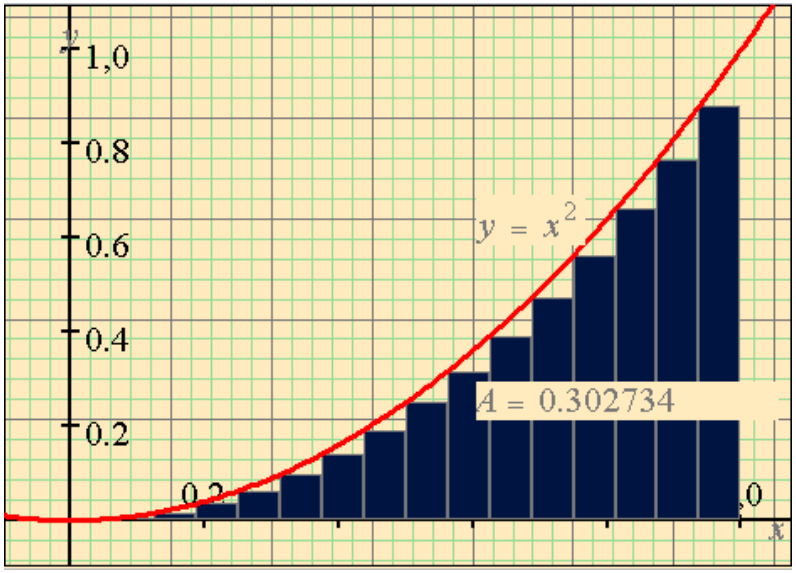
```
// Generate a sequence of x_i in [0,1] with dx=0.1
// in between each of them
float dx = 1.0f/10.0f;
thrust::sequence(x.begin(), x.end(), 0.0f, dx);

// Apply the square operation to each of the x_i
// to transform into f(x_i) = y_i = x_i^2
thrust::transform(x.begin(), x.end(),
                  y.begin(),
                  square());
```

```
x: 0.0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1.0
y: 0.0  0.01 0.04 0.09 0.16 0.25 0.36 0.49 0.64 0.81 1.0
```

Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

Reduce: Simple Numerical Integration

- Apply what we learned to estimate the area under a curve
- Create a constant vector of widths
- Create a vector of heights from the function values
- Apply multiply operation on each element of width and height
- Sum all the computed areas to get the total area
- In calculus, this is a method of estimating the integral



$$\int_0^1 f(x)dx \approx \sum_{i=1}^{n} f(x_i)\Delta x$$

Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

Simple Numerical Integration: Example

```
thrust::device_vector<int> width(11, 0.1);
width       =  0.1   0.1   0.1   0.1   0.1   0.1   0.1   0.1   0.1   0.1   0.1

thrust::sequence(x.begin(), x.end(), 0.0f, 0.1f);
x           =  0.0   0.1   0.2   0.3   0.4   0.5   0.6   0.7   0.8   0.9   1.0

thrust::transform(x.begin(), x.end(), height.begin(), square());
height      =  0.0  0.01  0.04  0.09  0.16  0.25  0.36  0.49  0.64  0.81   1.0

thrust::transform(width.begin(), width.end(), height.begin(), area.begin(),
thrust::multiplies<float>())
area        =  0.0 0.001 0.004 0.009 0.016 0.025 0.036 0.049 0.064 0.081   0.1

total_area = thrust::reduce(area.begin(), area.end());
total_area =  0.385


thrust::inclusive_scan(area.begin(), area.end(), accum_areas.begin());
accum_areas =  0.0 0.001 0.005 0.014 0.030 0.055 0.091 0.140 0.204 0.285 0.385
```

Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

Scan: Simple Numerical Integration

- What happens if we are interested in the integral $F(t) = F(0) + \int_0^t f(x)dt$ on the interval [0, 1] instead of just a number?
- Calculate a running sum by using scan
- `thrust::inclusive_scan(y_dx.begin(), y_dx.end(), F.begin());`
- $f(x_i)*dx$ = 0.0 0.001 0.004 0.009 0.016 0.025 0.036 0.049 0.064 0.081 0.1
  F(t) = 0.0 0.001 0.005 0.014 0.030 0.055 0.091 0.140 0.204 0.285 0.385
- The last element of the scan (0.385) is the same as the output of reduce
- In mathematical terms,

$$\int_0^1 f(x)dx = F(1) - F(0)$$

Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

Scan: Calculating the Fibonacci Sequence by Matrix Multiplication

- The reduce and scan operators can also work with a user defined type
- The Fibonacci Sequence is defined as

$$F_{n+1} = F_n + F_{n-1} \quad \text{with} \quad F_0 = 0, F_1 = 1$$

- By "unrolling" the recurrence we have

$$\begin{bmatrix} F_{n+1} \\ F_n \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F_n \\ F_{n-1} \end{bmatrix}$$

- Thus we can compute F_n by matrix multiplication

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \textcircled{1} & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \textcircled{2} & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \textcircled{3} & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} \textcircled{5} & 3 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} \textcircled{8} & 5 \\ 5 & 3 \end{bmatrix}$$

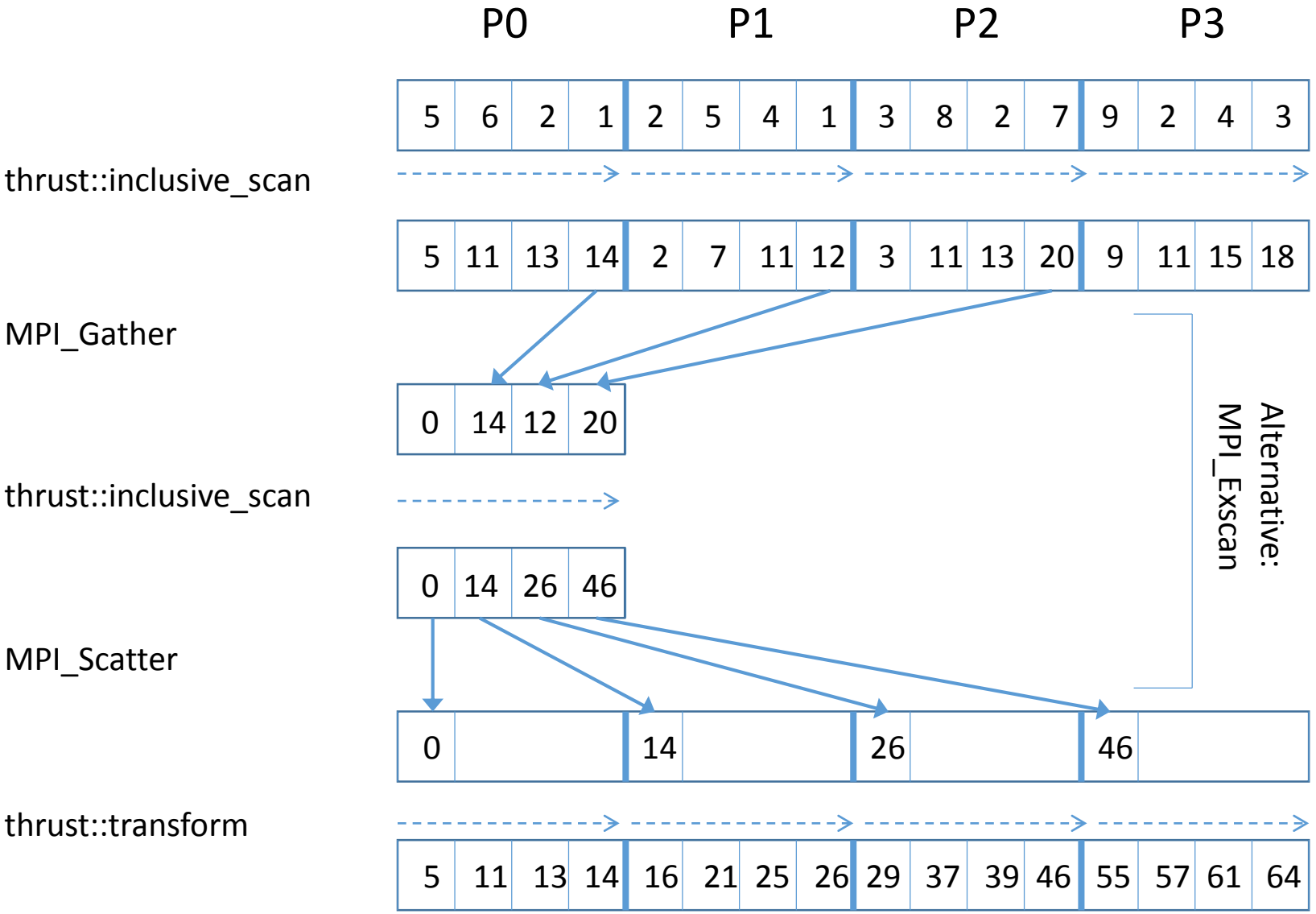
Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

Fibonacci Sequence using a Matrix Scan

```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/scan.h>
#include <thrust/copy.h>
#include <thrust/iterator/transform_iterator.h>
#include <thrust/iterator/constant_iterator.h>

// A simple data structure for 2x2 matrix
template <typename T>
struct mat22 {
    T m00, m10, m01, m11;

    __host__ __device__
    mat22() {}
    __host__ __device__
    mat22(T a, T b, T c, T d) : m00(a), m10(b), m01(c), m11(d) {}
};

// Overload the multiplication operator for 2x2 matrices
template <typename T>
__host__ __device__
mat22<T>
operator*(const mat22<T> &lhs, const mat22<T> &rhs)
{
    return mat22<T>(
            lhs.m00*rhs.m00+lhs.m01*rhs.m10,
            lhs.m10*rhs.m00+lhs.m11*rhs.m10,
            lhs.m00*rhs.m01+lhs.m01*rhs.m11,
            lhs.m10*rhs.m01+lhs.m11*rhs.m11);
}

// Wrap the overloaded multiplication operator into a binary functor
template <typename T>
struct matmul : public thrust::binary_function<mat22<T>, mat22<T>, mat22<T> > {
    __host__ __device__
    mat22<T> operator()(const mat22<T> &lhs, const mat22<T> &rhs) {
        return lhs*rhs;
    }
};

// Extract the (0, 0) element from the 2x2 matrix
template <typename T>
struct zeroth_elem : public thrust::unary_function<mat22<T>, T> {
    __host__ __device__
    T operator()(const mat22<T> &mat) {
        return mat.m00;
    }
};

int main(void)
{
    const int N = 20;

    mat22<unsigned long> A(1, 1, 1, 0);

    thrust::constant_iterator<mat22<unsigned long> > begin(A);
    thrust::device_vector<mat22<unsigned long> > vect(N);

    // we multiply a bunch of matrix A together.
    thrust::inclusive_scan(begin, begin + N,
                           vect.begin(),
                           matmul<unsigned long>());

    // extract the (0,0) element from the matrices
    thrust::device_vector<unsigned int> fib(N);
    thrust::transform(vect.begin(), vect.end(),
                      fib.begin(),
                      zeroth_elem<unsigned long>());

    thrust::copy(fib.begin(), fib.end(),
                 std::ostream_iterator<unsigned long>(std::cout, " "));
    std::cout << std::endl;
}
```

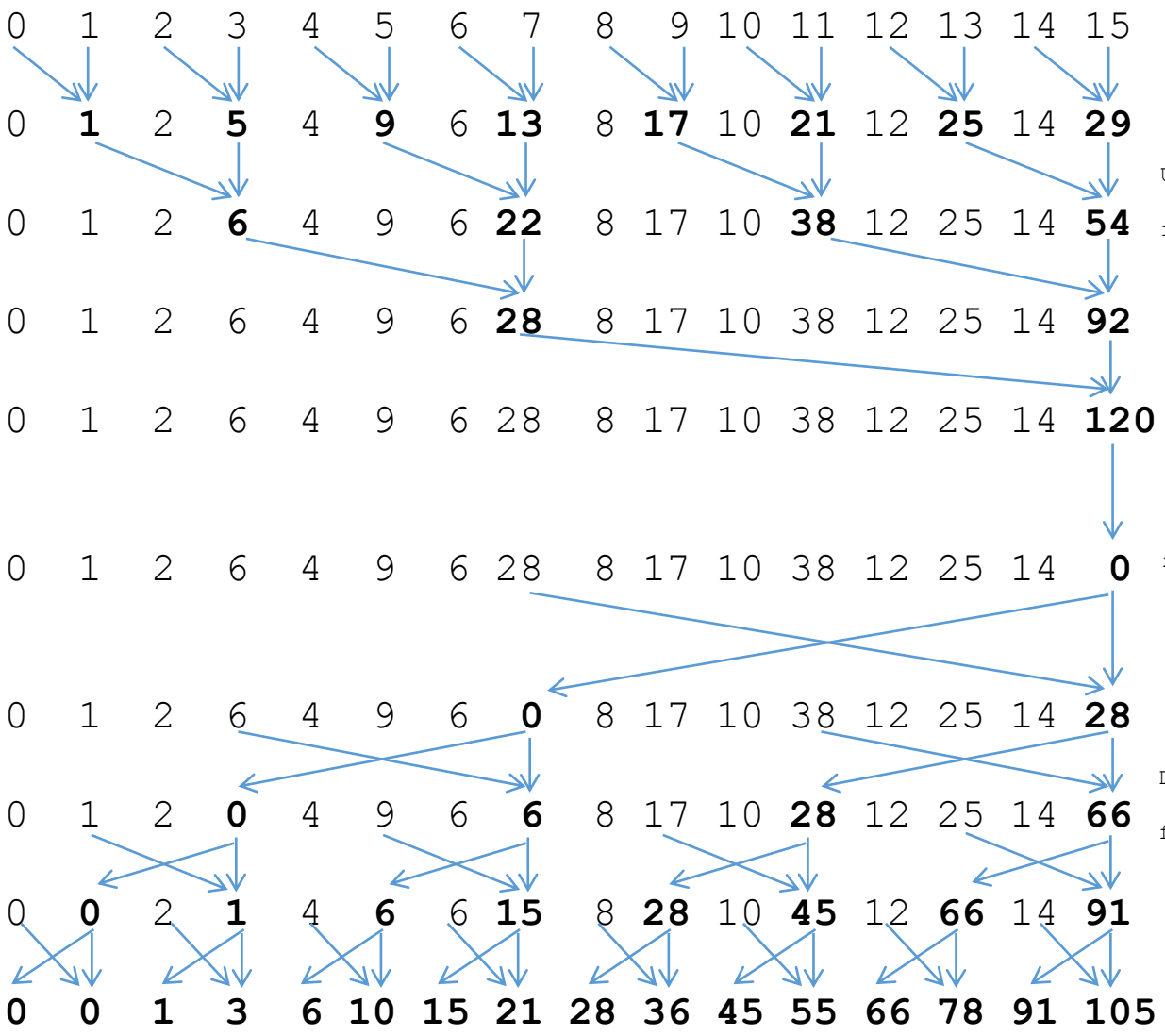
Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

Distributed Scan Algorithm



Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

Tree Scan Algorithm



Up-sweep (reduce)

```
for d = 0 to log2n-1 do
  for k from 0 to n-1 by 2^(d+1) pardo
    x[k+2^(d+1)-1] = x[k+2^d-1] + x[k+2^(d+1)-1]
```

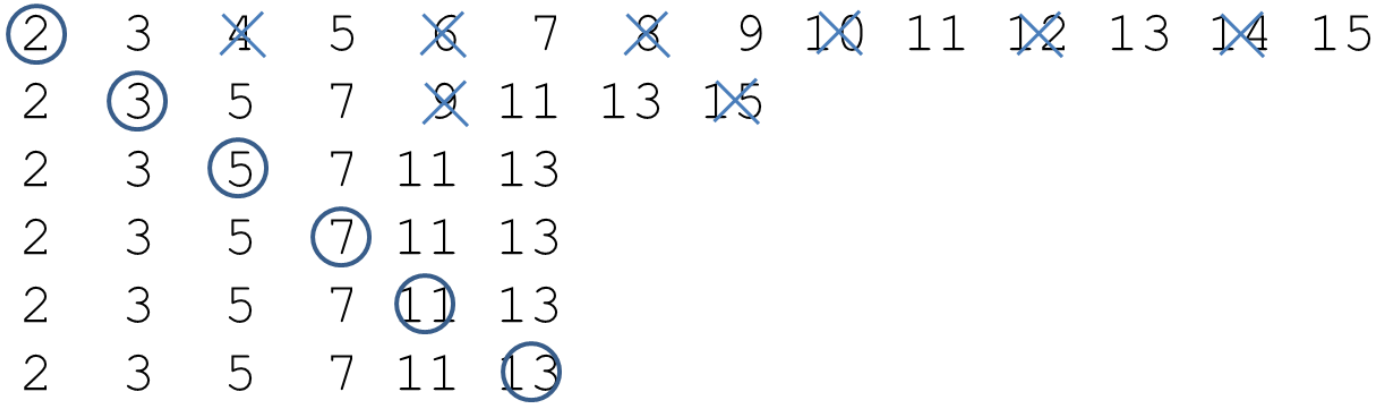
```
x[n-1] = 0
```

Down-sweep

```
for d = log2n-1 down to 0 do
  for k from 0 to n-1 by 2^(d+1) pardo
    t = x[k+2^d-1]
    x[k+2^d-1] = x[k+2^(d+1)-1]
    x[k+2^(d+1)-1] = t + x[k+2^(d+1)-1]
```

Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

Compaction: Finding Prime Numbers Using the Sieve of Eratosthenes

- Start with a vector containing the sequence of integers from 2 to N
- The first element in this vector is prime
- Use compaction to copy only elements of the vector not divisible by this prime into an updated vector (Thrust copy_if operator)
- The second element in this vector is prime
- Repeat the two steps above until you reach the end of the vector
- 

Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

Kokkos from Sandia (Carter Edwards)

- Parallel dispatch
  - parallel_for({#teams, #threads/team}, functor)
  - parallel_reduce({#teams, #threads/team}, functor)
  - parallel_scan({#teams, #threads/team}, functor)
- Multidimensional array layout
  - Leading dimension (right most) is parallel work dimension
  - Choose array layout for required access pattern (e.g., AoS vs. SoA) transparently to application
  - Provides views of data
  - Subject of separate proposal
- Like Thrust and the Parallelism TS, for node-level only (not distributed memory)

```
// Reduction functor for computing the sum of squares.
//
// More advanced reduction examples will show how to control the
// reduction's "join" operator.  If the join operator is not provided,
// it defaults to binary operator+ (adding numbers together).
struct squaresum {
  // Specify the type of the reduction value with a "value_type"
  // typedef.  In this case, the reduction value has type int.
  typedef int value_type;

  // The reduction functor's operator() looks a little different than
  // the parallel_for functor's operator().  For the reduction, we
  // pass in both the loop index i, and the intermediate reduction
  // value lsum.  The latter MUST be passed in by nonconst reference.
  // (If the reduction type is an array like int[], indicating an
  // array reduction result, then the second argument is just int[].)
  KOKKOS_INLINE_FUNCTION
  void operator () (const int i, int& lsum) const {
    lsum += i*i; // compute the sum of squares
  }
};

int main (int argc, char* argv[]) {
  Kokkos::initialize (argc, argv);
  const int n = 10;

  // Compute the sum of squares of integers from 0 to n-1, in
  // parallel, using Kokkos.
  int sum = 0;
  Kokkos::parallel_reduce (n, squaresum (), sum);
  printf ("Sum of squares of integers from 0 to %i, "
          "computed in parallel, is %i\n", n - 1, sum);

  // Compare to a sequential loop.
  int seqSum = 0;
  for (int i = 0; i < n; ++i) {
    seqSum += i*i;
  }
  printf ("Sum of squares of integers from 0 to %i, "
          "computed sequentially, is %i\n", n - 1, seqSum);
  Kokkos::finalize ();
  return (sum == seqSum) ? 0 : -1;
}
```

Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

C++17 Parallel Algorithms Library

- "An object of an execution policy type indicates the kinds of parallelism allowed in the execution of an algorithm and expresses the consequent requirements on the element access functions."
- Classes define types used to disambiguate parallel algorithm overloading
  - Execution may not be parallelized
    class sequential_execution_policy { };
    constexpr sequential_execution_policy seq{};
  - Execution may be parallelized
    class parallel_execution_policy { };
    constexpr parallel_execution_policy par{};
  - Execution may be vectorized and parallelized
    class parallel_vector_execution_policy { };
    constexpr parallel_vector_execution_policy par_vec{};

Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

C++17: Execution Policy Example

```
std::vector<int> v = ...

// standard sequential sort
std::sort(v.begin(), v.end());

using namespace std::experimental::parallel;

// explicitly sequential sort
sort(seq, v.begin(), v.end());

// permitting parallel execution
sort(par, v.begin(), v.end());

// permitting vectorization as well
sort(par_vec, v.begin(), v.end());

// sort with dynamically-selected execution
size_t threshold = ...
execution_policy exec = seq;
if (v.size() > threshold)
{
  exec = par;
}

sort(exec, v.begin(), v.end());
```

Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

C++17: Parallel Algorithms

- “Parallel algorithms have template parameters named ExecutionPolicy which describe the manner in which the execution of these algorithms may be parallelized and the manner in which they apply the element access functions.”
- Execution policies
  - sequential_execution_policy: “sequential order in the calling thread”
  - parallel_execution_policy: “unordered fashion in either the invoking thread or in a thread implicitly created by the library…any such invocations executing in the same thread are indeterminately sequenced with respect to each other”
  - parallel_vector_execution_policy: “unordered fashion in unspecified threads, and unsequenced with respect to one another within each thread (…may be interleaved on a single thread)”

Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

C++17: Table of Parallel Algorithms

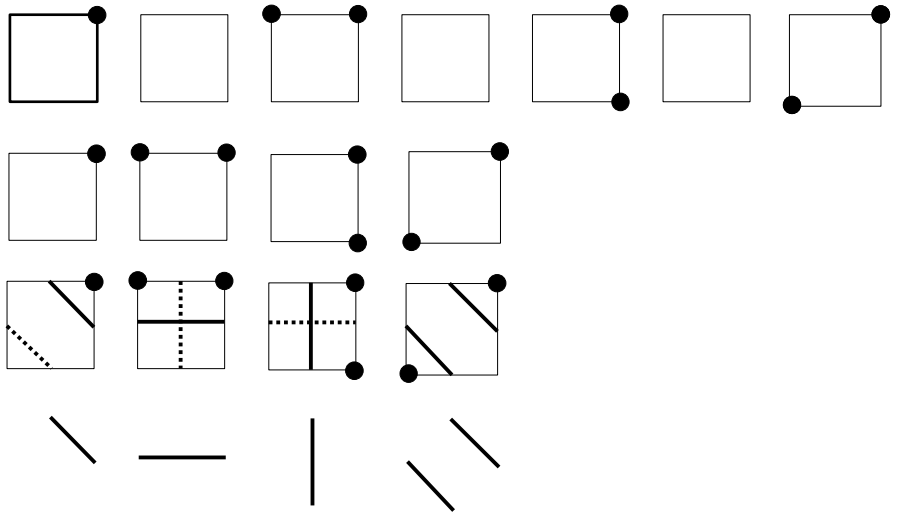
- "The Parallel Algorithms Library provides overloads for each of the algorithms named in [the table below], corresponding to the algorithms with the same name in the C++ Standard Algorithms Library"
- "…the overloads shall have an additional template type parameter named ExecutionPolicy"

adjacent_difference	adjacent_find	all_of	any_of
copy	copy_if	copy_n	count
count_if	equal	exclusive_scan	fill
fill_n	find	find_end	find_first_of
find_if	find_if_not	for_each	for_each_n
generate	generate_n	includes	inclusive_scan
inner_product	inplace_merge	is_heap	is_heap_until
is_partitioned	is_sorted	is_sorted_until	lexicographical_compare
max_element	merge	min_element	minmax_element
mismatch	move	none_of	nth_element
partial_sort	partial_sort_copy	partition	partition_copy
reduce	remove	remove_copy	remove_copy_if
remove_if	replace	replace_copy	replace_copy_if
replace_if	reverse	reverse_copy	rotate
rotate_copy	search	search_n	set_difference
set_intersection	set_symmetric_difference	set_union	sort
stable_partition	stable_sort	swap_ranges	transform
transform_exclusive_scan	transform_inclusive_scan	transform_reduce	uninitialized_copy
uninitialized_copy_n	uninitialized_fill	uninitialized_fill_n	unique
unique_copy			

Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

Isosurface with Marching Cube – the Naive Way

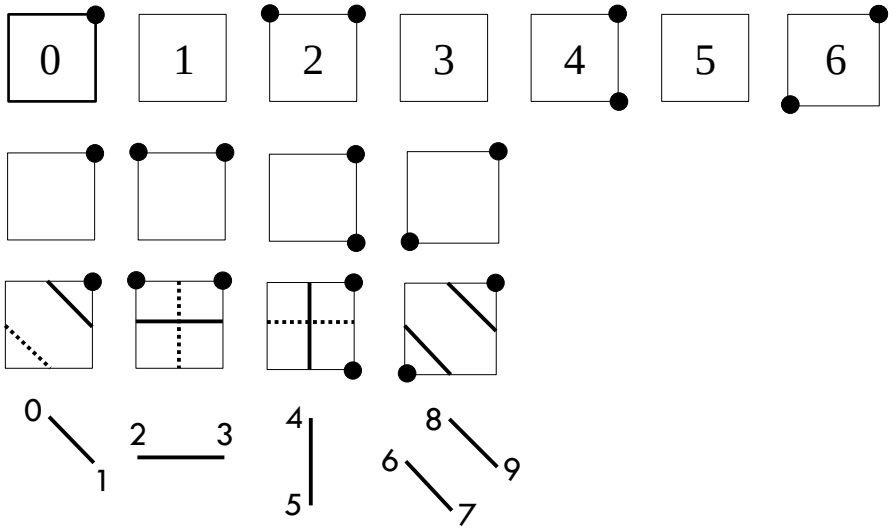
- Classify all cells by *transform*
- Use *copy_if* to compact valid cells.
- For each valid cell, generate same number of geometries with flags.
- Use *copy_if* to do stream compaction on vertices.
- This approach is too slow, more than 50% of time was spent moving huge amount of data in global memory.
- Can we avoid calling *copy_if* and eliminate global memory movement?



Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

Isosurface with Marching Cube – Optimization

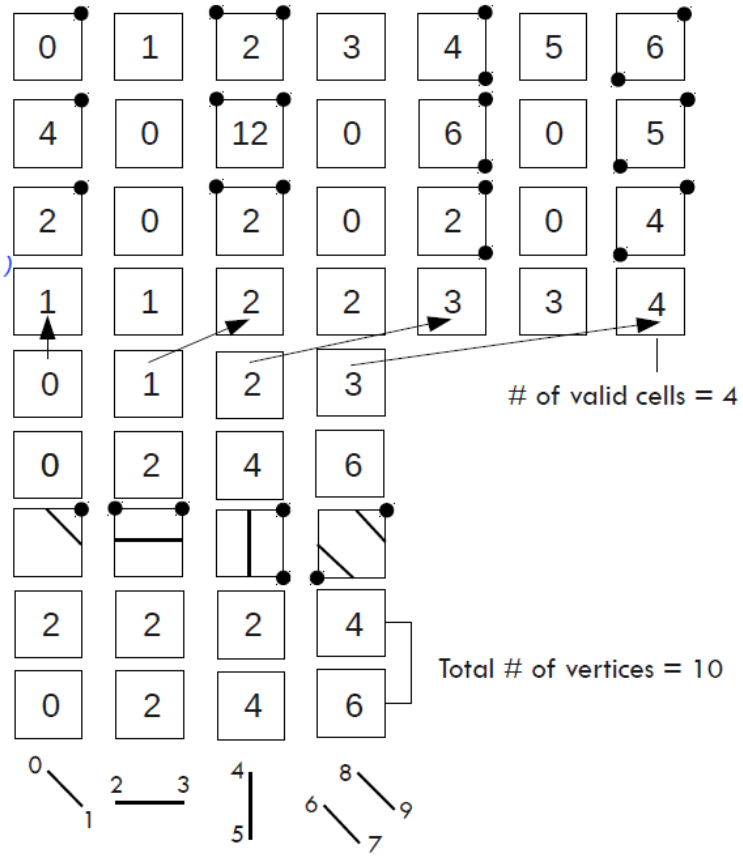
- Inspired by HistoPyramid
- The filter is essentially a mapping from input cell id to output vertex id
- Is there a "reverse" mapping?
- If there is a reverse mapping, the filter can be very "lazy"
- Given an output vertex id, we *only* apply operations on the cell that would generate the vertex
- Actually for a range of output vertex ids



Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

Isosurface with Marching Cubes Algorithm

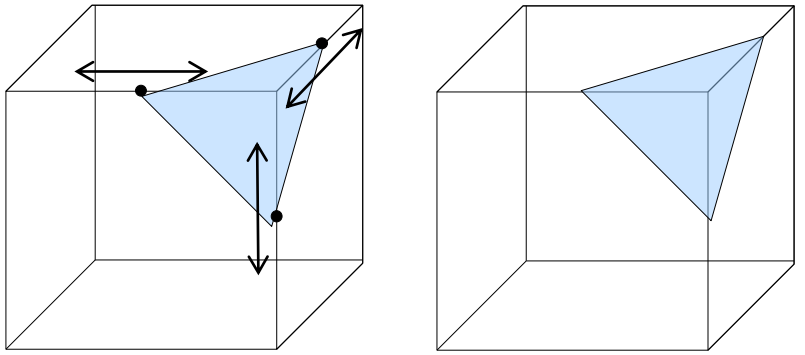
1. input
   transform(classify_cell)
2. caseNums
3. numVertices
   transform_inclusive_scan(is_valid_cell)
4. validCellEnum
5. CountingIterator
   upper_bound
6. validCellIndices
   make_permutation_iterator
7. numVerticesCompacted
   exclusive_scan
8. numVerticesEnum
   for_each(isosurface_functor)
9. outputVertices

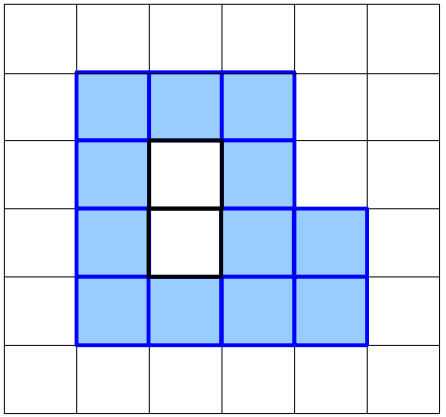


Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

Variations on Isosurface: Cut Surfaces and Threshold

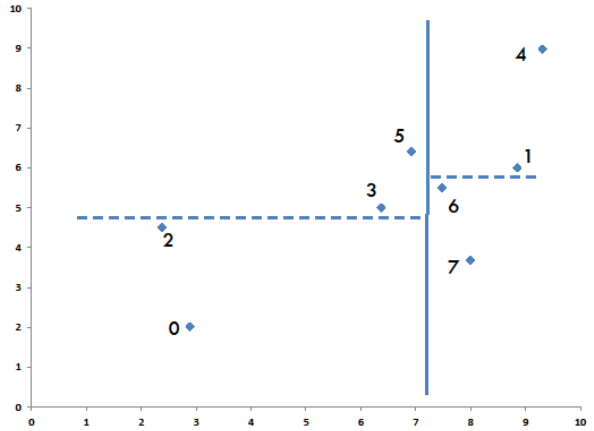
- Cut surface
  - Two scalar fields, one for generating geometry (cut surface) the other for scalar interpolation
  - Less than 10 LOC change, negligible performance impact to isosurface
  - One 1D interpolation per triangle vertex
- Threshold
  - Classify cells, this time based on whether value at each vertex falls within threshold range, then stream compact valid cells and generate geometry for valid cells
  - Additional pass of cell classification and stream compaction to remove interior cells





Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

Data-Parallel KD Tree



		Point Ids								X Ranks								Y Ranks							
Init	computeGlobalRanks	0	1	2	3	4	5	6	7	1	6	0	2	7	3	4	5	0	5	2	3	7	6	4	1
Level 1	computeFlags(X)	F	T	F	F	T	F	T	T	**F**	**T**	**F**	**F**	**T**	**F**	**T**	**T**	F	T	F	F	T	F	T	T
	segmentedSplit	0	2	3	5	1	4	6	7	1	0	2	3	6	7	4	5	0	2	3	6	5	7	4	1
	flags	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T
	segmentIds	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	renumberRanks									1	0	2	3	2	3	0	1	0	1	2	3	2	3	1	0
Level 2	computeFlags(Y)	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	**F**	**F**	**T**	**T**	**T**	**T**	**F**	**F**
	segmentedSplit	0	2	3	5	6	7	1	4	1	0	2	3	0	1	2	3	0	1	2	3	1	0	2	3
	flags	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T
	segmentIds	0	0	1	1	2	2	3	3	0	0	1	1	2	2	3	3	0	0	1	1	2	2	3	3
	renumberRanks									1	0	0	1	0	1	0	1	0	1	0	1	1	0	0	1

Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

Additional Operators

Blelloch's "Vector Models for Data-Parallel Computing"

Data Structures
- Graphs: Neighbor reducing, distributing excess across edges
- Trees: Leaffix and rootfix operations, tree manipulations
- Multidimensional arrays

Computational Geometry
- Generalized binary search
- k-D tree
- Closest pair
- Quickhull
- Merge Hull

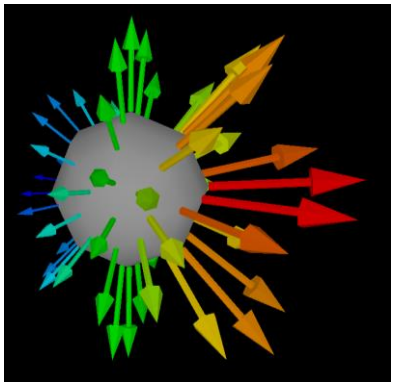
Graph Algorithms
- Minimum spanning tree
- Maximum flow
- Maximal independent set

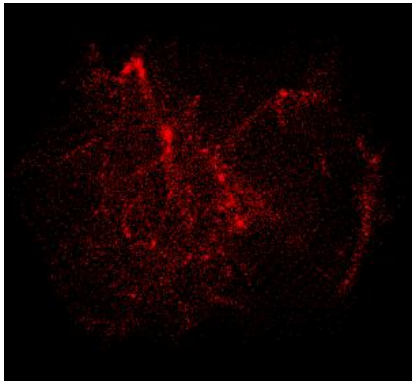
Numerical Algorithms
- Matrix-vector multiplication
- Linear-systems solver
- Simplex
- Outer product
- Sparse-matrix multiplication

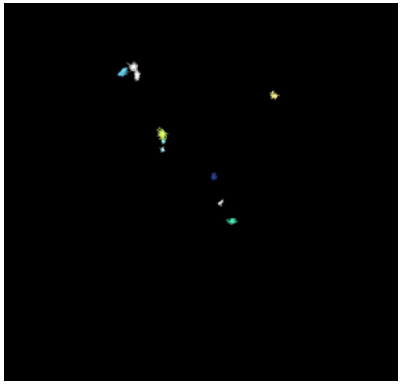
http://www.cs.cmu.edu/~blelloch/papers/Ble90.pdf

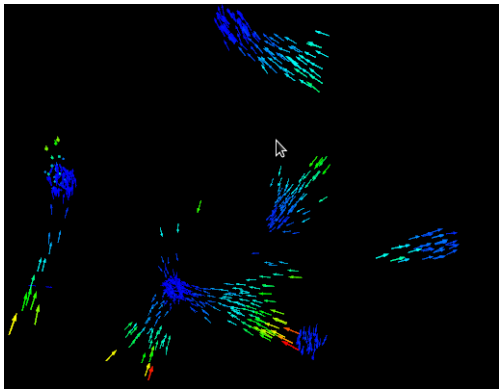
Our implementations

- Glyphs
- KD-Tree Construction
- Halo finder for cosmology simulations
- "Boid" simulation (flocking birds)





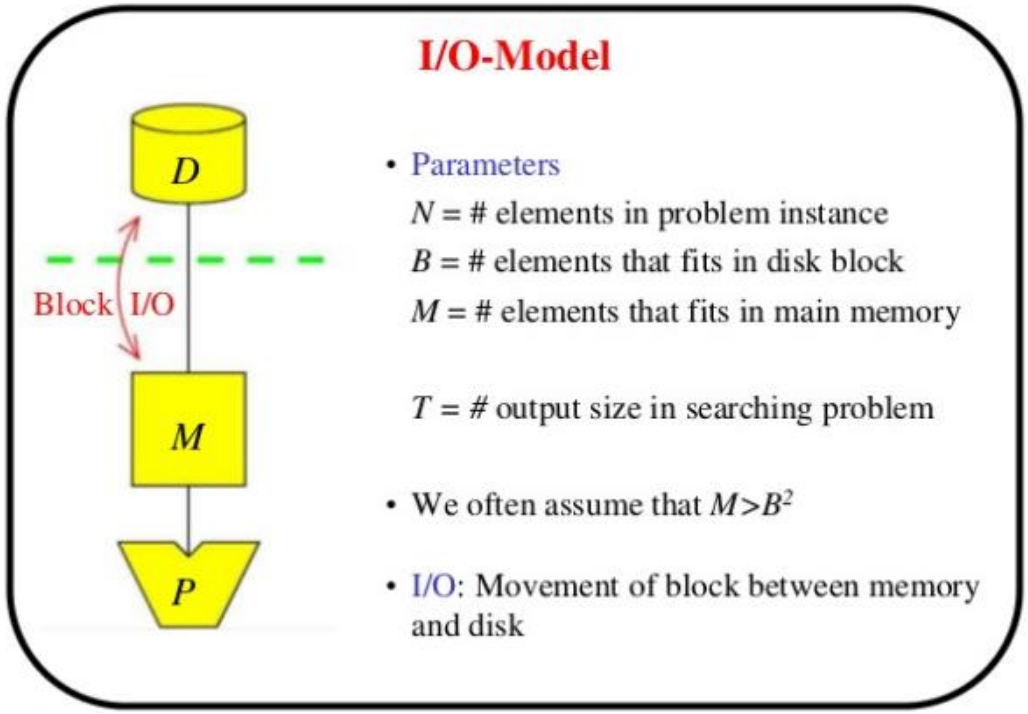




Chris Sewell, Guest Lectures at the University of Oregon 2014, and at Programming Models Workshop 2016
https://datascience.lanl.gov/data/Presentation2014Oregon.pdf

External Memory (Streaming or Out-of-Core) Algorithms

- When all the data does not fit in memory at once, or if I/O is very expensive, algorithms may be designed to minimize the number of I/Os rather than the number of flops

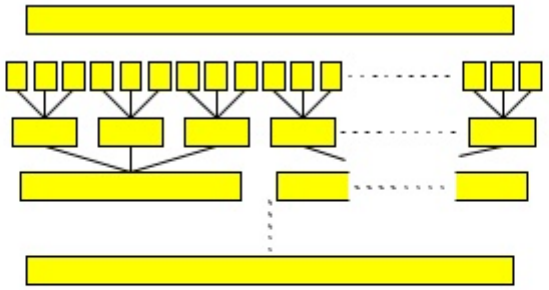


Lars Arge, Presentation at the University of Aarhus, 2013:
http://www.slideshare.net/ktoshik/io-efficient-algorithms-and-data-structures-34-lecture-by-lars-arge

External Sort

- Merge sort:
  - Create *N/M* memory sized sorted runs
  - Merge runs together *M/B* at a time

$\Rightarrow O(\log_{M/B} \frac{N}{M})$ phases using $O(N/B)$ I/Os each



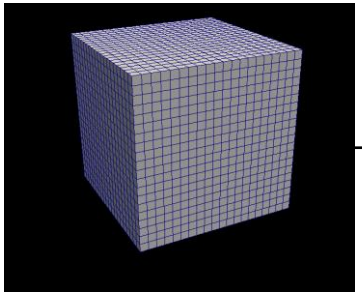
- Distribution sort similar (but harder – split elements)

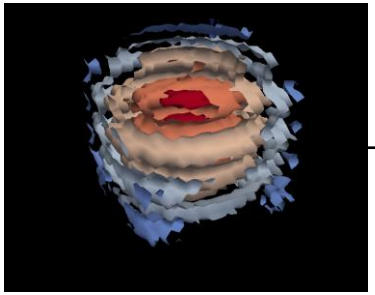
Lars Arge, Presentation at the University of Aarhus, 2013:
http://www.slideshare.net/ktoshik/io-efficient-algorithms-and-data-structures-34-lecture-by-lars-arge

VTK-m

Accelerating VTK for Multi-core/Many-core

VTK-m Framework





Control Environment

Grid Topology
Array Handle
Invoke

Device Adapter

Allocate
Transfer
Schedule
Sort
…

Execution Environment

Cell Operations
Field Operations
Basic Math
Make Cells

Worklet

Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

Simulations

In Situ Vis Library
(Integration with Sim)

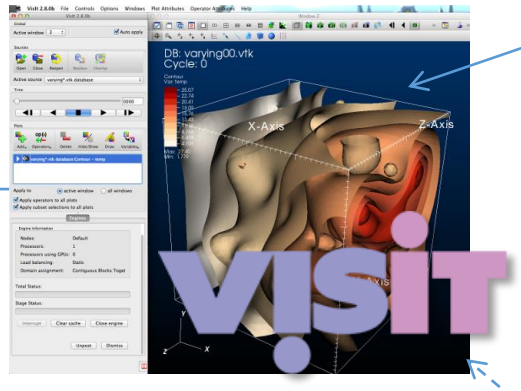
ParaView Catalyst

GUI / Parallel Management

Base Vis Library
(Algorithm Implementation)

VTK

Libsim



Multithreaded Algorithms
Processor Portability

VTK-m

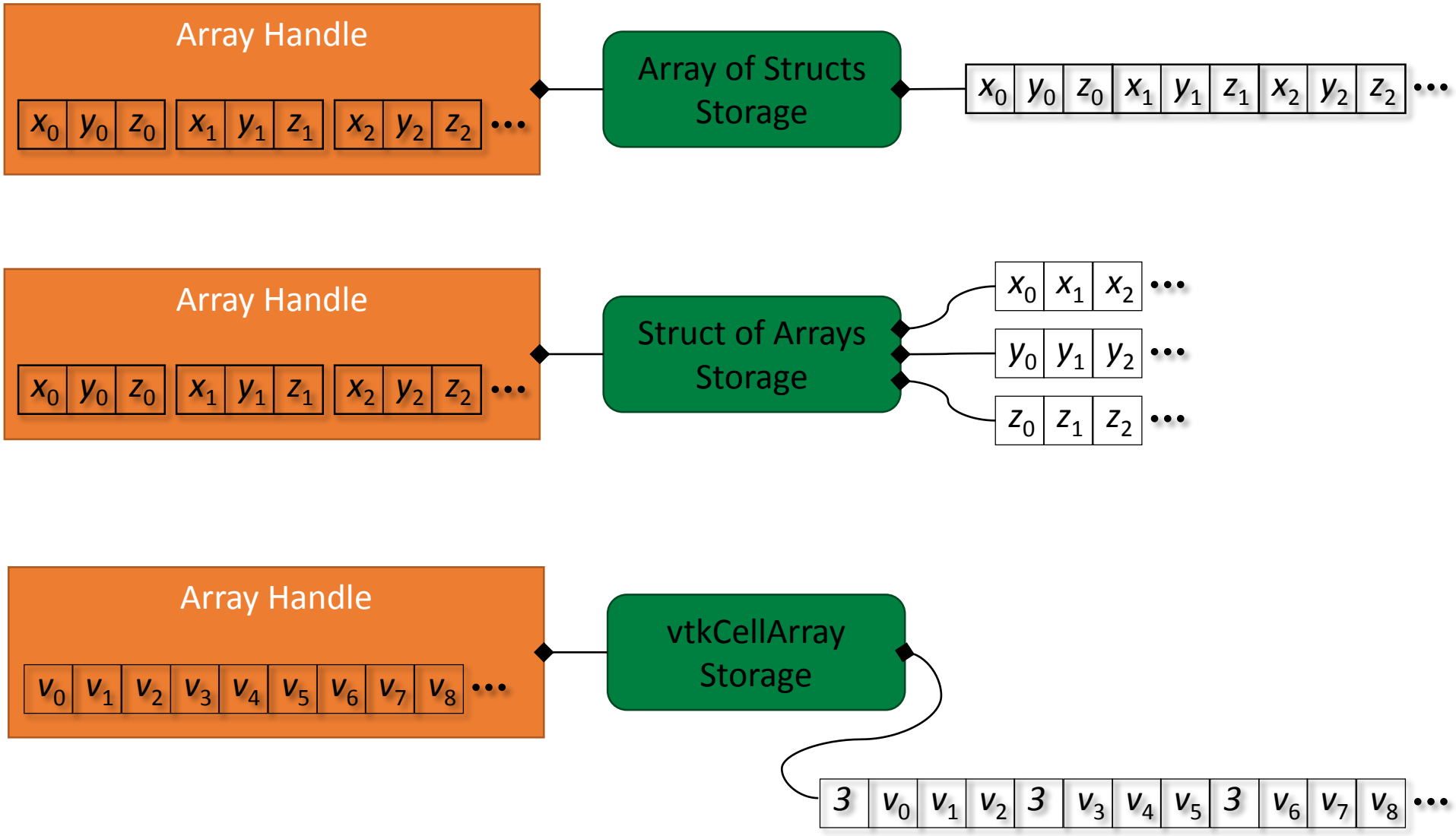
Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

ArrayHandle

- `vtkm::cont::ArrayHandle<type>` manages an "array" of data
  - Acts like a reference-counted smart pointer to an array
  - Manages transfer of data between control and execution
  - Can allocate data for output
- Relevant methods
  - `GetNumberOfValues()`
  - `GetPortalConstControl()`
  - `ReleaseResources()`, `ReleaseResourcesExecution()`
- Functions to create an ArrayHandle
  - `vtkm::cont::make_ArrayHandle(const T*array,vtkm::Id size)`
  - `vtkm::cont::make_ArrayHandle(const std::vector<T>&vector)`
  - Both of these do a shallow (reference) copy.
    - Do not let the original array be deleted or vector to go out of scope!

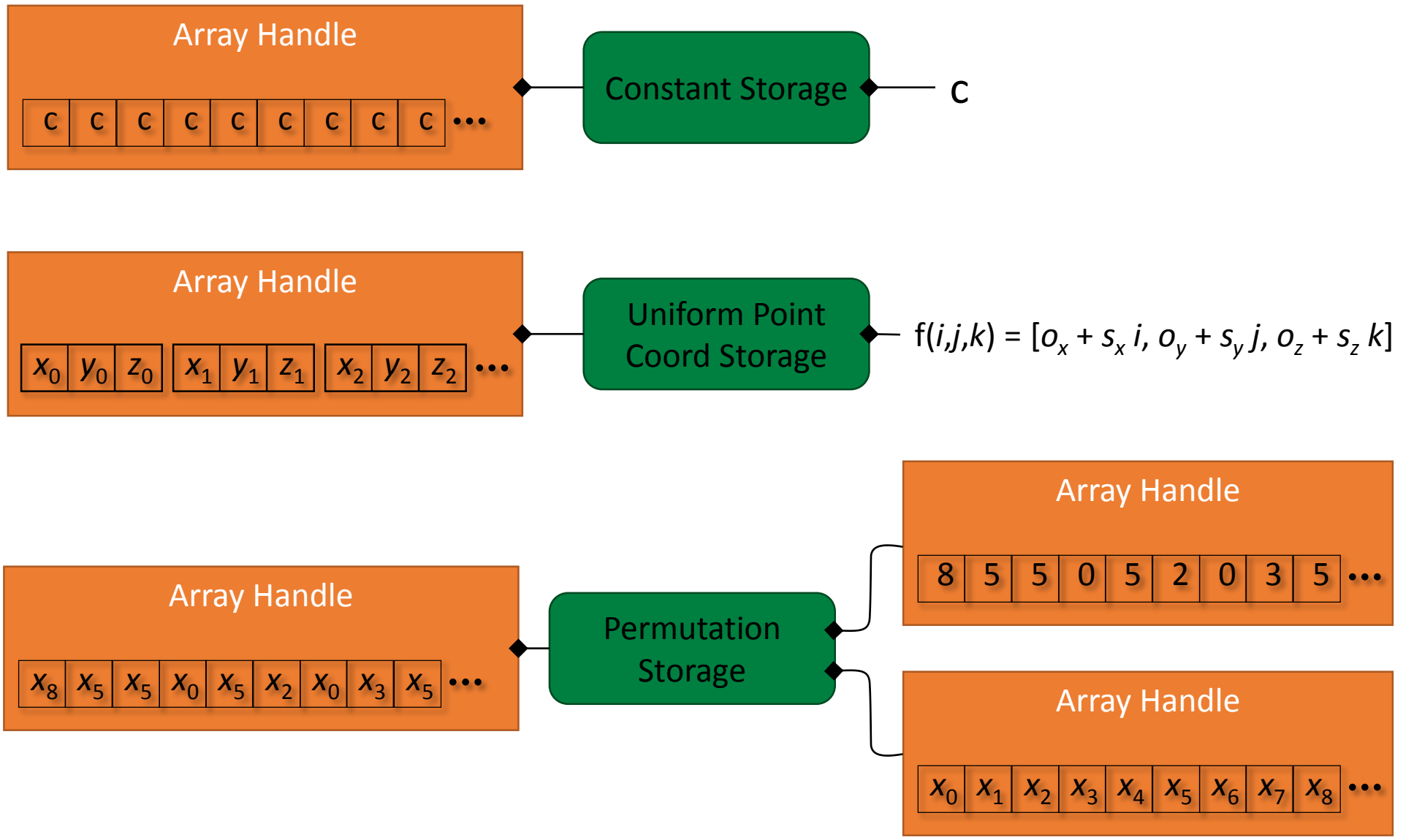
Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

Array Handle Storage



Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

Fancy Array Handles



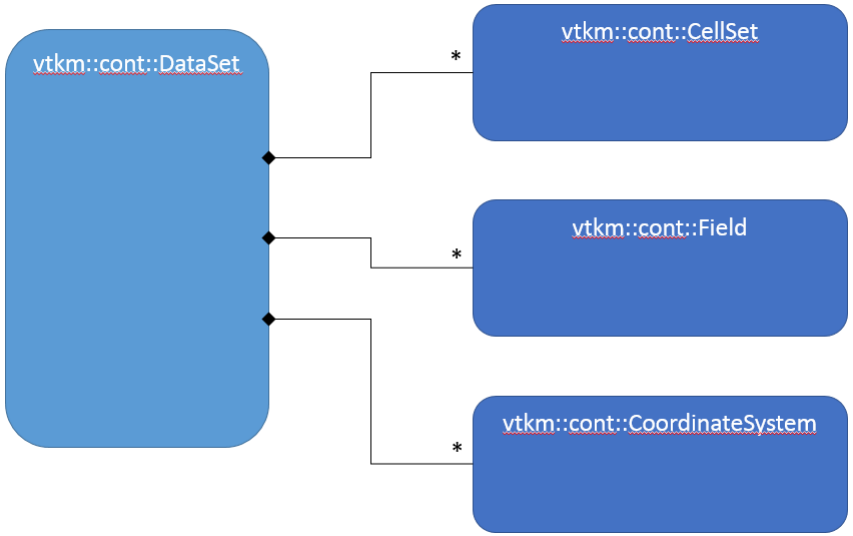
Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

DynamicArrayHandle

- DynamicArrayHandle is a magic untyped reference to an ArrayHandle
- Statically holds a list of potential types and storages the contained array might have
  - Can be changed with ResetTypeList and ResetStorageList
  - Changing these lists requires creating a new object
- Parts of VTK-m will automatically staticly cast a DynamicArrayHandle as necessary
  - Requires the actual type to be in the list of potential types

Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

A DataSet Has



- 1 or more CellSet
  - Defines the connectivity of the cells
  - Examples include a regular grid of cells or explicit connection indices
- 0 or more Field
  - Holds an ArrayHandle containing field values
  - Field also has metadata such as the name, the topology association (point, cell, face, etc), and which cell set the field is attached to
- 0 or more CoordinateSystem
  - Really just a Field with a special meaning
  - Contains helpful features specific to common coordinate systems

Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

Worklet Types

- `WorkletMapField`: Applies worklet on each value in an array.
- `WorkletMapTopology`: Takes from and to topology elements (e.g. point to cell or cell to point). Applies worklet on each “to” element. Worklet can access field data from both “from” and “to” elements. Can output to “to” elements.
- Many more to come…

Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

```
struct Sine: public vtkm::worklet::WorkletMapField {
  typedef void ControlSignature(FieldIn<>, FieldOut<>);
  typedef _2 ExecutionSignature(_1);

  template<typename T>
  VTKM_EXEC_EXPORT
  T operator()(T x) const {
    return vtkm::Sin(x);
  }
};
```

Execution Environment

Control Environment

```
vtkm::cont::ArrayHandle<vtkm::Float32> inputHandle =
    vtkm::cont::make_ArrayHandle(input);
vtkm::cont::ArrayHandle<vtkm::Float32> sineResult;

vtkm::worklet::DispatcherMapField<Sine> dispatcher;
dispatcher.Invoke(inputHandle, sineResult);
```

Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

Elements of a Worklet

1. Subclass of one of the base worklet types
2. Typedefs for `ControlSignature` and `ExecutionSignature`
3. A parenthesis operator
   1. Must have VTKM_EXEC_EXPORT
   2. Input parameters are by value or const reference
   3. Output parameters are by reference
   4. The method must be declared const

```
struct ImagToPolar: public vtkm::worklet::WorkletMapField {
  typedef void ControlSignature(FieldIn<vtkm::TypeListTagScalar>,
                                FieldIn<vtkm::TypeListTagScalar>,
                                FieldOut<vtkm::TypeListTagScalar>,
                                FieldOut<vtkm::TypeListTagScalar>);
  typedef void ExecutionSignature(_1, _2, _3, _4);

  template<typename T1, typename T2, typename T3, typename T4>
  VTKM_EXEC_EXPORT
  void operator()(T1 real, T2 imaginary,
                  T3 &magnitude, T4 &phase) const
```

1

2

3.1

3.2

3.3

3.4

Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

Device Adapter Algorithms

- Implementations of data-parallel primitives
  - Copy
  - LowerBounds
  - Reduce
  - ReduceByKey
  - ScanInclusive
  - ScanExclusive
  - Sort
  - SortByKey
  - StreamCompact
  - Unique
  - UpperBounds

```
//2. now we need to do a sort by key, making duplicate ids be adjacent
Algorithm::SortByKey(uniqueIds, this->InterpolationWeights);

//3. lastly we need to do a unique by key, but since vtkm doesn't
// offer that feature, we use a zip handle.
// We need to use a custom comparison operator as we only want to compare
// the id2 which is the first entry in the zip pair
vtkm::cont::ArrayHandleZip<Id2HandleType, WeightHandleType> zipped =
            vtkm::cont::make_ArrayHandleZip(uniqueIds, this->InterpolationWeights);
Algorithm::Unique( zipped, FirstValueSame());
}
```

Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

Worklet Example: Cell Average

```
2  //  Copyright (c) Kitware, Inc.
20
21 #ifndef vtk_m_worklet_CellAverage_h
22 #define vtk_m_worklet_CellAverage_h
23
24 #include <vtkm/worklet/WorkletMapTopology.h>
25
26 namespace vtkm {
27 namespace worklet {
28
29 //simple functor that returns the average point value.
30 class CellAverage :
31         public vtkm::worklet::WorkletMapPointToCell
32 {
33 public:
34   typedef void ControlSignature(FieldInPoint<Scalar> inPoints,
35                                 TopologyIn topology,
36                                 FieldOutCell<Scalar> outCells);
37   typedef void ExecutionSignature(_1, FromCount, _3);
38   typedef _2 InputDomain;
39
40   template<typename PointValueVecType, typename OutType>
41   VTKM_EXEC_EXPORT
42   void operator()(const PointValueVecType &pointValues,
43                   const vtkm::IdComponent &numPoints,
44                   OutType &average) const
45   {
46     OutType sum = static_cast<OutType>(pointValues[0]);
47     for (vtkm::IdComponent pointIndex = 1; pointIndex < numPoints; ++pointIndex)
48       {
49       sum = sum + static_cast<OutType>(pointValues[pointIndex]);
50       }
51
52     average = sum / static_cast<OutType>(numPoints);
53   }
54
55 };
56
57 }
58 } // namespace vtkm::worklet
59
60 #endif // vtk_m_worklet_CellAverage_h
```

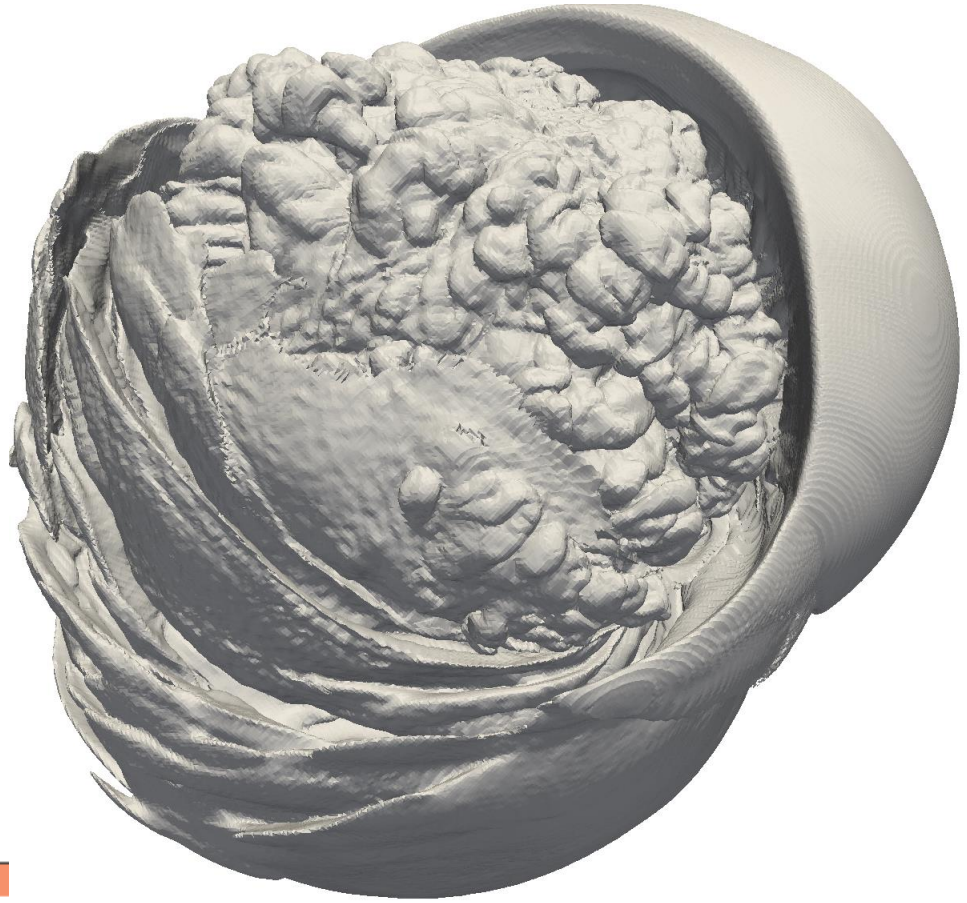
Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

Filter Example: Cell Average

```
2  //  Copyright (c) Kitware, Inc.
20 #include <vtkm/cont/DynamicCellSet.h>
21 #include <vtkm/worklet/DispatcherMapTopology.h>
22
23 namespace vtkm {
24 namespace filter {
25
26 //-----------------------------------------------------------------------------
27 CellAverage::CellAverage():
28   vtkm::filter::CellFilter<CellAverage>(),
29   Worklet()
30 {
31 }
32
33 //-----------------------------------------------------------------------------
34 template<typename T,
35          typename StorageType,
36          typename DerivedPolicy,
37          typename DeviceAdapter>
38 vtkm::filter::FieldResult CellAverage::DoExecute(const vtkm::cont::DataSet &input,
39                                                  const vtkm::cont::ArrayHandle<T, StorageType>& field,
40                                                  const vtkm::filter::FieldMetadata&,
41                                                  const vtkm::filter::PolicyBase<DerivedPolicy>&,
42                                                  const DeviceAdapter&)
43 {
44   vtkm::cont::DynamicCellSet cellSet =
45                   input.GetCellSet(this->GetActiveCellSetIndex());
46
47   //todo: we need to ask the policy what storage type we should be using
48   //If the input is implicit, we should know what to fall back to
49   vtkm::cont::ArrayHandle<T> outArray = field;
50
51   vtkm::worklet::DispatcherMapTopology<vtkm::worklet::CellAverage,
52                                        DeviceAdapter > dispatcher(this->Worklet);
53
54   //todo: we need to use the policy to determine the valid conversions
55   //that the dispatcher should do, including the result from GetCellSet
56   dispatcher.Invoke(field, cellSet, outArray);
57   vtkm::cont::Field outField(this->GetOutputFieldName(),
58                              vtkm::cont::Field::ASSOC_CELL_SET,
59                              cellSet.GetCellSet().GetName(),
60                              outArray);
61
62   return vtkm::filter::FieldResult(outField);
63 }
64 }
65 } // namespace vtkm::filter
```

Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

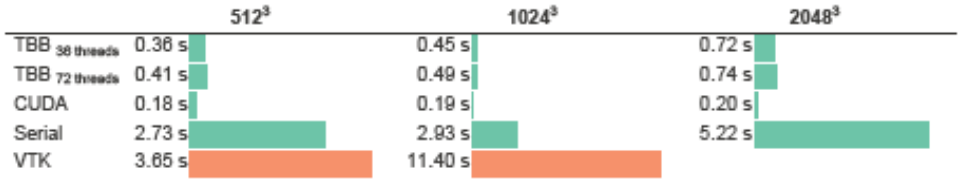
Isosurface



Algorithm	Device	Time
vtkMarchingCubes	Serial	11.917 s
vtkMarchingCubes	32 MPI Ranks	1.352 s
vtkMarchingCubes	64 MPI Ranks	1.922 s
PISTON	Serial	19.895 s
PISTON	CUDA	0.514 s
PISTON	TBB	0.955 s
VTK-m	Serial	20.784 s
VTK-m	CUDA	0.560 s
VTK-m	TBB	1.161 s

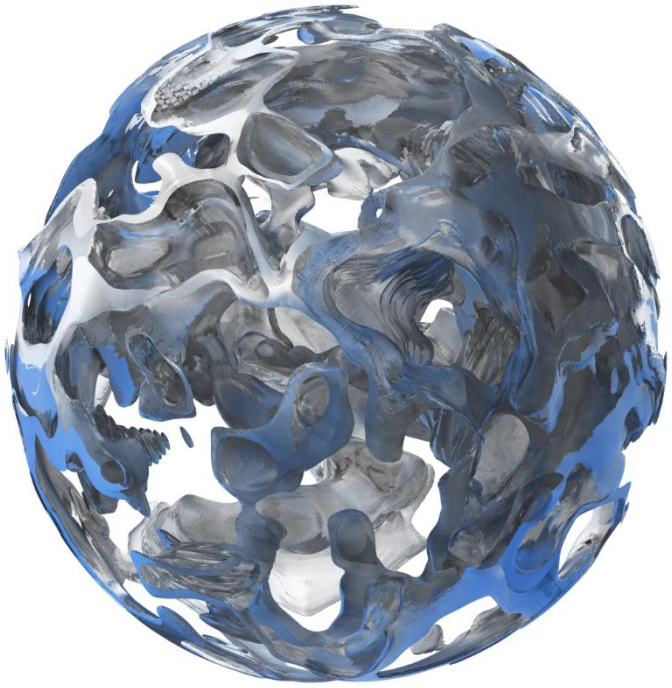
Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

Surface Simplification



Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

Ray Tracing

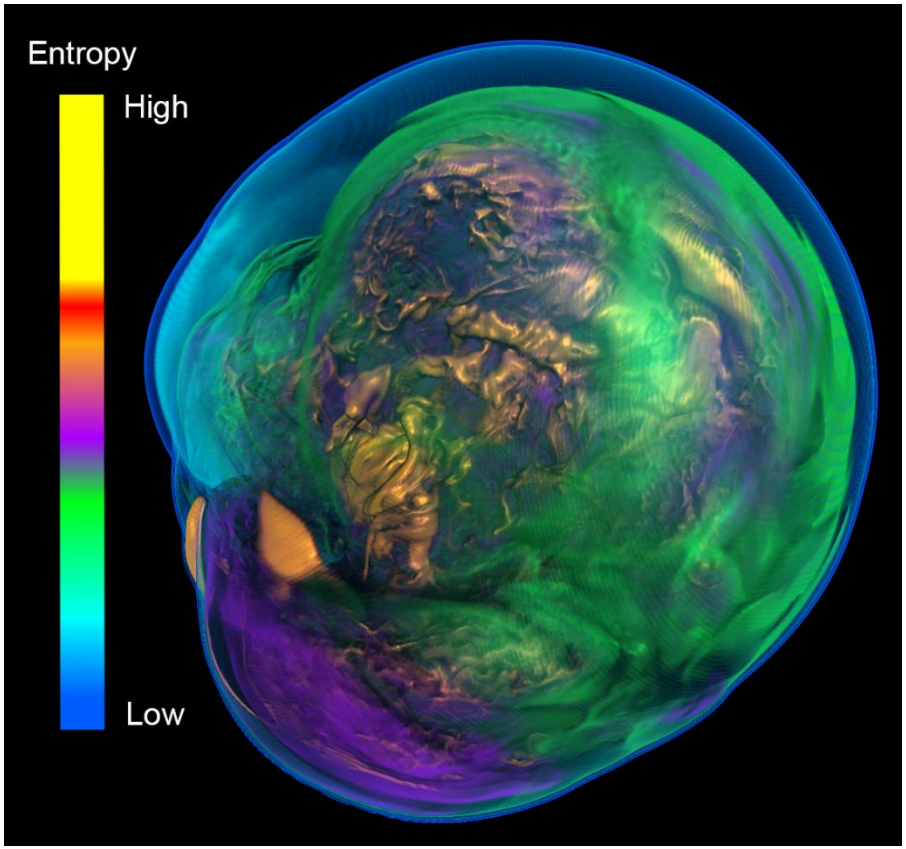


Data Set	Algorithm	Millions of Rays Per Second
LT_350K	OptiX Prime	357.6
	EAVL	150.8
	VTK-m	164.5
LT_372K	OptiX Prime	322.4
	EAVL	124.7
	VTK-m	140.8
RM_350K	OptiX Prime	436.5
	EAVL	197.5
	VTK-m	200.8
RM_650K	OptiX Prime	420.4
	EAVL	172.9
	VTK-m	166.0
RM_970K	OptiX Prime	347.1
	EAVL	152.8
	VTK-m	163.5
RM_1.7M	OptiX Prime	266.8
	EAVL	136.6
	VTK-m	148.8
RM_3.2M	OptiX Prime	264.5
	EAVL	124.8
	VTK-m	134.5
Seismic	OptiX Prime	267.8
	EAVL	106.3
	VTK-m	119.4

Data Set	Algorithm	Millions of Rays Per Second
LT_350K	Embree	51.9
	EAVL	27.7
	VTK-m	38.5
LT_372K	Embree	56.5
	EAVL	28.1
	VTK-m	36.0
RM_350K	Embree	64.8
	EAVL	33.3
	VTK-m	47.8
RM_650K	Embree	65.9
	EAVL	35.6
	VTK-m	49.1
RM_970K	Embree	59.1
	EAVL	29.3
	VTK-m	41.0
RM_1.7M	Embree	52.4
	EAVL	27.0
	VTK-m	37.8
RM_3.2M	Embree	48.4
	EAVL	28.3
	VTK-m	33.9
Seismic	Embree	43.2
	EAVL	25.2
	VTK-m	34.5

Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

Direct Volume Rendering

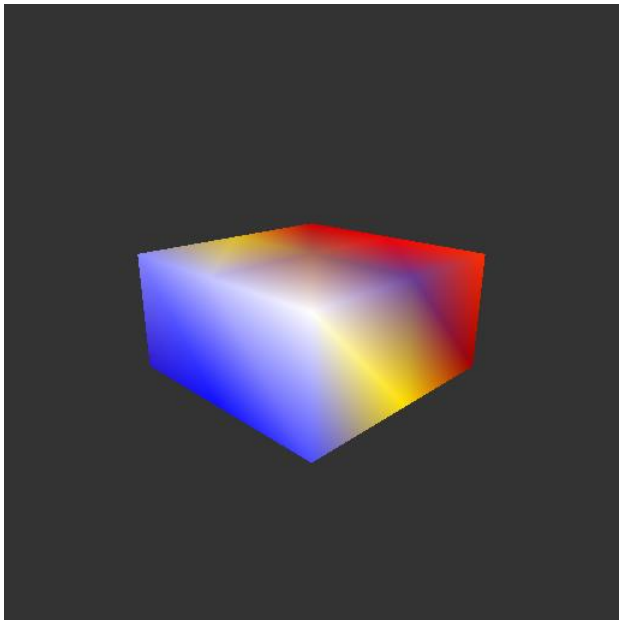


Device	Algorithm	Frames Per Second
CPU	VTK	2.08
	VTK-m	1.25
GPU	Dax	3.55
	VTK-m	6.79
MIC	Dax	0.07
	VTK-m	0.28

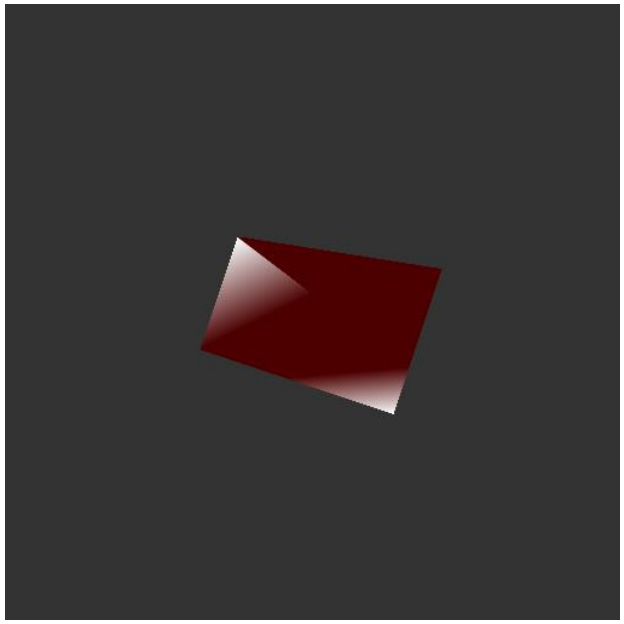
Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

Demo

- In vtk-m/examples/demo
- Reads specified VTK file or generates a default input uniform structured grid data set
- Uses VTK-m's rendering engine to render input data set to an image file using OS Mesa (or EGL, in development)
- Uses VTK-m's Marching Cubes filter to compute isosurface
- Renders output data set to another image file



Rendering of test input data



Rendering of test output data

Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

Demo Part 1: Reading Input

```
2  //  Copyright (c) Kitware, Inc.
20
21 #include <vtkm/cont/testing/MakeTestDataSet.h>
22 #include <vtkm/rendering/Window.h>
23 #include <vtkm/rendering/RenderSurface.h>
24 #include <vtkm/rendering/Scene.h>
25 #include <vtkm/rendering/Plot.h>
26 #include <vtkm/rendering/SceneRendererOSMesa.h>
27 #include <vtkm/cont/DeviceAdapter.h>
28 #include <vtkm/cont/testing/Testing.h>
29 #include <vtkm/io/reader/VTKDataSetReader.h>
30 #include <vtkm/filter/MarchingCubes.h>
31 #include <vtkm/cont/DataSetFieldAdd.h>
32
33 #include <iostream>
34
35 // This example reads an input vtk file specified on the command-line (or generates a default
36 // input data set if none is provided), uses VTK-m's rendering engine to render it to an
37 // output file using OS Mesa, instantiates an isosurface filter using VTK-m's filter
38 // mechanism, computes an isosurface on the input data set, packages the output of the filter
39 // in a new data set, and renders this output data set in a separate iamge file, again
40 // using VTK-m's rendering engine with OS Mesa.
41
42 int main(int argc, char* argv[])
43 {
44   // Input variable declarations
45   vtkm::cont::DataSet inputData;
46   vtkm::Float32 isovalue;
47   std::string fieldName;
48
49   // Get input data from specified file, or generate test data set
50   if (argc < 3)
51   {
52     vtkm::cont::testing::MakeTestDataSet maker;
53     inputData = maker.Make3DUniformDataSet0();
54     isovalue = 100.0f;
55     fieldName = "pointvar";
56   }
57   else
58   {
59     std::cout << "using: " << argv[1] << " as MarchingCubes input file" << std::endl;
60     vtkm::io::reader::VTKDataSetReader reader(argv[1]);
61     inputData = reader.ReadDataSet();
62     isovalue = atof(argv[2]);
63     fieldName = "SCALARS:pointvar";
64   }
```

Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

Demo Part 2: Rendering Data Set

```
66  typedef vtkm::rendering::SceneRendererOSMesa< > SceneRenderer;
67  typedef vtkm::rendering::RenderSurfaceOSMesa RenderSurface;
68
69  // Set up a view for rendering the input data
70  const vtkm::cont::CoordinateSystem coords = inputData.GetCoordinateSystem();
71  SceneRenderer sceneRenderer;
72  vtkm::rendering::View3D &view = sceneRenderer.GetView();
73  vtkm::Float64 coordsBounds[6];
74  coords.GetBounds(coordsBounds,VTKM_DEFAULT_DEVICE_ADAPTER_TAG());
75  vtkm::Vec<vtkm::Float32,3> totalExtent;
76  totalExtent[0] = vtkm::Float32(coordsBounds[1] - coordsBounds[0]);
77  totalExtent[1] = vtkm::Float32(coordsBounds[3] - coordsBounds[2]);
78  totalExtent[2] = vtkm::Float32(coordsBounds[5] - coordsBounds[4]);
79  vtkm::Float32 mag = vtkm::Magnitude(totalExtent);
80  vtkm::Normalize(totalExtent);
81  view.LookAt = totalExtent * (mag * .5f);
82  view.Up = vtkm::make_Vec(0.f, 1.f, 0.f);
83  view.NearPlane = 1.f;
84  view.FarPlane = 100.f;
85  view.FieldOfView = 60.f;
86  view.Height = 512;
87  view.Width = 512;
88  view.Position = totalExtent * (mag * 2.f);
89  vtkm::rendering::ColorTable colorTable("thermal");
90  sceneRenderer.SetActiveColorTable(colorTable);
91  sceneRenderer.SetView(view);
92
93  // Create a scene for rendering the input data
94  vtkm::rendering::Scene3D scene;
95  vtkm::rendering::Color bg(0.2f, 0.2f, 0.2f, 1.0f);
96  vtkm::rendering::RenderSurfaceOSMesa surface(512,512,bg);
97  scene.plots.push_back(vtkm::rendering::Plot(inputData.GetCellSet(),
98                                              inputData.GetCoordinateSystem(),
99                                              inputData.GetField(fieldName),
00                                              colorTable));
01
02  // Create a window and use it to render the input data using OS Mesa
03  vtkm::rendering::Window3D<SceneRenderer, RenderSurface> w1(scene,
04                                                             sceneRenderer,
05                                                             surface,
06                                                             bg);
07  w1.Initialize();
08  w1.Paint();
09  w1.SaveAs("demo_input.pnm");
```

Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

Demo Part 3: Marching Cubes Filter

```
111  // Create an isosurface filter
112  vtkm::filter::MarchingCubes filter;
113  filter.SetGenerateNormals(false);
114  filter.SetMergeDuplicatePoints(false);
115  filter.SetIsoValue(isovalue);
116  vtkm::filter::DataSetResult result = filter.Execute( inputData,
117                                                       inputData.GetField(fieldName) );
118  filter.MapFieldOntoOutput(result, inputData.GetField(fieldName));
119  vtkm::cont::DataSet& outputData = result.GetDataSet();
120
121  // Render a separate image with the output isosurface
122  std::cout << "about to plot the results of the MarchingCubes filter" << std::endl;
123  scene.plots.clear();
124  scene.plots.push_back(vtkm::rendering::Plot(outputData.GetCellSet(),
125                                              outputData.GetCoordinateSystem(),
126                                              outputData.GetField(fieldName),
127                                              colorTable));
128
129  vtkm::rendering::Window3D< SceneRenderer, RenderSurface> w2(scene,
130                                                              sceneRenderer,
131                                                              surface,
132                                                              bg);
133  w2.Initialize();
134  w2.Paint();
135  w2.SaveAs("demo_output.pnm");
136
137  return 0;
138 }
```

Chris Sewell, Ken Moreland, and Robert Maynard, Presentation at the GPU Technology Conference 2016
http://m.vtk.org/images/3/36/GTC2016-VTKm.pptx

“Big Data”

Hadoop, MapReduce, Spark

Hadoop Distributed File System (HDFS) (1)

- Inspired by "Google File System"
- Stores large files (typically gigabytes-terabytes) across multiple machines, replicating across multiple hosts
  - Breaks up files into fixed-size blocks (typically 64 MiB), distributes blocks
  - The blocks of a file are replicated for fault tolerance
  - The block size and replication factor are configurable per file
  - Default replication value (3) - data is stored on three nodes: two on the same rack, and one on a different rack
- File system intentionally not fully POSIX-compliant
  - Write-once-read-many access model for files. A file once created, written, and closed cannot be changed. This assumption simplifies data coherency issues and enables high throughput data access
  - Intend to add support for appending-writes in the future
  - Can rename & remove files
- "Namenode" tracks names and where the blocks are

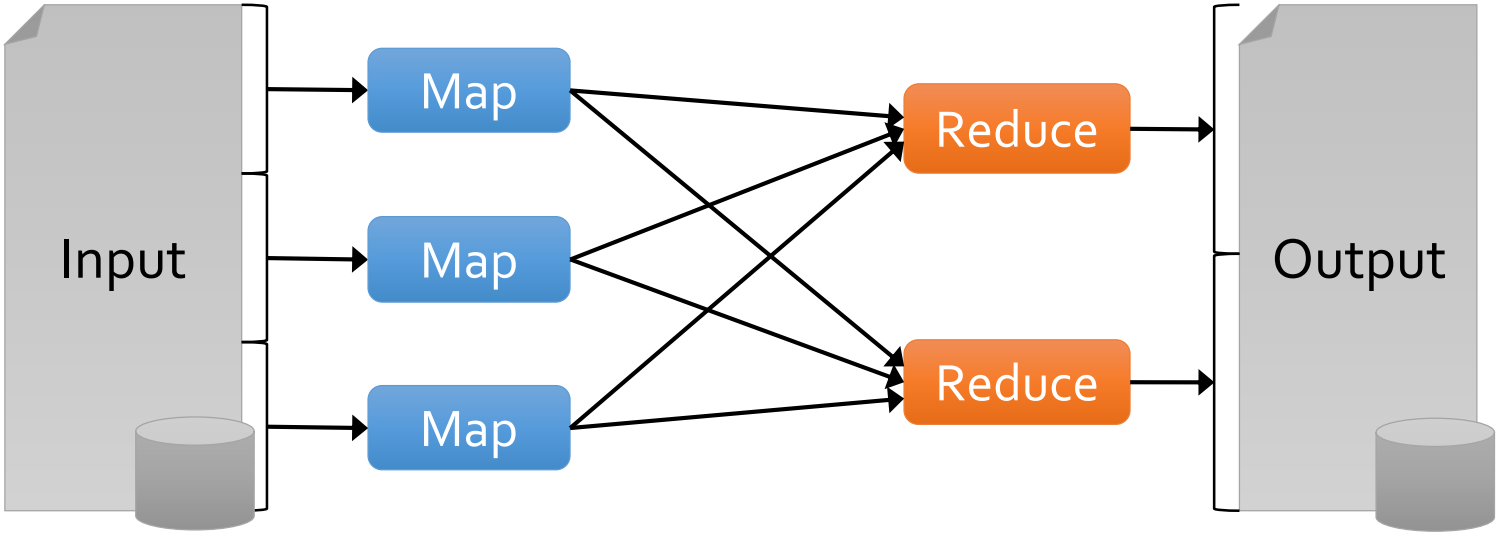
Hadoop Distributed File System (HDFS) (2)

- Hadoop can work with any distributed file system but this loses locality
- To reduce network traffic, Hadoop must know which servers are closest to the data; HDFS does this
- Hadoop job tracker schedules jobs to task trackers with an awareness of the data location
  - For example, if node A contains data (x,y,z) and node B contains data (a,b,c), the job tracker schedules node B to perform tasks on (a,b,c) and node A would be scheduled to perform tasks on (x,y,z)
  - This reduces the amount of traffic that goes over the network and prevents unnecessary data transfer
  - Location awareness can significantly reduce job-completion times when running data-intensive jobs

Source: http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

(Parallel) MapReduce

- MapReduce is a programming model for processing and generating large data sets with a parallel, distributed algorithm on a cluster
- Programmer defines two functions, map & reduce
  - Map(k1,v1) → list(k2,v2). Takes a series of key/value pairs, processes each, generates zero or more output key/value pairs
  - Reduce(k2, list (v2)) → list(v3). Executed once for each unique key k2 in the sorted order; iterate through the values associated with that key and produce zero or more outputs
- System "shuffles" data between map and reduce (so "reduce" function has set of data for its keys) automatically handles system failures, etc.



David Wheeler, Class Presentation at George Mason University
www.dwheeler.com/essays/hadoop-spark.ppt

Image from Matei Zaharia, et. al., Tutorial at spark.apache.org, https://spark.apache.org/talks/overview.pptx

MapReduce: Word Count Example (Pseudocode)

```
map(String input_key, String input_value):
   // input_key: document name
   // input_value: document contents
   for each word w in input_value:
      EmitIntermediate(w, "1");

reduce(String output_key, Iterator intermediate_values):
   // output_key: a word
   // output_values: a list of counts
   int result = 0;
   for each v in intermediate_values:
      result += ParseInt(v);
   Emit(AsString(result));
```

Any 12
Ball 1
Computer 3
…

http://research.google.com/archive/mapreduce-osdi04-slides/index-auto-0004.html

David Wheeler, Class Presentation at George Mason University
www.dwheeler.com/essays/hadoop-spark.ppt

MapReduce Combiner

- Can also define an option function “Combiner” (to optimize bandwidth)
  - If defined, runs after Mapper & before Reducer on every node that has run a map task
  - Combiner receives as input all data emitted by the Mapper instances on a given node
  - Combiner output sent to the Reducers, instead of the output from the Mappers
  - Is a "mini-reduce" process which operates only on data generated by one machine
- If a reduce function is both commutative and associative, then it can be used as a Combiner as well
- Useful for word count – combine local counts

Source: https://developer.yahoo.com/hadoop/tutorial/module4.html

Motivation for Spark

- Acyclic data flow is inefficient for applications that repeatedly reuse a *working set* of data:
  - **Iterative** algorithms (machine learning, graphs)
  - **Interactive** data mining tools (R, Excel, Python)
- With current frameworks, apps reload data from stable storage on each query

Matei Zaharia, et. al., Tutorial at spark.apache.org
https://spark.apache.org/talks/overview.pptx

Solution: Resilient Distributed Datasets (RDDs)

- Allow apps to keep working sets in memory for efficient reuse
- Retain the attractive properties of MapReduce
  - Fault tolerance, data locality, scalability
- Support a wide range of applications

Matei Zaharia, et. al., Tutorial at spark.apache.org
https://spark.apache.org/talks/overview.pptx

Apache Spark

- Processing engine; instead of just “map” and “reduce”, defines a large set of *operations* (transformations & actions)
  - Operations can be arbitrarily combined in any order
- Open source software
- Supports Java, Scala and Python
- Key construct: Resilient Distributed Dataset (RDD)

Spark example #1 (Scala)

```
// "sc" is a "Spark context" – this transforms the file into an RDD
val textFile = sc.textFile("README.md")
// Return number of items (lines) in this RDD; count() is an action
textFile.count()
// Demo filtering.  Filter is a tranform.  By itself this does no real work
val linesWithSpark = textFile.filter(line => line.contains("Spark"))
// Demo chaining – how many lines contain "Spark"?  count() is an action.
textFile.filter(line => line.contains("Spark")).count()
// Length of line with most words.  Reduce is an action.
textFile.map(line => line.split(" ").size).reduce((a, b) => if (a > b) a else b)
// Word count – traditional map-reduce.  collect() is an action
val wordCounts = textFile.flatMap(line => line.split(" ")).map(word => (word,
1)).reduceByKey((a, b) => a + b)
wordCounts.collect()
```

Source: https://spark.apache.org/docs/latest/quick-start.html

Sample Spark transformations

- map(func): Return a new distributed dataset formed by passing each element of the source through a function func.
- filter(func): Return a new dataset formed by selecting those elements of the source on which func returns true
- union(otherDataset): Return a new dataset that contains the union of the elements in the source dataset and the argument.
- intersection(otherDataset): Return a new RDD that contains the intersection of elements in the source dataset and the argument.
- distinct([numTasks])): Return a new dataset that contains the distinct elements of the source dataset
- join(otherDataset, [numTasks]): When called on datasets of type (K, V) and (K, W), returns a dataset of (K, (V, W)) pairs with all pairs of elements for each key. Outer joins are supported through leftOuterJoin, rightOuterJoin, and fullOuterJoin.

Source: https://spark.apache.org/docs/latest/programming-guide.html

Spark – RDD Persistence

- You can persist (cache) an RDD
- When you persist an RDD, each node stores any partitions of it that it computes in memory and reuses them in other actions on that dataset (or datasets derived from it)
- Allows future actions to be much faster (often >10x).
- Mark RDD to be persisted using the persist() or cache() methods on it. The first time it is computed in an action, it will be kept in memory on the nodes.
- Cache is fault-tolerant – if any partition of an RDD is lost, it will automatically be recomputed using the transformations that originally created it
- Can choose storage level (MEMORY_ONLY, DISK_ONLY, MEMORY_AND_DISK, etc.)
- Can manually call unpersist()

Spark vs. Hadoop MapReduce

- Performance: Spark normally faster but with caveats
  - Spark can process data in-memory; Hadoop MapReduce persists back to the disk after a map or reduce action
  - Spark generally outperforms MapReduce, but it often needs lots of memory to do well; if there are other resource-demanding services or can't fit in memory, Spark degrades
  - MapReduce easily runs alongside other services with minor performance differences, & works well with the 1-pass jobs it was designed for
- Ease of use: Spark is easier to program
- Data processing: Spark more general
- Maturity: Spark maturing, Hadoop MapReduce mature

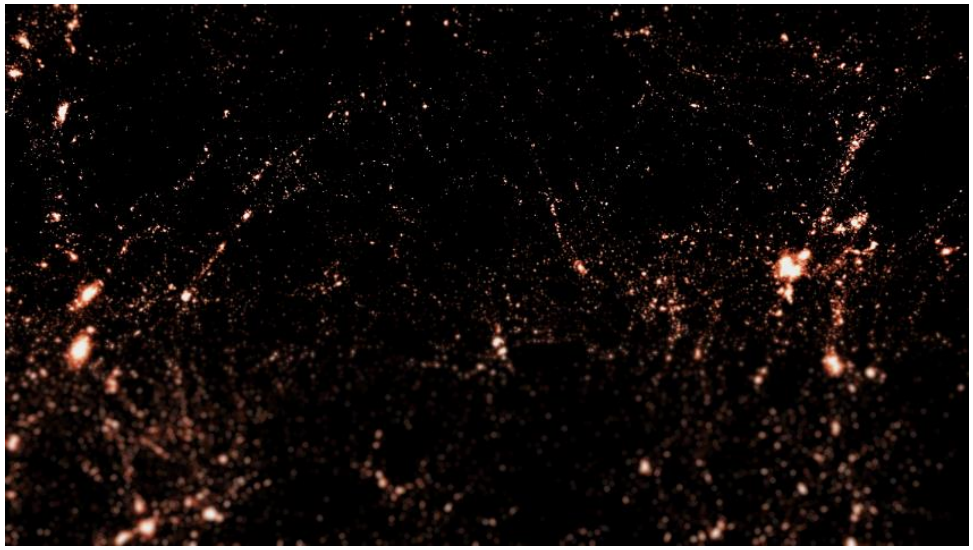
"Spark vs. Hadoop MapReduce" by Saggi Neumann (November 24, 2014)
https://www.xplenty.com/blog/2014/11/apache-spark-vs-hadoop-mapreduce/

Analysis Example

Halo analysis for a cosmology simulation

Intro to Cosmology Simulations

- The Hardware/Hybrid Accelerated Cosmology Code (HACC) simulates the distribution of dark matter in the universe over time
- To achieve high accuracy, simulations may use 100s of billions of particles
- Over time, particles cluster into massive, gravitationally-bound objects called halos
- Hot gas falls into this potential well, heats up, and triggers star formation
- Stars form galaxies, and some halos grow large enough to host clusters of galaxies



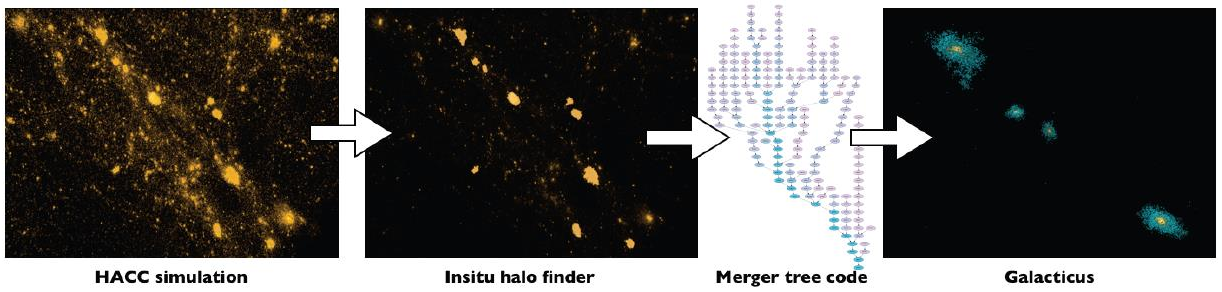
Visualization of a simulation's particle distribution, zoomed in to a sub-region of the volume of a single node, showing halos that have formed in this region at the final time step. Image credit: Silvio Rizzi and Joe Insley.

Chris Sewell, Presentation at Supercomputing 2015:
http://datascience.dsscale.org/wp-content/uploads/sites/3/2016/06/Large-ScaleCompute-IntensiveAnalysisViaACombinedIn-situAndCo-schedulingWorkflowApproach.pdf

Dark Matter Halos

- Dark matter halos are only observed indirectly through tracers, such as hot gas in the X-ray spectrum and the distribution of galaxies in the optical spectrum
- For the optical spectrum, the friend-of-friend (FOF) halo definition is often used, which traces the iso-density contours of halos allowing it to capture irregular boundaries
  - Merger trees are used to track how halos merge and accrete mass over time, since the age and formation history of a halo affect the type of galaxies that will reside in it
  - Sub-halo finding involves finding halos within halos, and galaxies will live within those structures
- Accurately determining halo centers is critical
  - If the halo center is not at the potential minimum, the spherical overdensity mass will be biased low
  - The correct placement of the central galaxy in the halo is crucial for comparison with observations
  - Halo properties such as concentration depend on accurate computation of halo centers

Illustration of the role of analysis in cosmology simulations. Figure from "Large Scale Simulations of Sky Surveys" by Heitmann et. al.



Chris Sewell, Presentation at Supercomputing 2015: http://datascience.dsscale.org/wp-content/uploads/sites/3/2016/06/Large-ScaleCompute-IntensiveAnalysisViaACombinedIn-situAndCo-schedulingWorkflowApproach.pdf

Analysis Tasks

- Power spectrum calculation (using cloud-in-cell density estimation on uniform grid and FFTs)
- Halo identification (using FOF definition)
- Halo center finding (“most bound particle”; the particle with the lowest potential)
- Halo mass estimation (based on a spherical overdensity definition)

Chris Sewell, Presentation at Supercomputing 2015:
http://datascience.dsscale.org/wp-content/uploads/sites/3/2016/06/Large-ScaleCompute-IntensiveAnalysisViaACombinedIn-situAndCo-schedulingWorkflowApproach.pdf

Data-Parallel Analysis Algorithms

- Distributed parallel halo finder uses overload zones so that each rank can independently find halos and their centers
- Supporting the cross-product of halo analysis algorithms with all the multi-core and many-core architectures on which HACC is run is a major burden
- Data-parallel primitives provide an abstraction by which portable parallel code can be implemented to take advantage of on-node parallelism
- Center-finding is the performance bottleneck
- Running our data-parallel center-finder on Titan's GPUs led to significant speed-ups

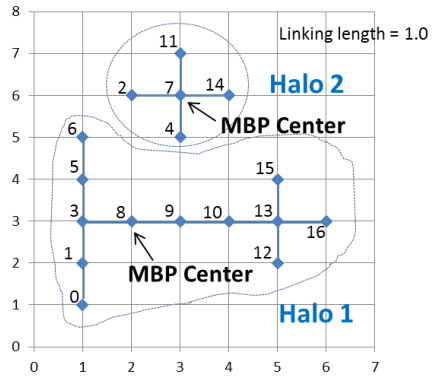


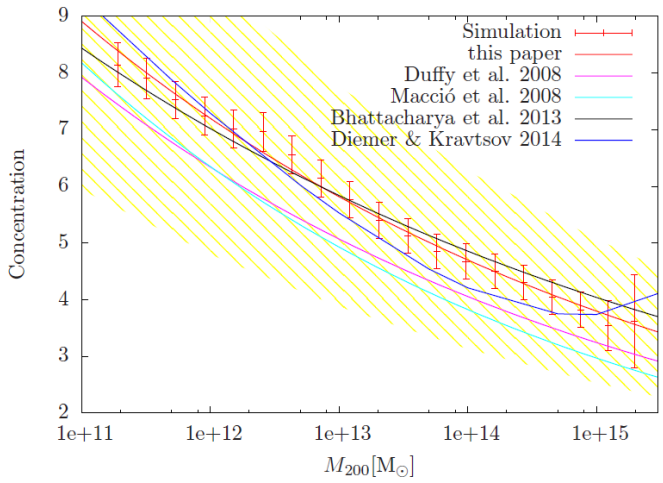
Illustration of friend-of-friend halos and their most bound particle (MBP) centers

Details of our data-parallel algorithms are in "Utilizing Many-Core Accelerators for Halo and Center Finding within a Cosmology Simulation", IEEE Symposium on Large Data Analysis and Visualization, Sewell et. al., Oct. 2015.

Chris Sewell, Presentation at Supercomputing 2015:
http://datascience.dsscale.org/wp-content/uploads/sites/3/2016/06/Large-ScaleCompute-IntensiveAnalysisViaACombinedIn-situAndCo-schedulingWorkflowApproach.pdf

The “Q Continuum” Simulation

- Evolved a 8192^3 (~ a half trillion) particle data set
- Utilized 16,384 (of 18,688) nodes on Titan supercomputer
- Halo analysis was performed at 100 time slices (between red shift values 10 to 0)
- This was the first time that the concentration-mass (c-M) relation has been measured from a single simulation volume over such an extended mass range



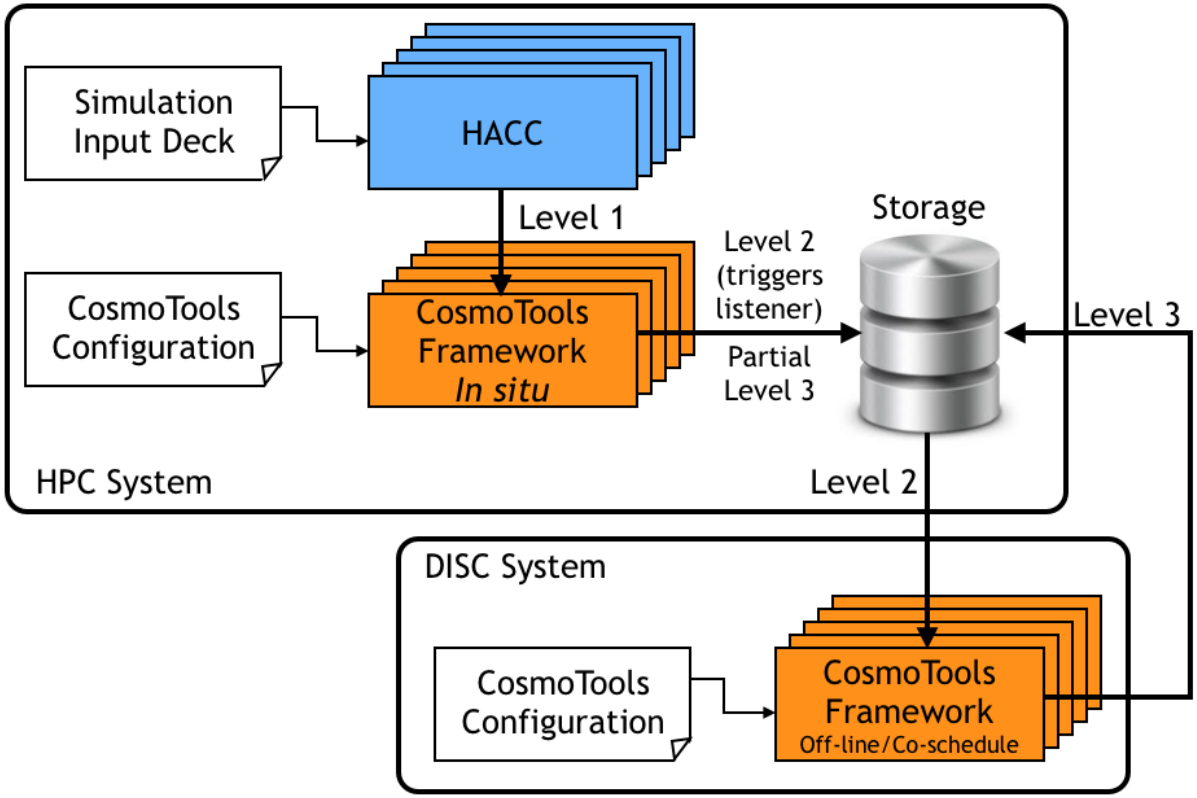
Concentration-mass relation over the full mass range covered by the Q Continuum simulation at redshift z=0

Strong lensing arc generated from the cluster-scale halo; Image credit: Joe Insley, Silvio Rizzi

The scientific results are presented in “The Q Continuum Simulation: Harnessing the Power of GPU Accelerated Supercomputers”, Astrophysical Journal Suppl. Series, Heitmann et. al., Aug. 2015, from which above figures are taken

Chris Sewell, Presentation at Supercomputing 2015:
http://datascience.dsscale.org/wp-content/uploads/sites/3/2016/06/Large-ScaleCompute-IntensiveAnalysisViaACombinedIn-situAndCo-schedulingWorkflowApproach.pdf

Combined In-Situ / Off-line (Co-scheduled) Workflow



Chris Sewell, Presentation at Supercomputing 2015:
http://datascience.dsscale.org/wp-content/uploads/sites/3/2016/06/Large-ScaleCompute-IntensiveAnalysisViaACombinedIn-situAndCo-schedulingWorkflowApproach.pdf