Visualization Toolkit: Faster, Better, Open Scientific Rendering and Compute

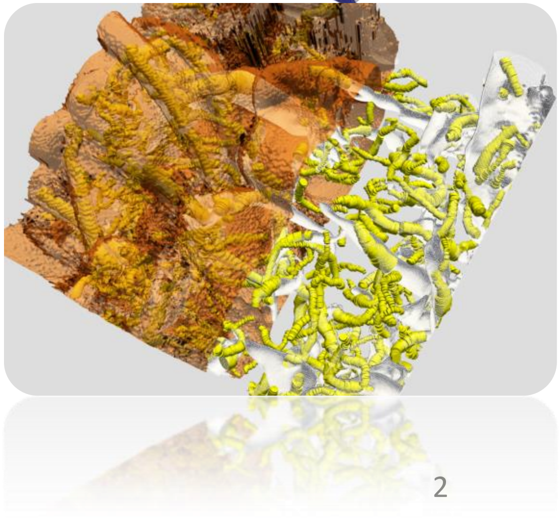
GTC, San Jose, CA
March, 2015

Marcus D. Hanwell
Robert Maynard

Kitware

Accelerating Visualization with Partnerships

- NVIDIA and Kitware collaborate to bring advances in scientific visualization
- Collaboration focuses
  - In-site visualization
  - Advanced rendering
- Improved use of NVIDIA GPUs



Kitware, Inc.

- Founded in 1998 by five former GE Research employees
- 98 current employees; 34 with PhDs
- Privately held, profitable from creation, no debt
- Offices
  - Clifton Park, NY
  - Carrboro, NC
  - Santa Fe, NM
  - Lyon, France

- 2011 Small Business Administration's Tibbetts Award
- HPCWire Readers and Editor's Choice
- Inc's 5000 List since 2008

Kitware

Kitware's customers & collaborators

Over 75 **academic** institutions including…

- Harvard
- Massachusetts Institute of Technology
- University of California, Berkeley
- Stanford University
- California Institute of Technology
- Imperial College London
- Johns Hopkins University
- Cornell University
- Columbia University
- Robarts Research Institute
- University of Pennsylvania
- Rensselaer Polytechnic Institute
- University of Utah
- University of North Carolina

Over 50 **government** agencies and labs including…

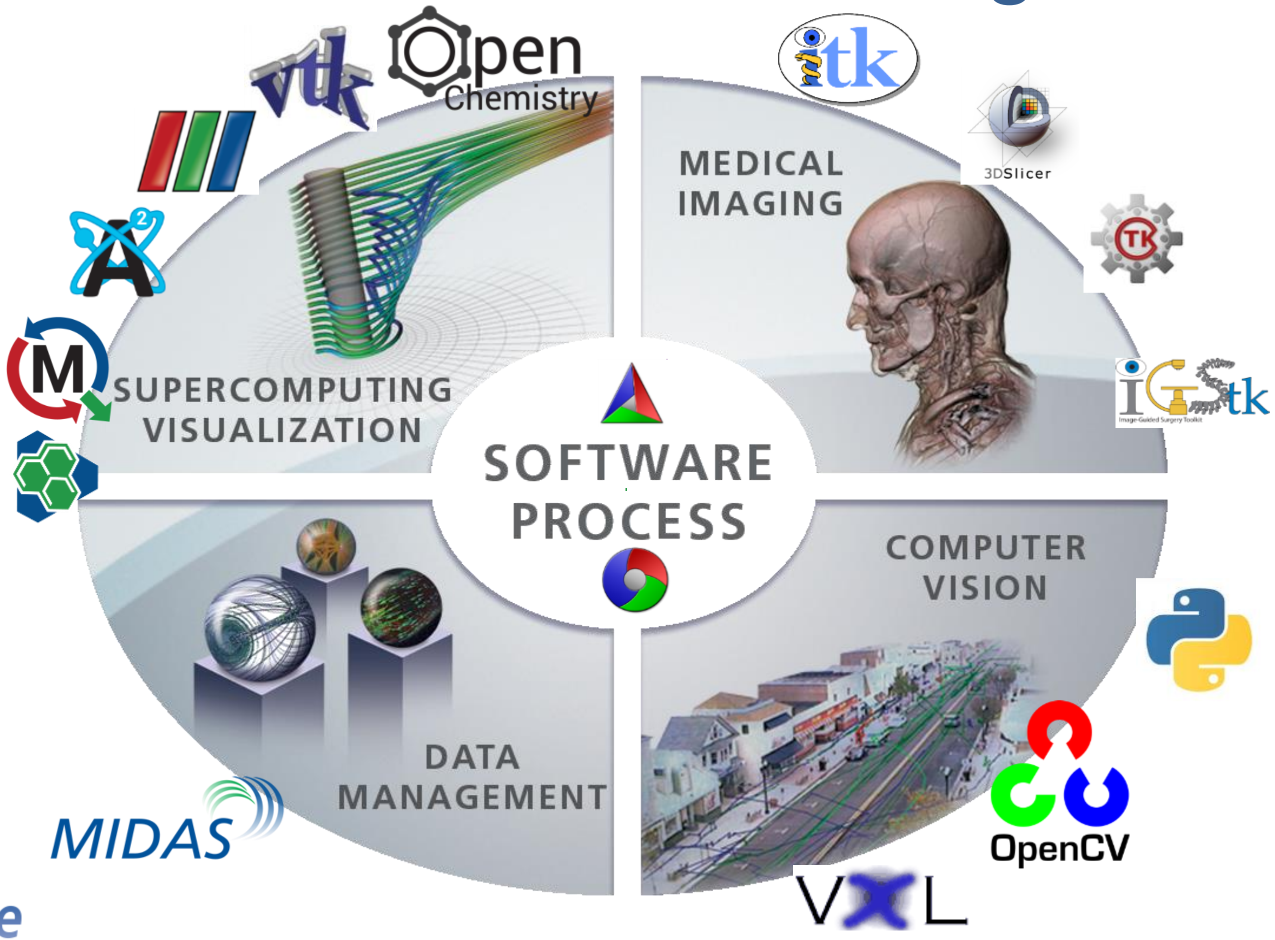
- National Institutes of Health (NIH)
- National Science Foundation (NSF)
- National Library of Medicine (NLM)
- Department of Defense (DOD)
- Department of Energy (DOE)
- Defense Advanced Research Projects Agency (DARPA)
- Army Research Lab (ARL)
- Air Force Research Lab (AFRL)
- Sandia (SNL)
- Los Alamos National Labs (LANL)
- Argonne (ANL)
- Oak Ridge (ORNL)
- Lawrence Livermore (LLNL)

Over 100 **commercial** companies in fields including…

- Automotive
- Aircraft
- Defense
- Energy technology
- Environmental sciences
- Finance
- Industrial inspection
- Oil & gas
- Pharmaceuticals
- Publishing
- 3D Mapping
- Medical devices
- Security
- Simulation

Kitware

Kitware: Core Technologies

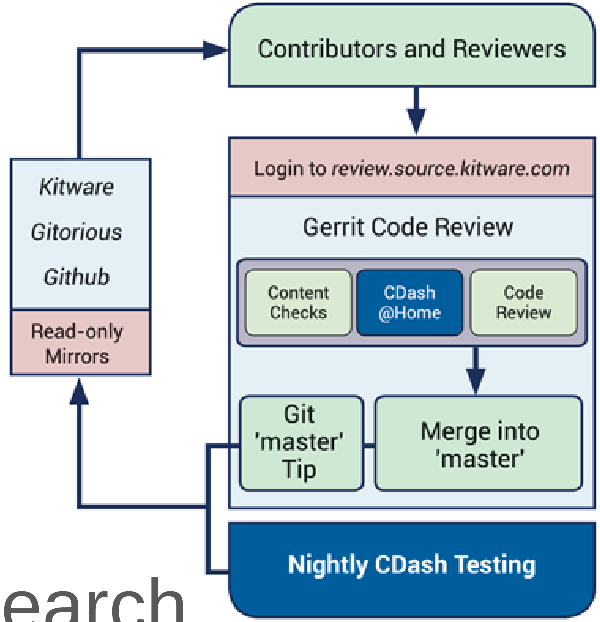


Business Model: Open Source

- Open-source Software
  - Normally BSD-licensed
  - Collaboration platforms
- Collaborative Research and Development
- Technology Integration
- Services, support, and consulting
- Training and webinars

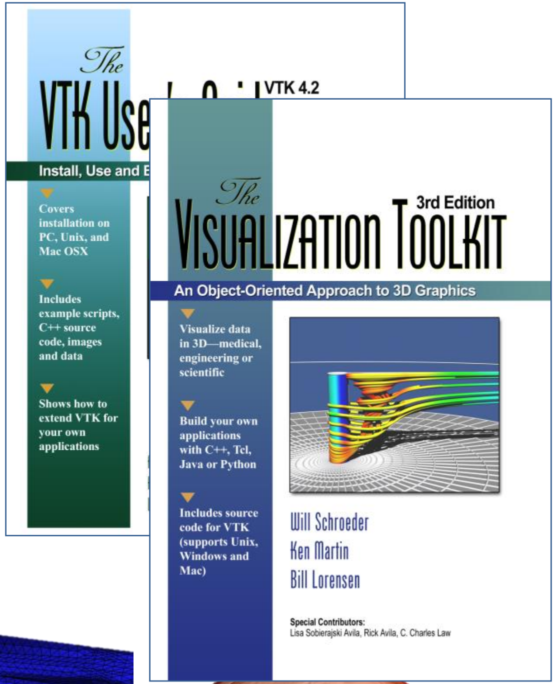
Overview of Software Process

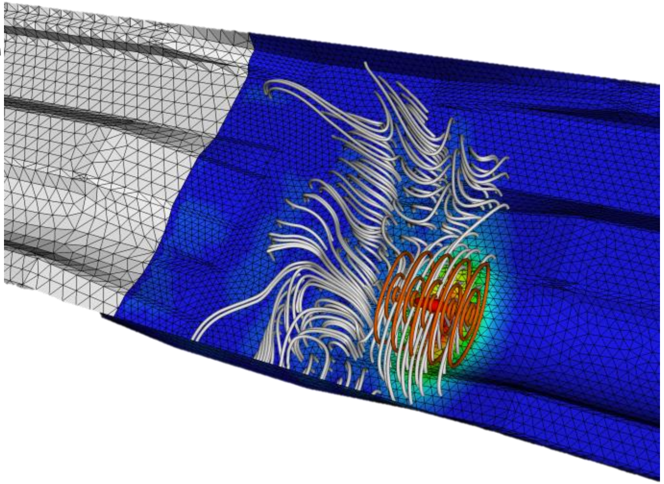
- Openly developed, reusable frameworks
  - Open-source frameworks
  - Developed openly
  - Cross-platform compatibility
  - Tested and verified
  - Contribution model
  - Supported by Kitware experts
- Liberally-licensed to facilitate research

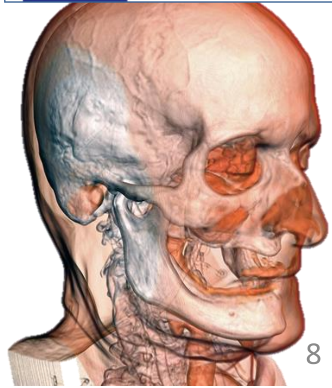


The Visualization Toolkit

- Founded in 1993 as example code for “The Visualization Textbook”.
- Used in many projects developed all over the world
  - ParaView, VisIt
  - Osirix, 3D Slicer
  - Mayavi, MOOSE



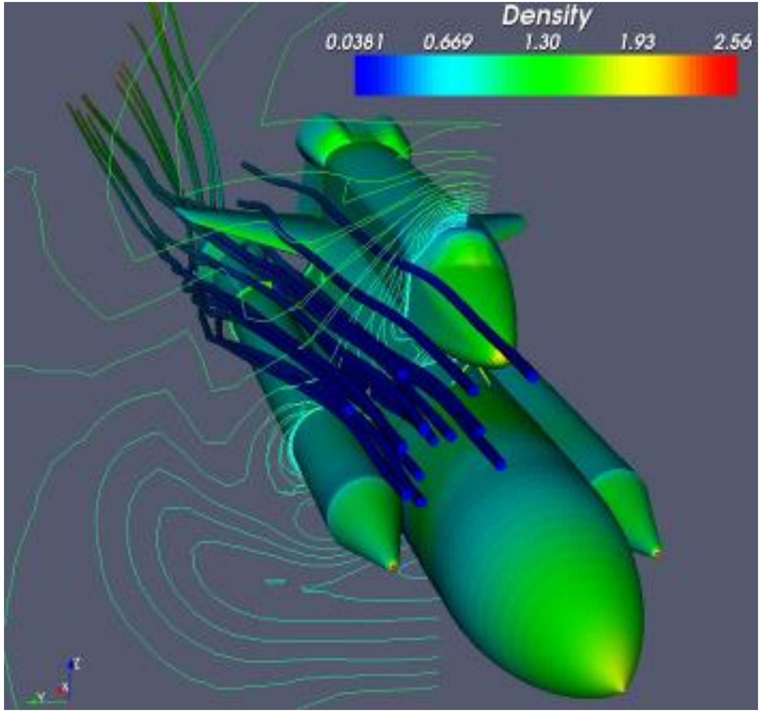




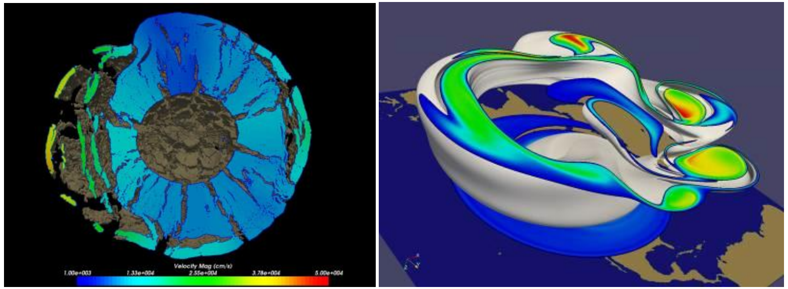
Kitware

Going From Data to Visualization

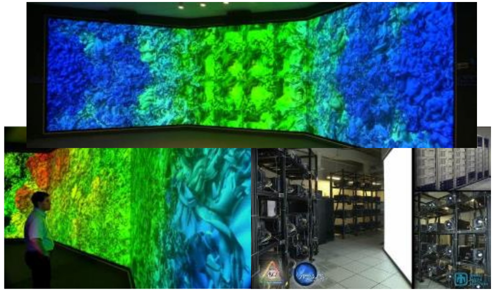

```
0265640 132304 133732 032051 037334 024721 015013 052226 001662
0265660 025537 064663 054606 043244 074076 124153 135216 126614
0265700 144210 056426 044700 042650 165230 137037 003655 006254
0265720 134453 124327 176005 027034 107614 170774 073702 067274
0265740 072451 007735 147620 061064 157435 113057 155356 114603
0265760 107204 102316 171451 046040 120223 001774 030477 046673
0266000 171317 116055 155117 134444 167210 041405 147127 050505
0266020 004137 046472 124015 134360 173550 053517 044635 021135
0266040 070176 047705 113754 175477 105532 076515 177366 056333
0266060 041023 074017 127113 003214 037026 037640 066171 123424
0266100 067701 037406 140000 165341 072410 100032 125455 056646
0266120 006716 071402 055672 132571 105645 170073 050376 072117
0266140 024451 007424 114200 077733 024434 012546 172404 102345
0266160 040223 050170 055164 164634 047154 126525 112514 032315
0266200 016041 176055 042766 025015 176314 017234 110060 014515
0266220 117156 030746 154234 125001 151144 163706 136237 164376
0266240 137055 062276 161755 115466 005322 132567 073216 002655
0266260 171466 126161 117155 065763 016177 014460 112765 055527
0266300 003767 175367 104754 036436 172172 150750 043643 145410
0266320 072074 000007 040627 070652 173011 002151 125132 140214
0266340 060115 014356 015164 067027 120206 070242 033065 131334
0266360 170601 170106 040437 127277 124446 136631 041462 116321
0266400 020243 005602 004146 121574 124651 006634 071331 102070
0266420 157504 160307 166330 074251 024520 114433 167273 030635
0266440 133614 106171 144160 010652 007365 026416 160716 100413
0266460 026630 007210 000630 121224 076033 140764 000737 003276
0266500 114060 042647 104475 110537 066716 104754 075447 112254
0266520 030374 144251 077734 015157 002513 173526 035531 150003
0266540 146207 015135 024446 130101 072457 040764 165513 156412
0266560 166410 067251 156160 106406 136770 030516 064740 022032
0266600 142166 123707 175121 071170 076357 037233 031136 015232
0266620 075074 016744 044055 102230 110063 033350 052765 172463
```

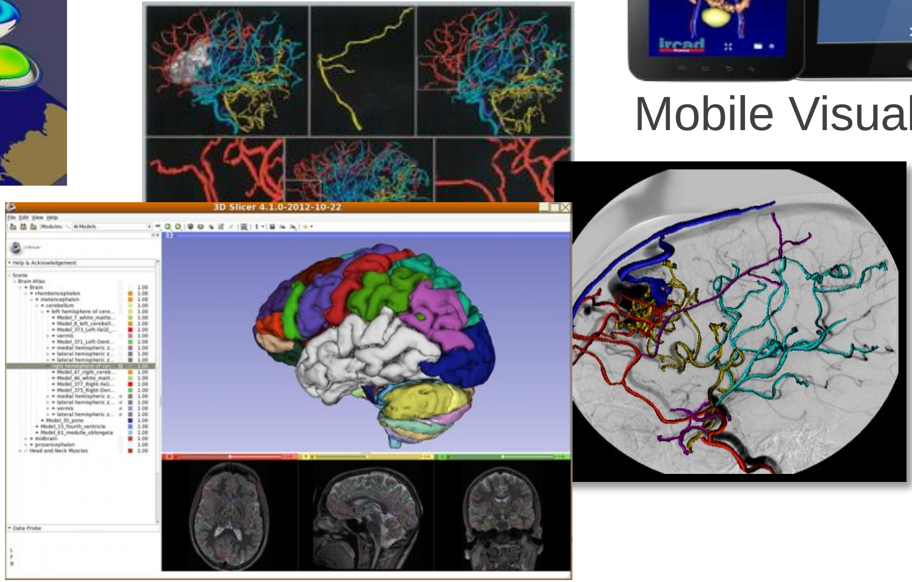
VTK Visualizations



HPC Visualization



Large Displays and Virtual Reality



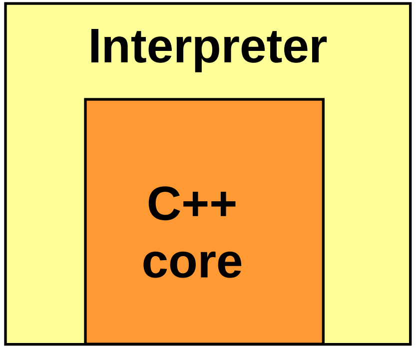
Interactive Medical Application and Visualization

Mobile Visualization

Kitware

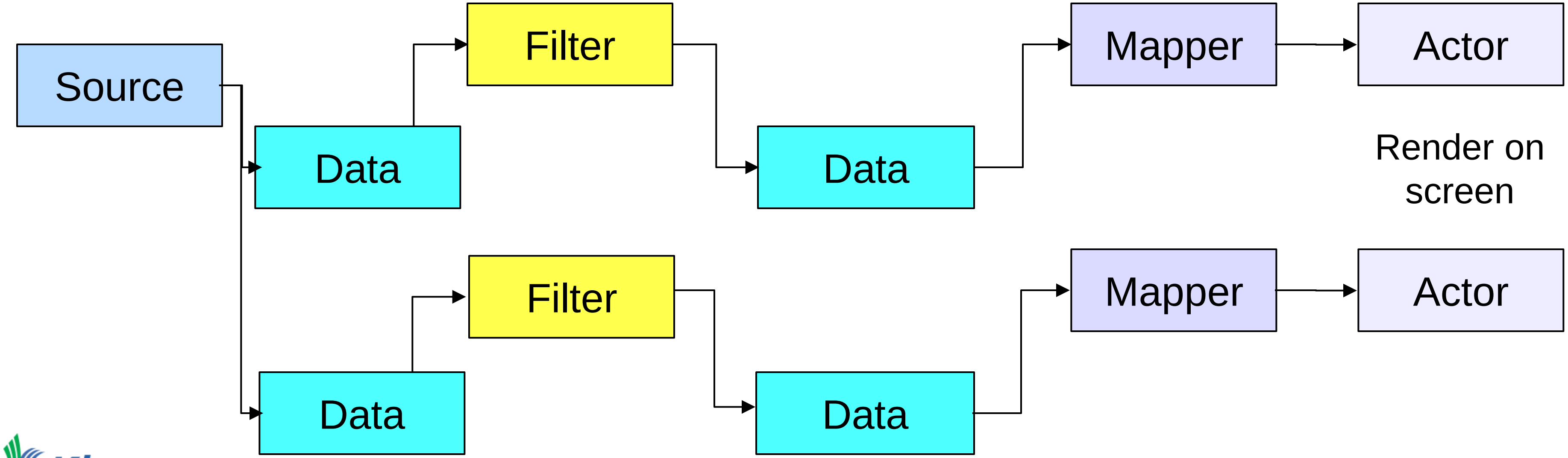
VTK Architecture

- Hybrid approach
  - Compiled C++ core (faster algorithms)
  - Interpreted applications (rapid development)
  - Interpreted layer generated automatically



The Visualization Pipeline

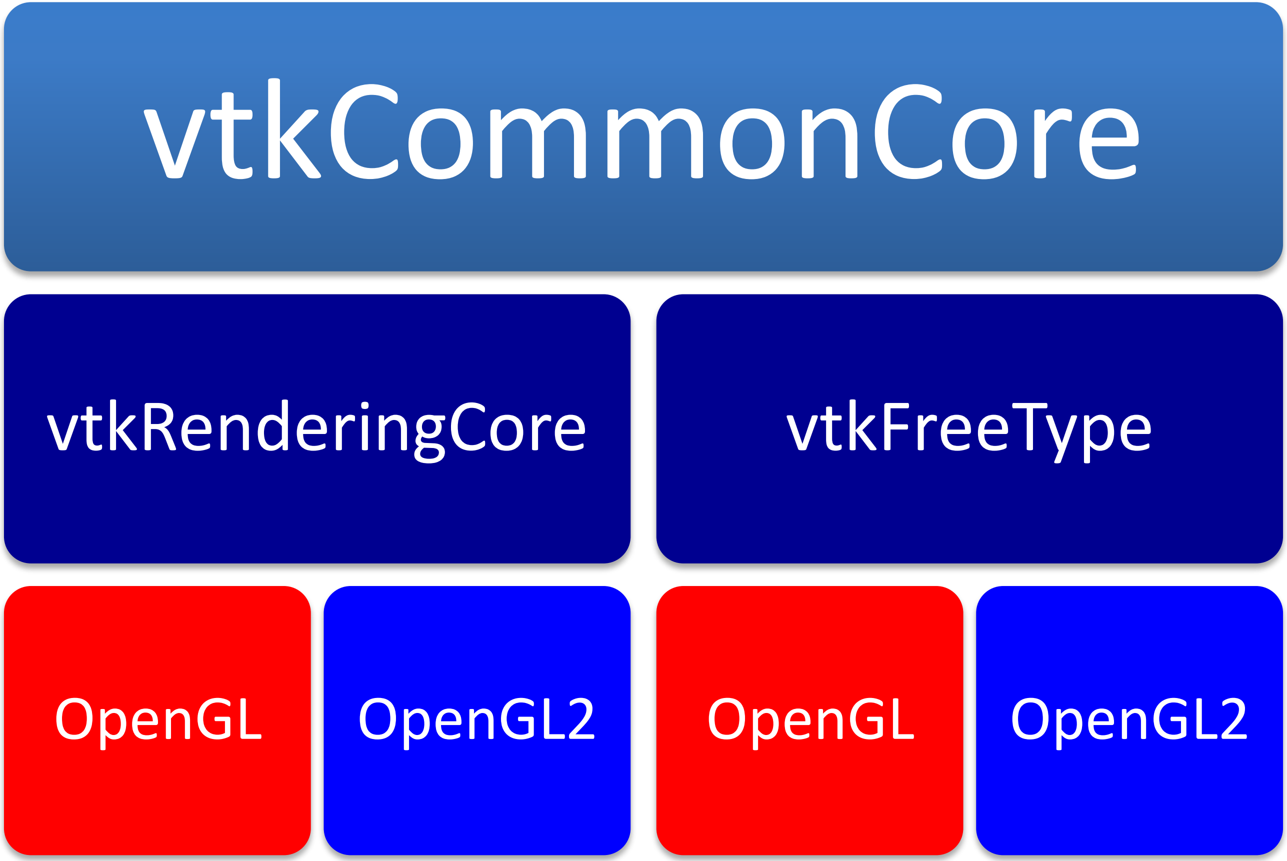
- A sequence of algorithms that operate on data objects to generate geometry



VTK Organization

- Libraries with public APIs
- Cross-platform, open-source, for reuse
- Implementation modules use factories
  - Rendering API uses OpenGL backend
  - Core rendering does not link to/use OpenGL

Basic Library Hierarchy



Legacy Rendering

- Based on OpenGL 1.1 APIs
  - Optionally uses some extensions
- Heavy use of display lists for interaction
- A “Painter” API to enable custom rendering
  - Virtual functions, switches, …
    - In tight loops for all vertices, normals, colors, etc

Polygonal Rendering Rewrite

- New minimum OpenGL version
  - OpenGL 2.1, OpenGL ES 2.0
- Rewrite to use minimal common subset
- Major overhaul of the rendering code
  - Use VBOs, VAOs, shaders, “new” OpenGL
- Retain same high level API

Volume Rendering Rewrite

- Improve portability of GPU code
  - Works well on Linux, Mac, and Windows
  - Uses less extensions, more core GL 2.1+
- Refactored to compute more in shaders
- Replicates important features
- Easier to develop new techniques

Removing Old Calls

- Not using matrix stacks
- GLSL, using modern approaches
- Optional extensions detected at runtime
- Not a single glVertex call, highly batched
- Some data structures need further work
  - vtkPolyData needs packed triangles

Performance Improvements

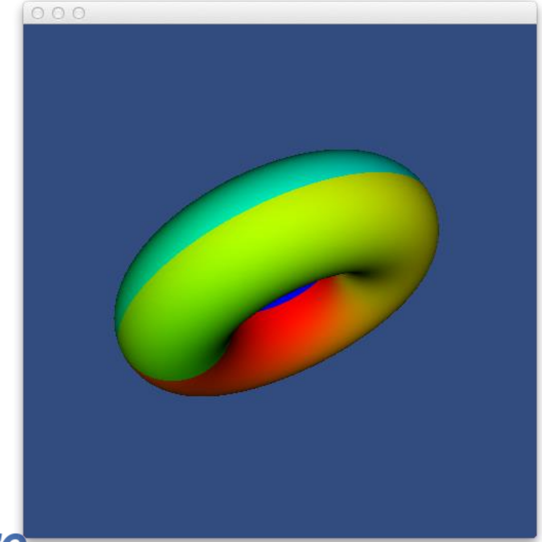
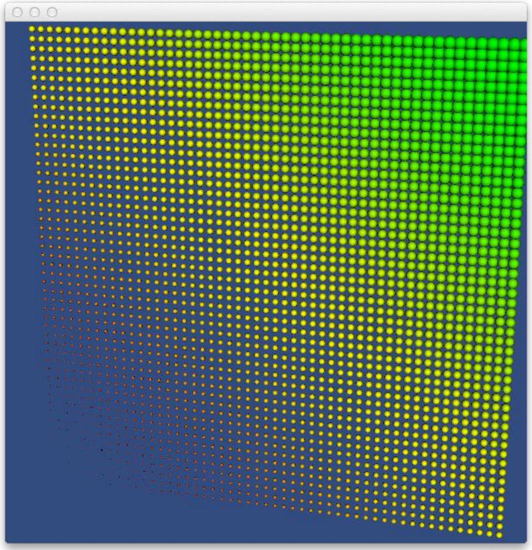
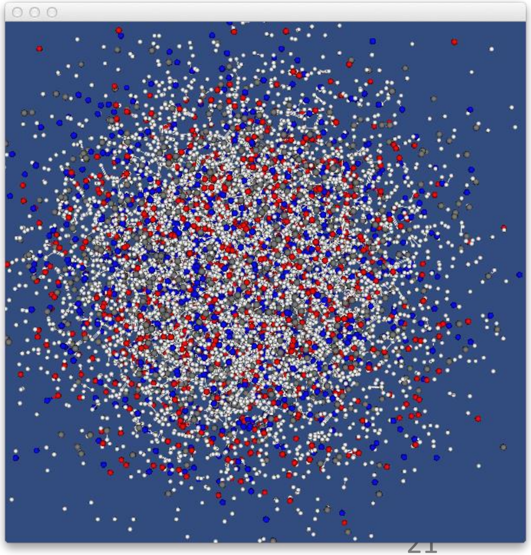
- In many cases now GPU bound
  - Previously large systems CPU bound
- Large polygonal models >100x faster!
- Much more portable depth peeling
- Reduced memory footprint significantly
- Initial render times reduced

Performance: Old vs New

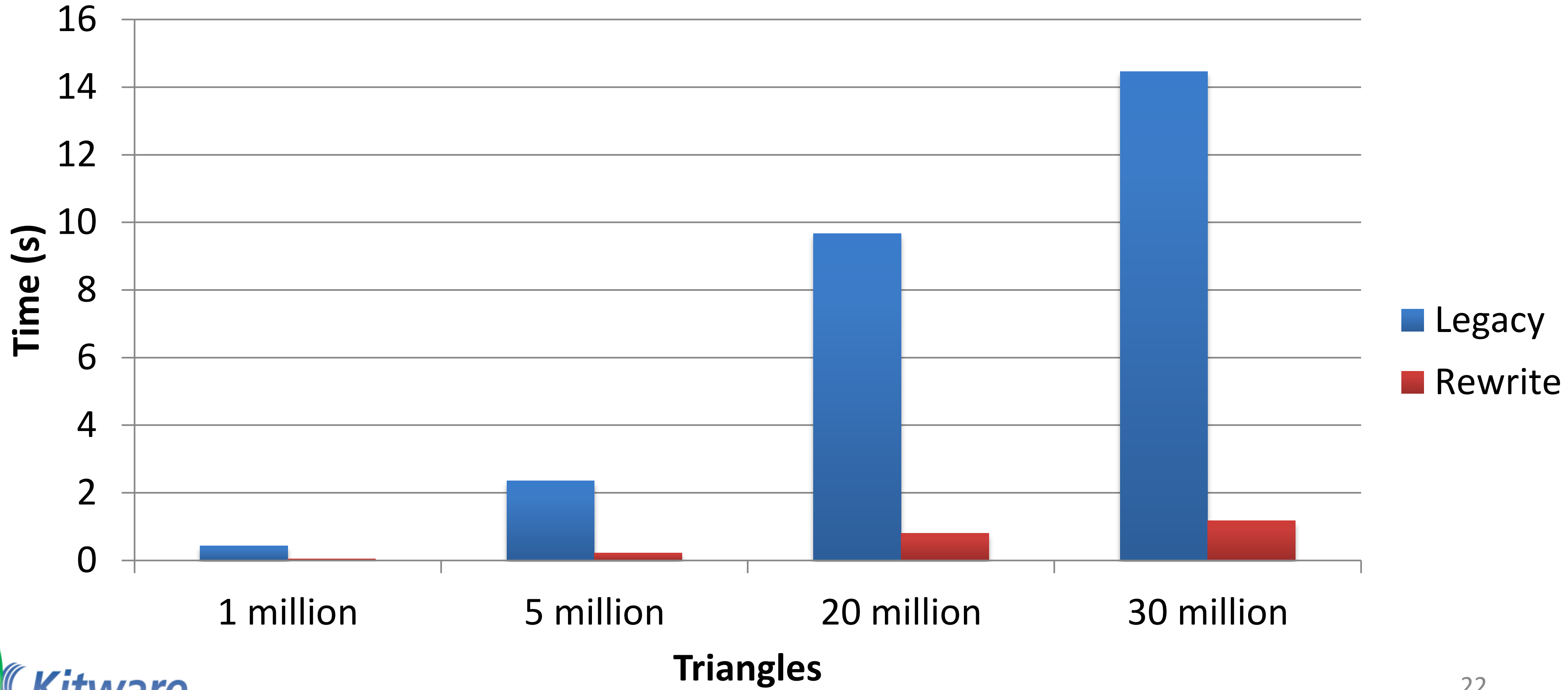
- Looking at static scenes
  - Time to first render
  - Average time of rotated subsequent renders
- Legacy rendering hits maximum size
  - Memory errors/limits
  - Only possible to compare smaller geometries

Benchmarking Tools (Polygonal)

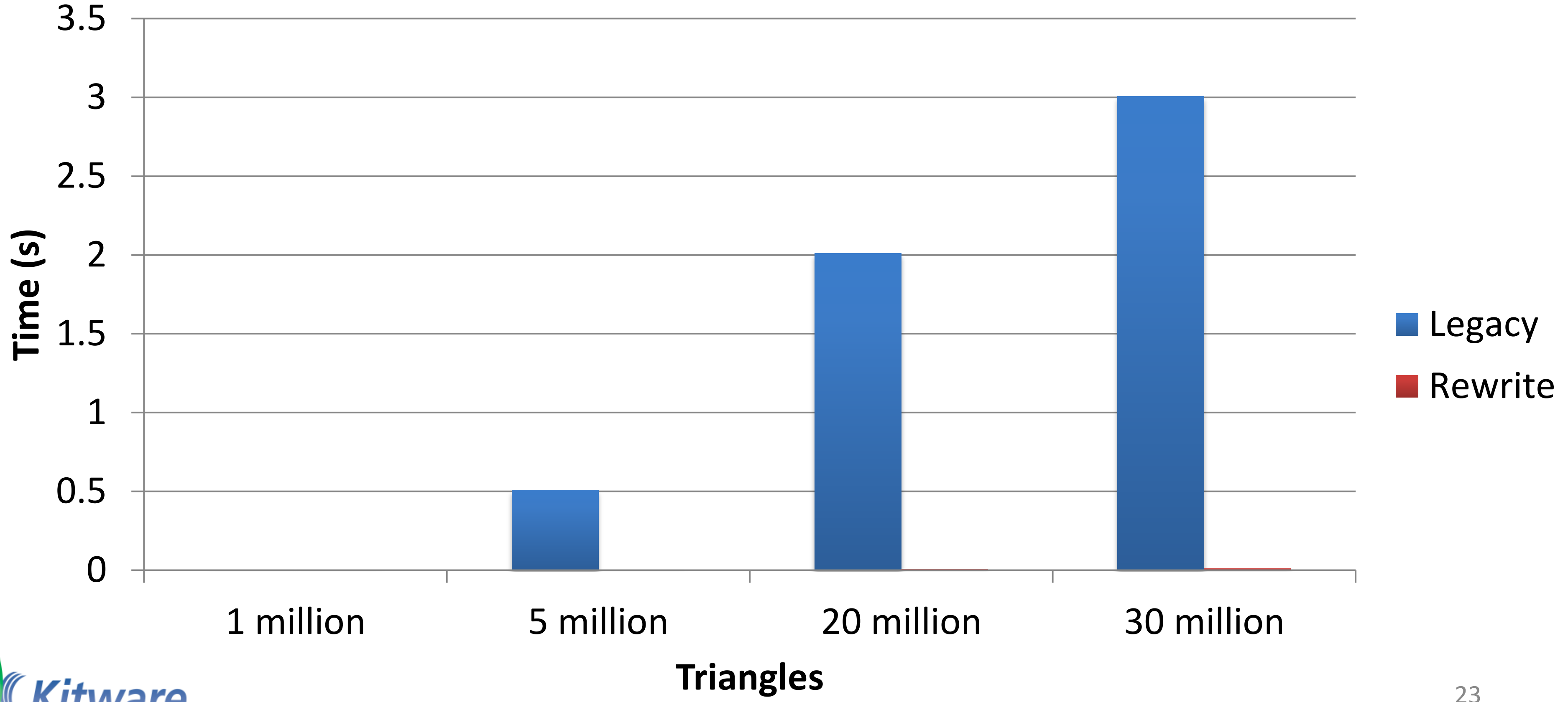
- Added some new benchmarking tools
- Aim to provide systematic comparison

Time For First Frame (K6000)



Time for Subsequent Frames (K6000)

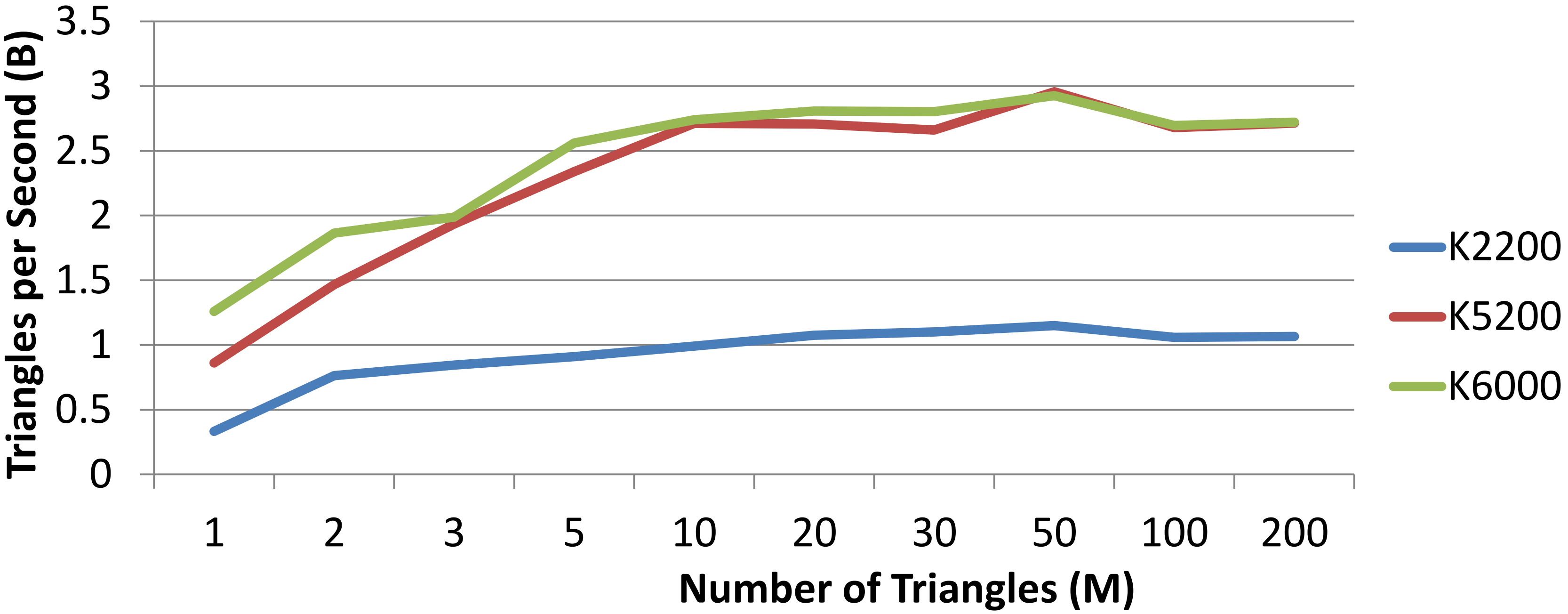


Kitware

Rendering Speeds

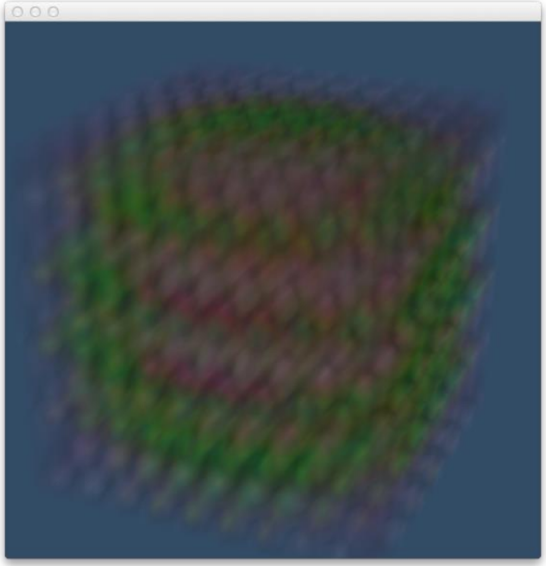
- Two orders of magnitude faster!
- Legacy rendering maxes out at 30 million
  – Not possible to compare above this
- Measured on a modern Linux system
  – Same on Windows, and Mac
- Memory footprint about half for triangles

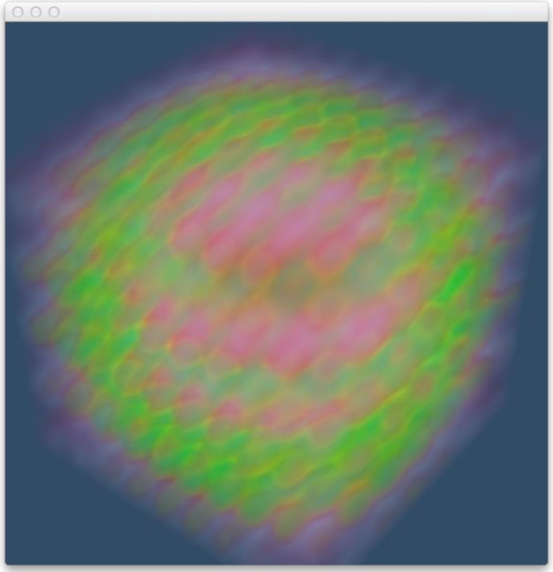
Comparison of Cards (Rewrite)



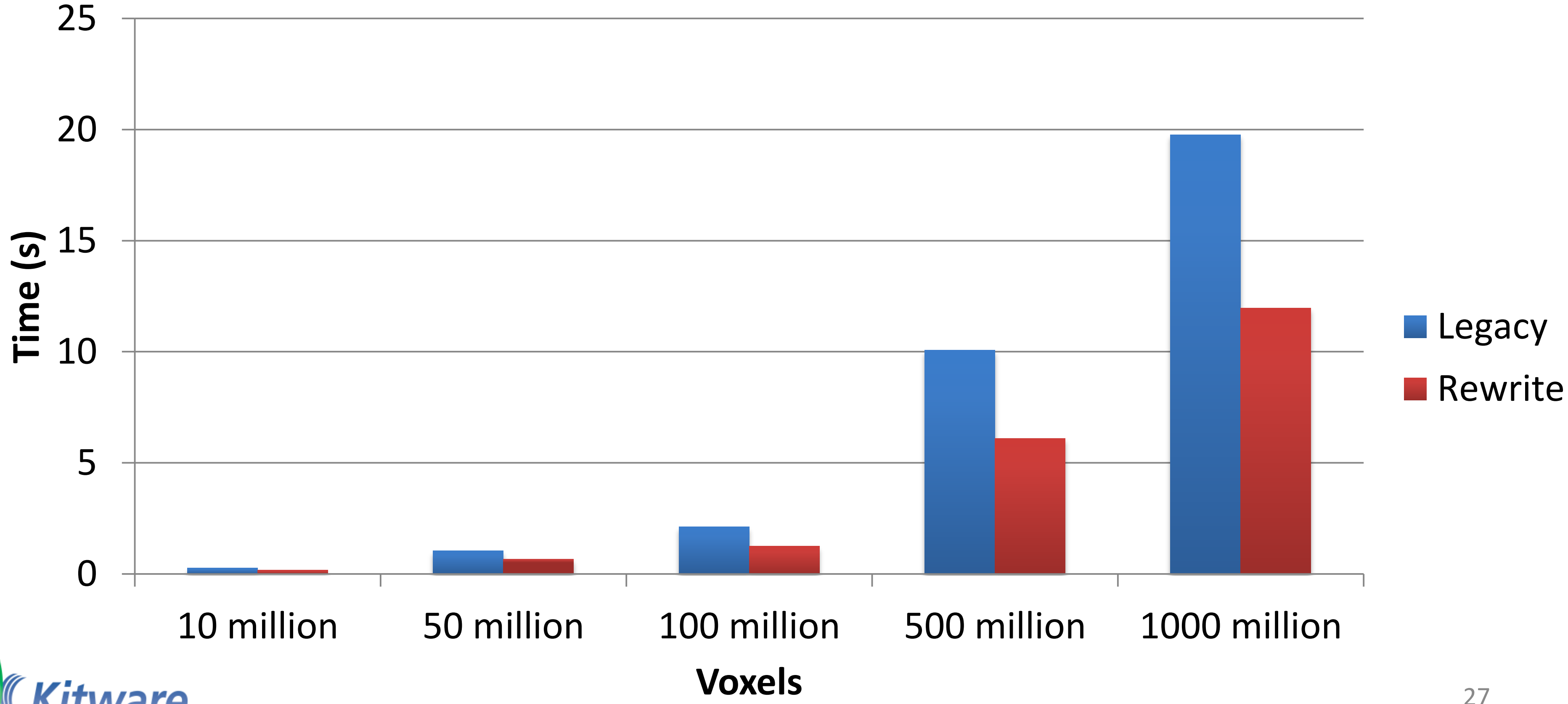
Benchmarking Tools (Volume)

- Uses same framework as polygonal
- Volumes of increasing size



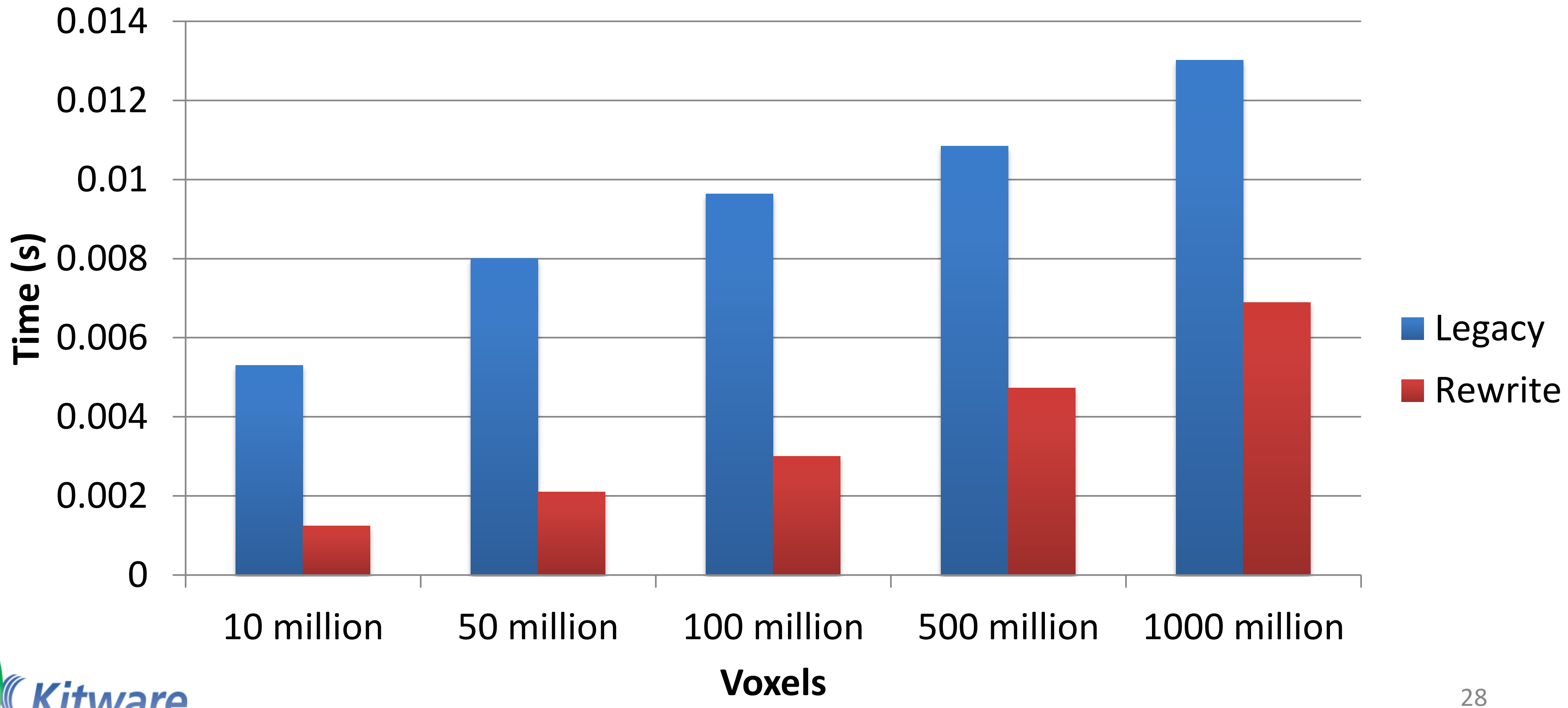


Time For First Frame (K40c)



Kitware

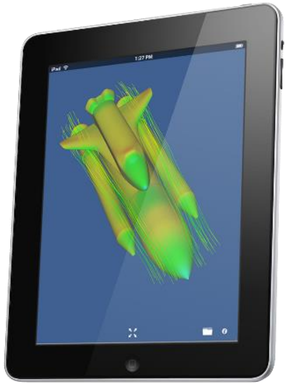
Time for Subsequent Frames (K40c)



Kitware

Mobile/Embedded

- New rendering can target ES 2.0+
- Some testing on Android and iOS
- Largely shared code with desktop code
- Simple multitouch interaction support



Custom Rendering

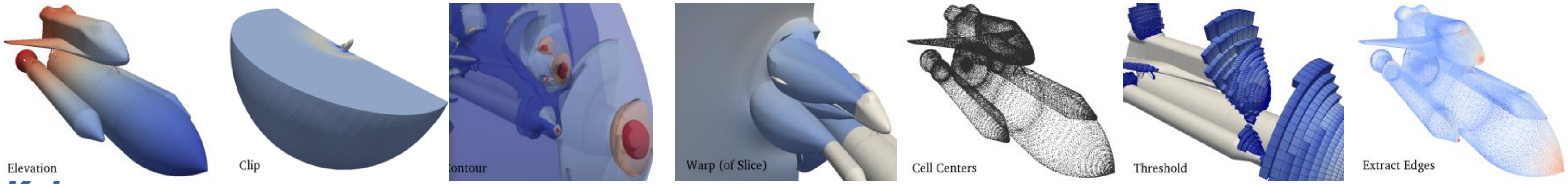
- Shaders can be overridden in mappers
- VBOs/IBOs created by reusable helpers
- Override the vtkMapper class
- Several examples of different rendering
  - Glyphing, impostors, composite data
  - Offer a reasonable starting point

Porting/Using New Rendering

- Many applications just change backend
  - VTK_RENDERING_BACKEND=OpenGL2
  - Compile time option, with possible link change
  - vtkRenderingOpenGL -> vtkRendering${VTK_RENDERING_BACKEND}
- Custom OpenGL will need to be ported

VTK-m Project Goals

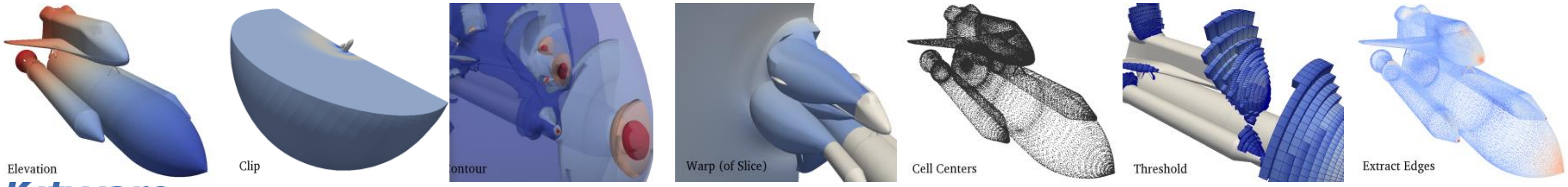
- A single place for the visualization community to collaborate, contribute, and leverage massively threaded algorithms.
- Reduce the challenges of writing highly concurrent algorithms by using data parallel algorithms



Kitware

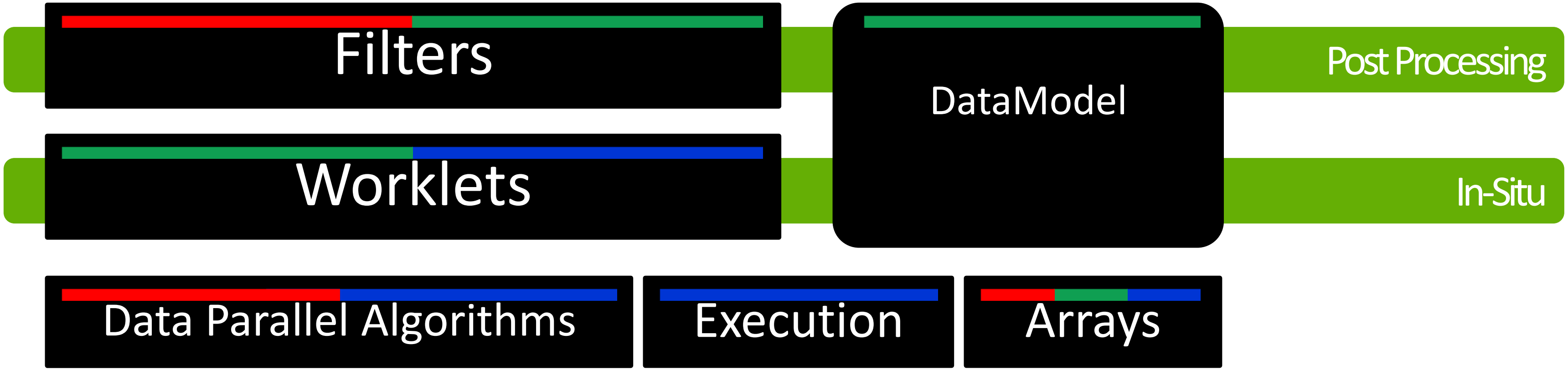
VTK-m Project Goals

- Make it easier for simulation codes to take advantage these parallel visualization and analysis tasks on a wide range of current and next-generation hardware.



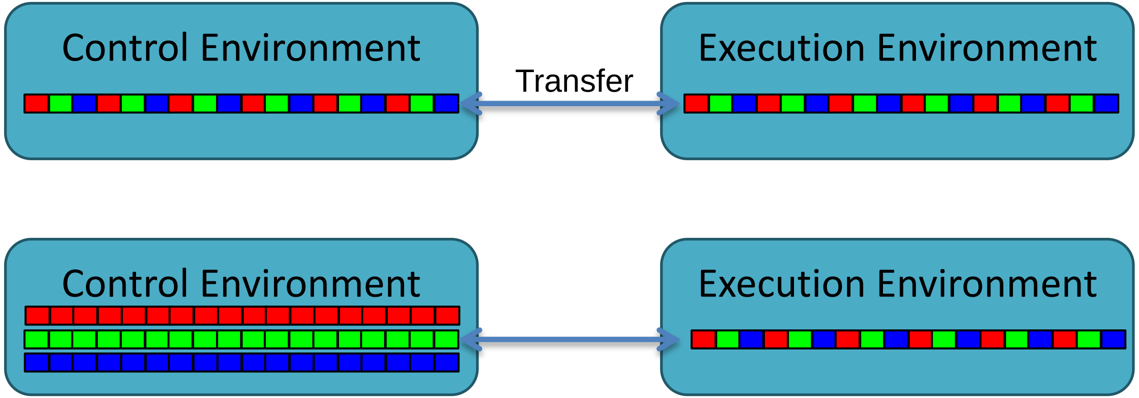
VTK-m Architecture

- Combines strengths of multiple projects:
  - EAVL, Oak Ridge National Laboratory
  - DAX, Sandia National Laboratory
  - PISTON, Los Alamos National Laboratory



VTK-m Arbitrary Composition

- VTK-m allows clients to access different memory layouts through the Array Handle and Dynamic Array Handle.
  - –Allows for efficient in-situ integration
  - –Allows for reduced data transfer



Kitware

VTK-m Arbitrary Composition

- VTK-m allows clients to construct data sets from cell and point arrangements that exactly match their original data
  - –In effect, this allows for hybrid and novel mesh types

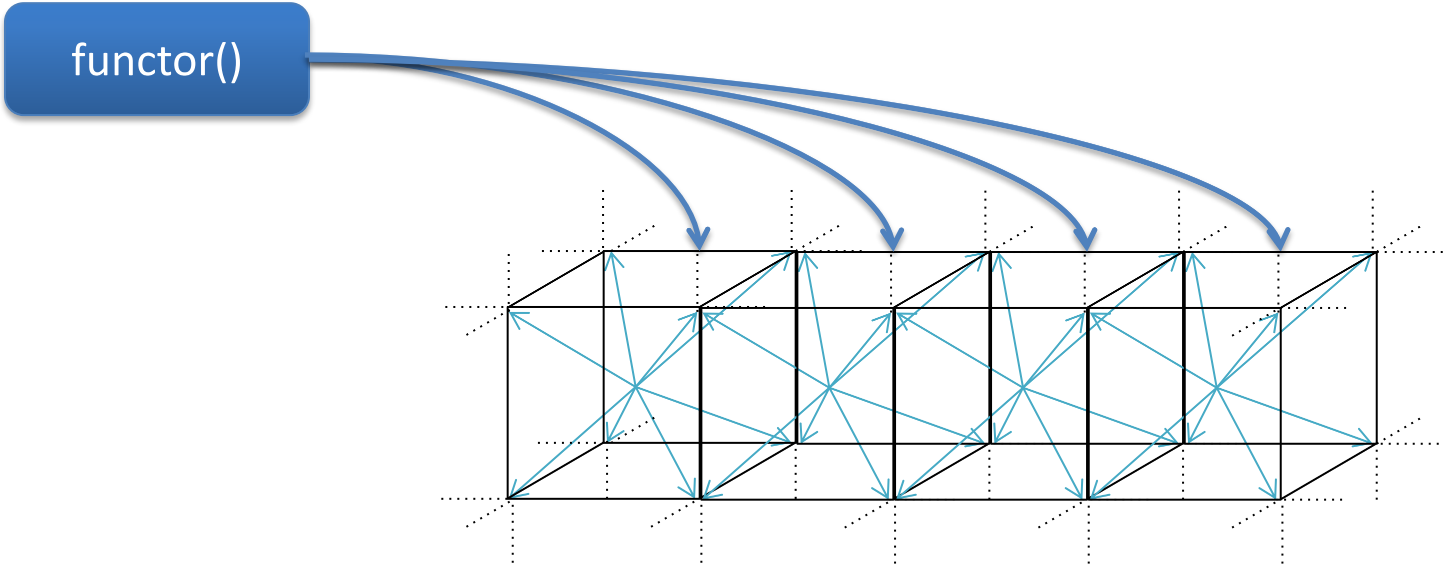
Cells	Coordinates	Point Arrangement: Explicit	Logical	Implicit
Structured	Strided	✓	✓	✓
	Separated	✓	✓	✓
Unstructured	Strided	✓	✓	✓
	Separated	✓	✓	✓

VTK-m Data Set

Kitware

[Baker, et al. 2010]

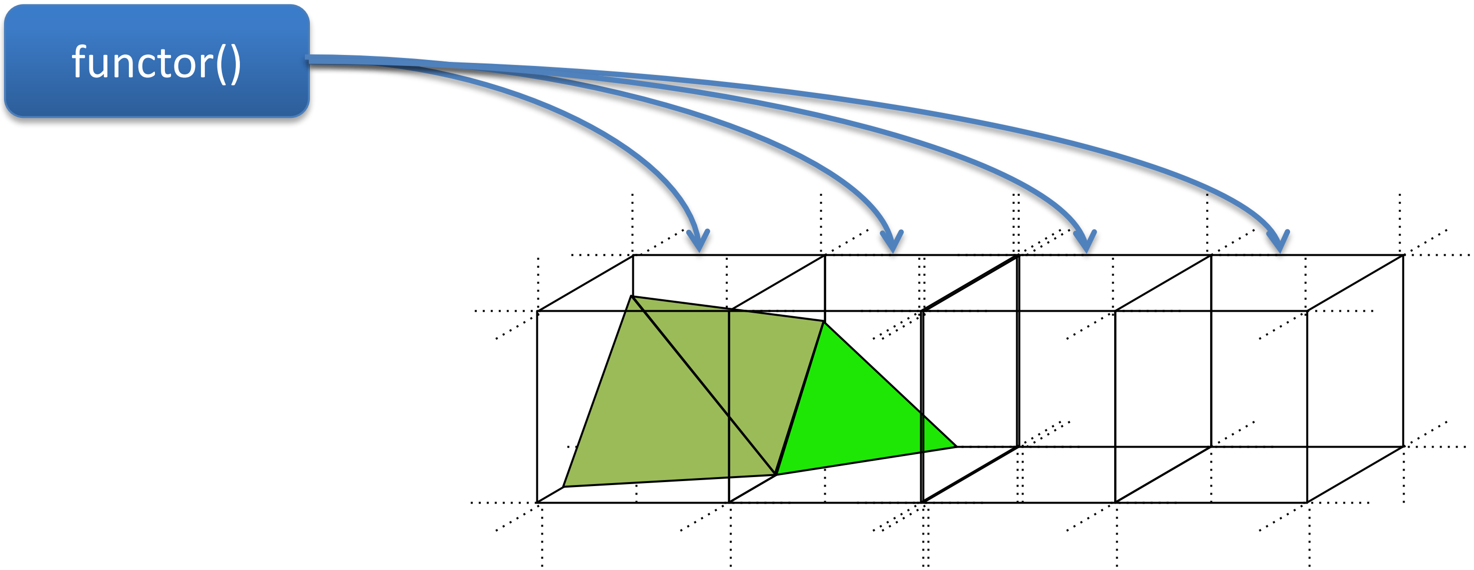
Functor Mapping Applied to Topologies



Kitware

[Baker, et al. 2010]

Functor Mapping Applied to Topologies



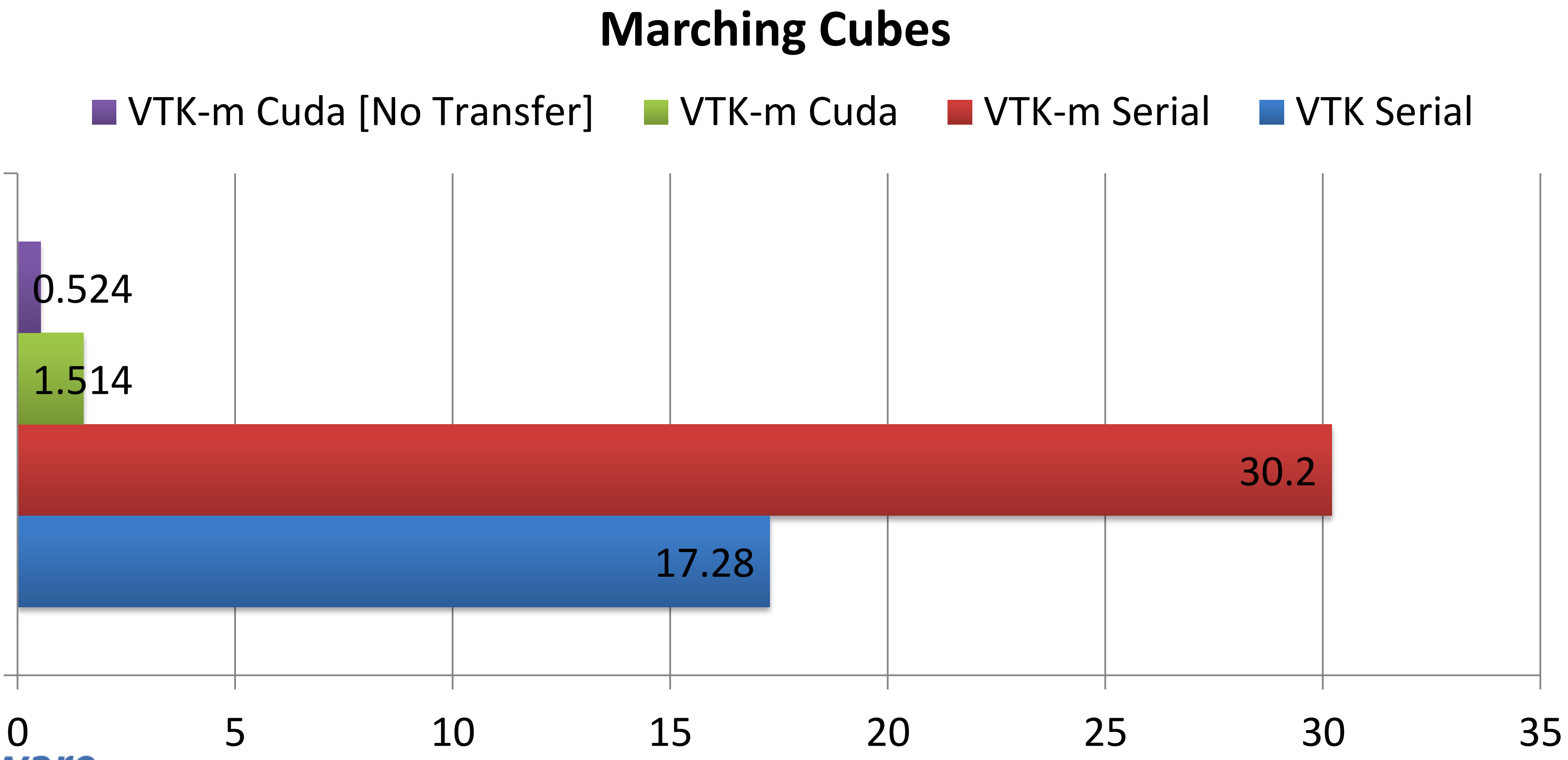
What We Have So Far

- Features
  - Core Types
  - Statically and Dynamically Typed Arrays
  - Device Interface (Serial, Cuda, TBB under development)
  - Basic Worklet and Dispatcher

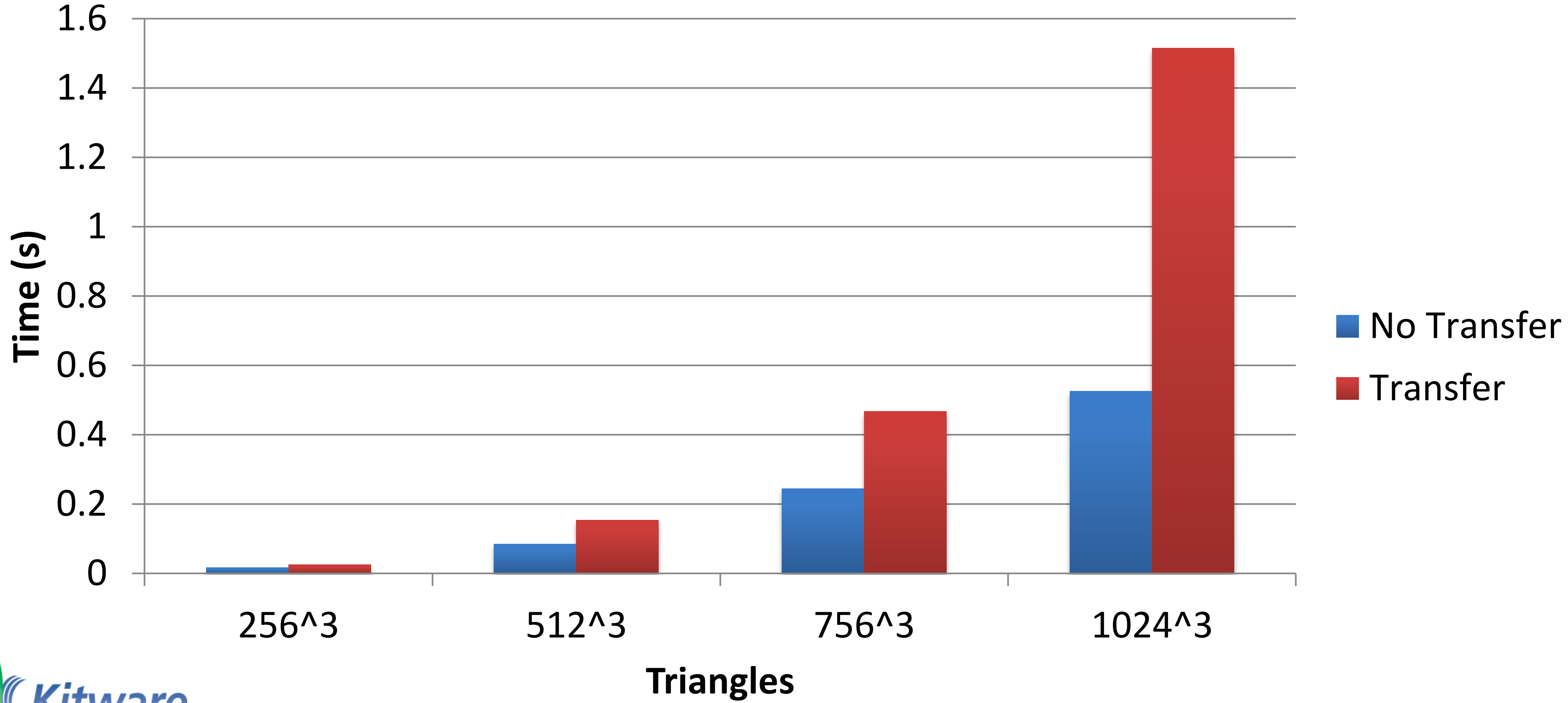
What We Have So Far

- Compiles with
  - gcc (4.8+), clang, msvc (2010+), icc, and pgi
- User Guide
- Ready for larger collaboration

2 x Intel Xeon CPU E5-2620 v3 @ 2.40GHz + NVIDIA Tesla K40c
Data: 1024^3 (floats)



2 x Intel Xeon CPU E5-2620 v3 @ 2.40GHz + NVIDIA Tesla K40c
Data: 1024^3 (floats)



Kitware

Future Directions

- Make custom rendering easier
- Improved support for mobile
- Improved support for multitouch
- Extend approaches to the web
- Optionally use new features (OpenGL 4.4)

Coprocessing/In-situ

- Use of VTK and VTK-m
  - Process data in place using VTK-m
  - Visualize and analyze using VTK
- Bringing highly parallelized visualization and analytics in science to all
- Create bridges between VTK and VTK-m

Thank You!

Marcus D. Hanwell

- mhanwell@kitware.com
- @mhanwell
- +MarcusHanwell

Robert Maynard

- robert.maynard@kitware.com
- @robertjmaynard

Checkout out Kitware @ www.kitware.com and VTK @ www.vtk.org

Please complete the Presenter Evaluation sent to you by email or through the GTC Mobile App. Your feedback is important!