Lessons from Building a Visualization Toolkit for Massively Threaded Architectures

Robert Maynard
Principal Engineer, Kitware

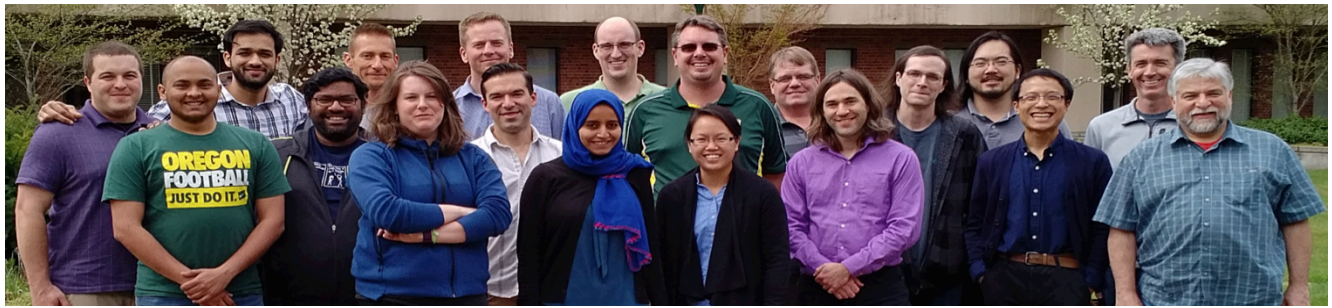
Kitware

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

VTK-m

A single place for the visualization community to collaborate, contribute, and leverage massively threaded algorithms.

Code Sprint, September 2015, LLNL



Code Sprint, April 2017, University of Oregon

Kitware

VTK-m

Reduce the challenges of writing highly concurrent algorithms by using data parallel algorithms

Done by writing ‘worklets’

```
struct GameOfLife : public vtkm::worklet::WorkletPointNeighborhood
{
  using ControlSignature = void(CellSetIn, FieldInNeighborhood prevstate, FieldOut state, FieldOut color);
  using ExecutionSignature = void(_2, _3, _4);

  template <typename NeighIn>
  VTKM_EXEC void operator()(const NeighIn& prevstate,
                            vtkm::UInt8& state,
                            vtkm::Vec<vtkm::UInt8, 4>& color) const
  {
    auto current = prevstate.Get(0, 0, 0);
    auto count = prevstate.Get(-1, -1, 0) + prevstate.Get(-1, 0, 0) + prevstate.Get(-1, 1, 0) +
      prevstate.Get(0, -1, 0) + prevstate.Get(0, 1, 0) + prevstate.Get(1, -1, 0) +
      prevstate.Get(1, 0, 0) + prevstate.Get(1, 1, 0);
    // Any live cell with fewer than two live neighbors dies, as if caused by under-population.
    // Any live cell with two or three live neighbors lives on to the next generation.
    // Any live cell with more than three live neighbors dies, as if by overcrowding.
    // Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.
```

Kitware

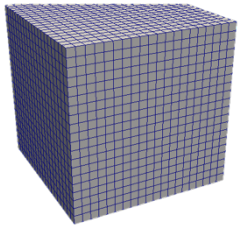
VTK-m

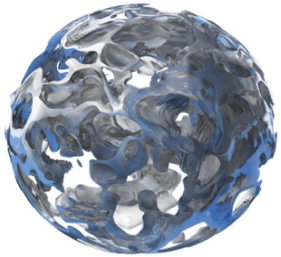
Reduce the challenges of writing highly concurrent algorithms by using data parallel algorithms

```
void NextFrame(vtkm::cont::ArrayHandle<vtkm::UInt8>& state,
               vtkm::cont::ArrayHandle<vtkm::Vec<vtkm::UInt8, 4>>& colors)
{
  vtkm::cont::ArrayHandle<vtkm::UInt8> prevstate;
  vtkm::cont::Algorithm(lastState, prevstate);

  vtkm::worklet::Invoker invoke;
  invoke(GameOfLife{}, prevstate, state, colors);
}
```

Kitware

VTK-m





Control

Filters

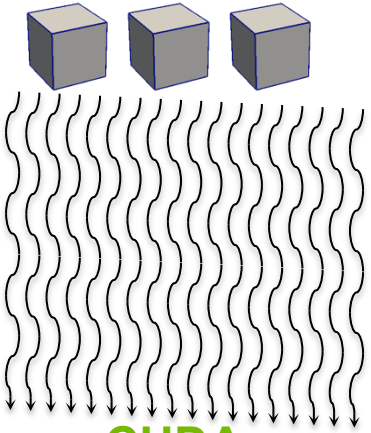
Control DataModel

Execution

Worklets

Execution DataModel

Data Parallel Algorithms
Arrays



CUDA

OpenMP

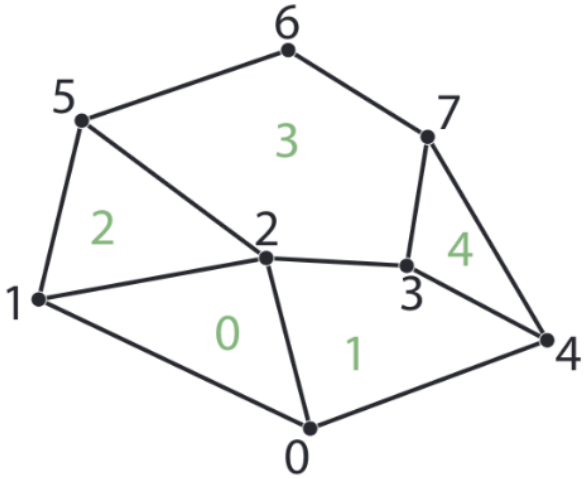
TBB

Kitware

VTK-m WorkletMapField

Iterates over any array (Point, Cell)

- Read/Write access
- Parallel for_each

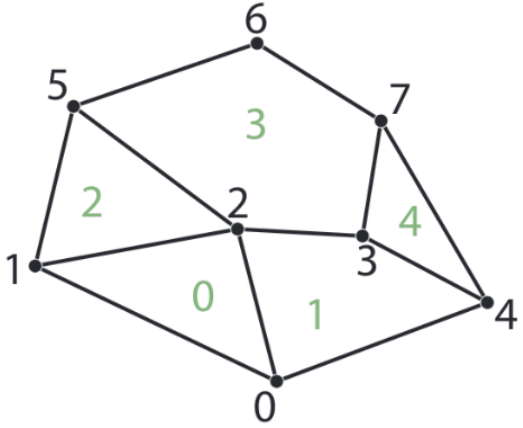


Kitware

VTK-m WorkletMapCellToPoint

Iterates over all points

- Read access to cell fields
- Read/Write access to point fields
- Point 3 has access to cells 1,3,4

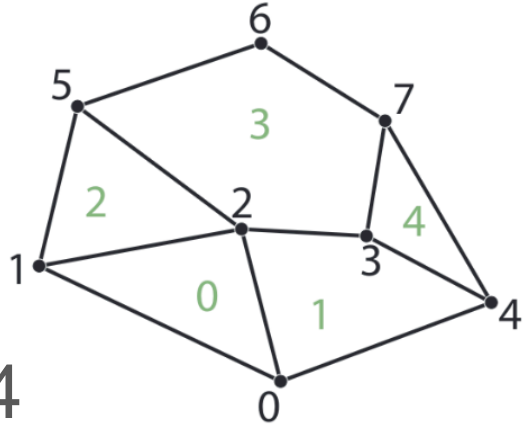


Kitware

VTK-m WorkletMapPointToCell

Iterates over all cells

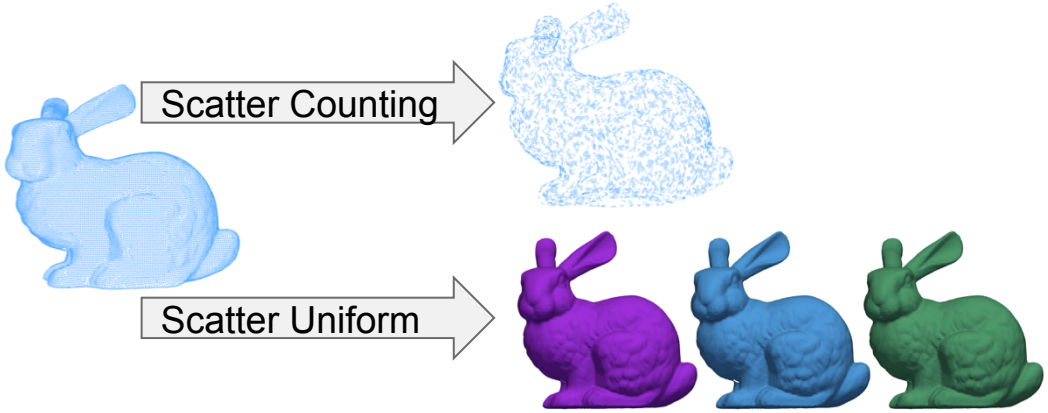
- Read access to point fields
- Read/Write access to cell fields
- Cell 1 has access to points 0,2,3,4



Kitware

Scattering

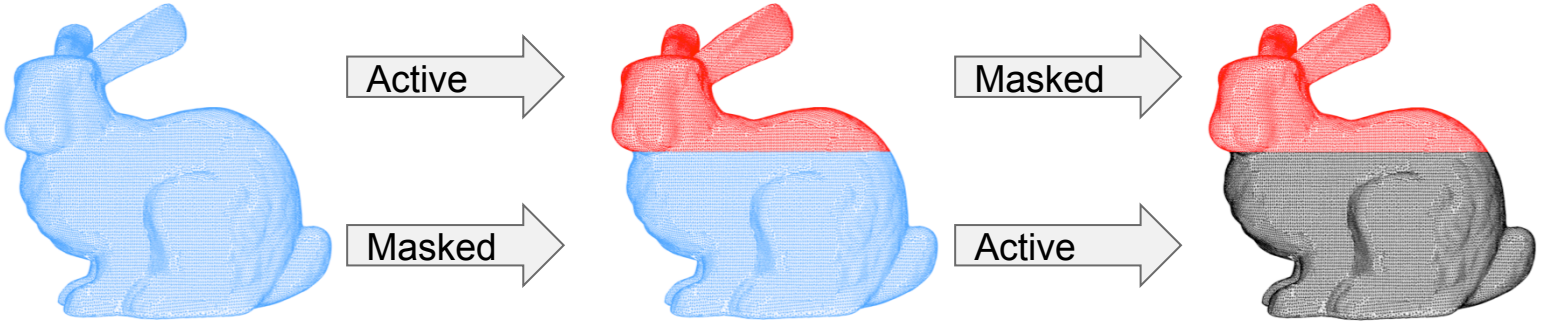
Many algorithms need more than 1 to 1 mapping. The operations might need to pass over elements that produce no value or the operation might need to produce multiple values for a single input element.



Kitware

Masking

Some algorithms need to be iterative on subsets of the input while maintaining a single output. For these kind of problems VTK-m provides the ability to enable/ disable a worklet execution based on a input mask.

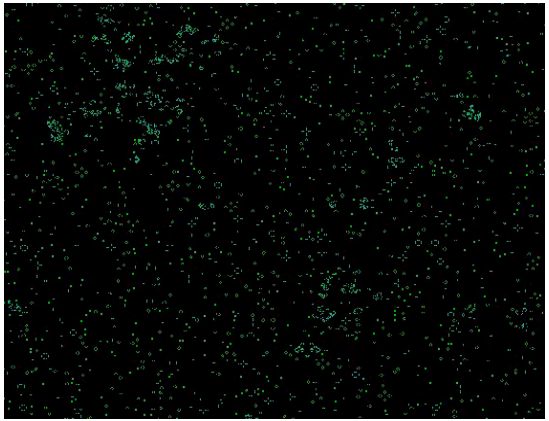


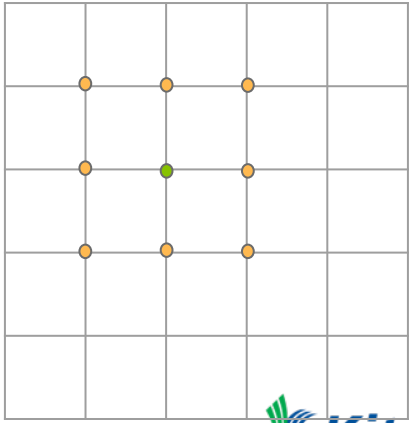
Kitware

WorkletPointNeighborhood

Iterates over all points

- Read access to points field neighborhood
- Write access to center point

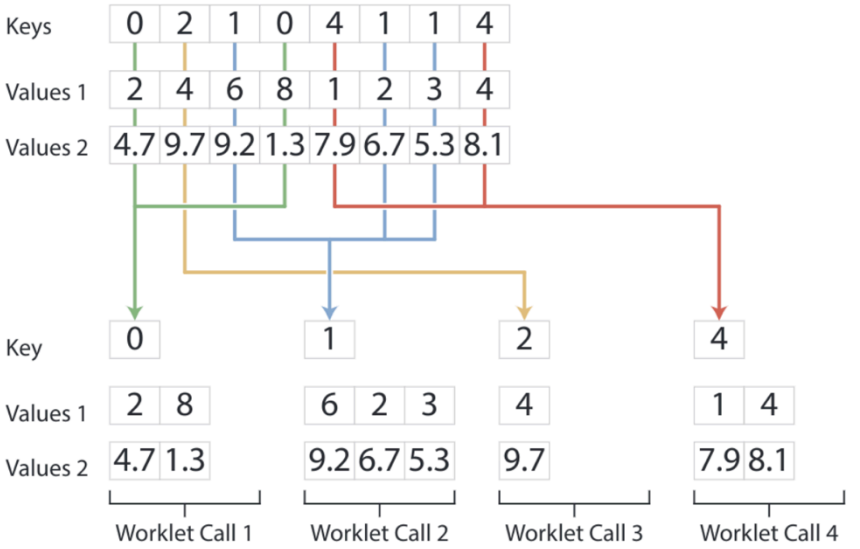




WorkletReduceByKey

Iterates over a key/value(s) array

- Read access to all values of a given key
- Write access for a given key



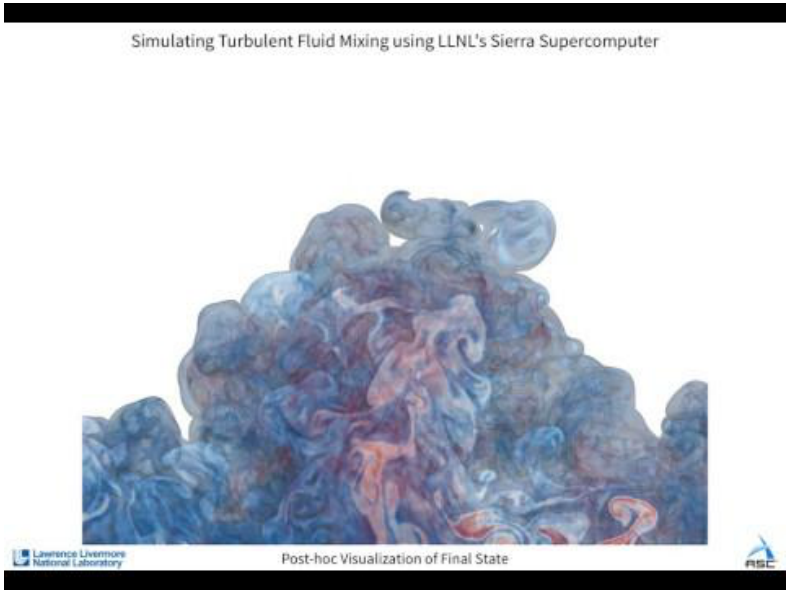
Reduce the challenges of writing highly concurrent algorithms by using data parallel algorithms

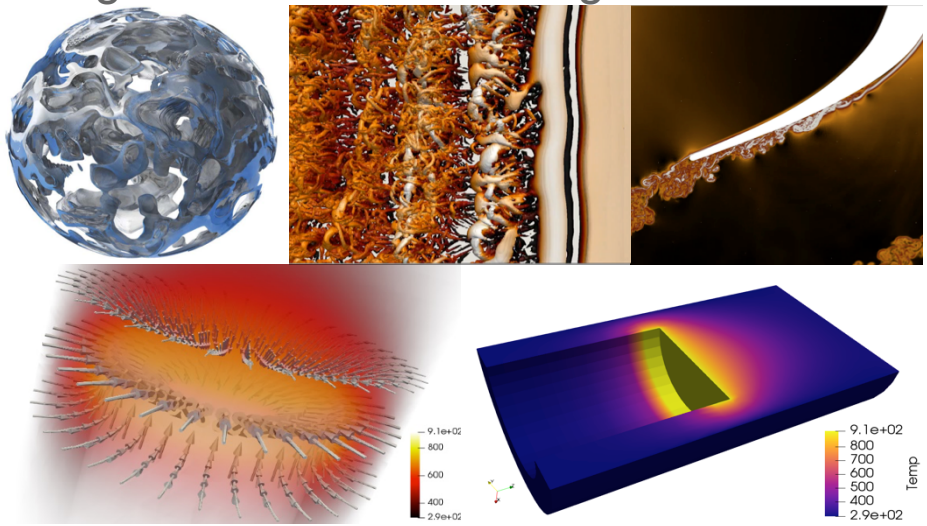
ForEach / ForEach3D
Transform
Sort / SortByKey
Reduce / ReduceByKey
Copy / CopyIf / CopySubRange
LowerBounds / UpperBounds
ScanInclusive / ScanInclusiveByKey
ScanExclusive / ScanExclusiveByKey
Unique / UniqueByKey

Kitware

VTK-m

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





Kitware

Simulations

In Situ Vis Library
(Integration with Sim)

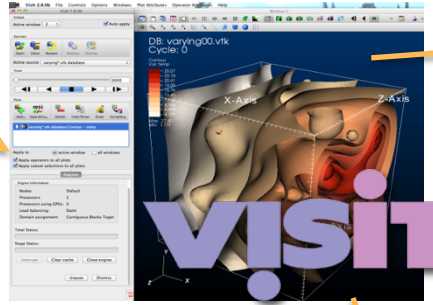
ParaView Catalyst

Libsim

Ascent

ADIOS

GUI / Parallel Management



Base Vis Library
(Algorithm Implementation)

VTK

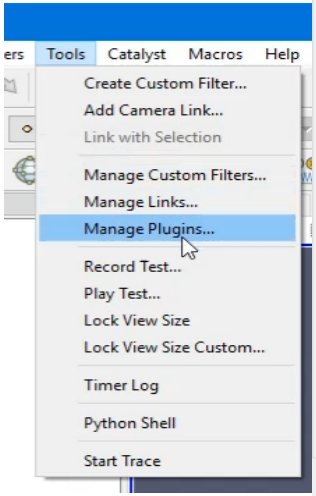
VTK-m

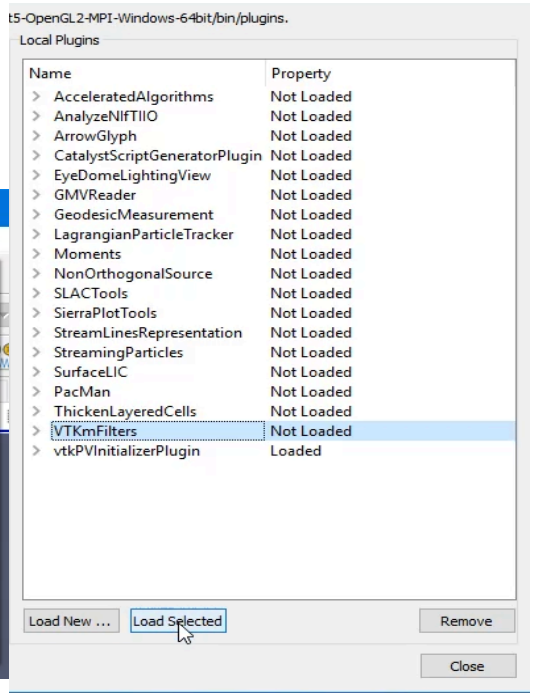
Multithreaded Algorithms
Processor Portability

Kitware

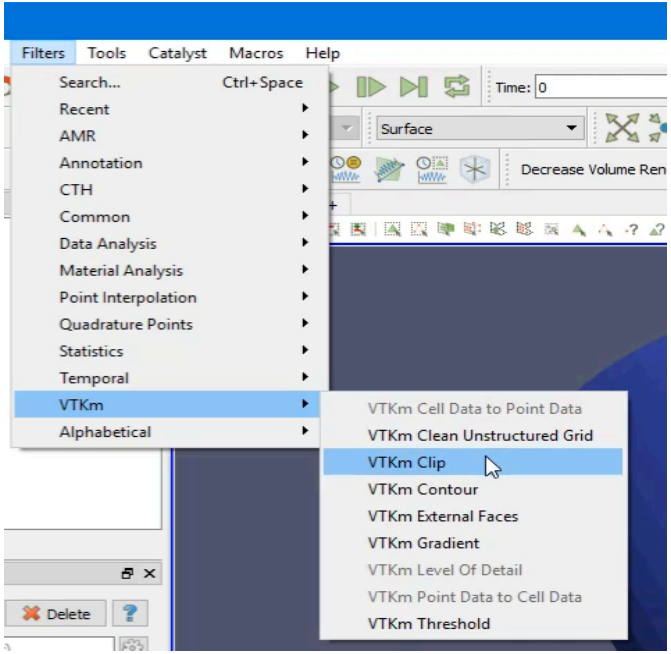
VTK-m In ParaView

1. Load VTK-m Plugin



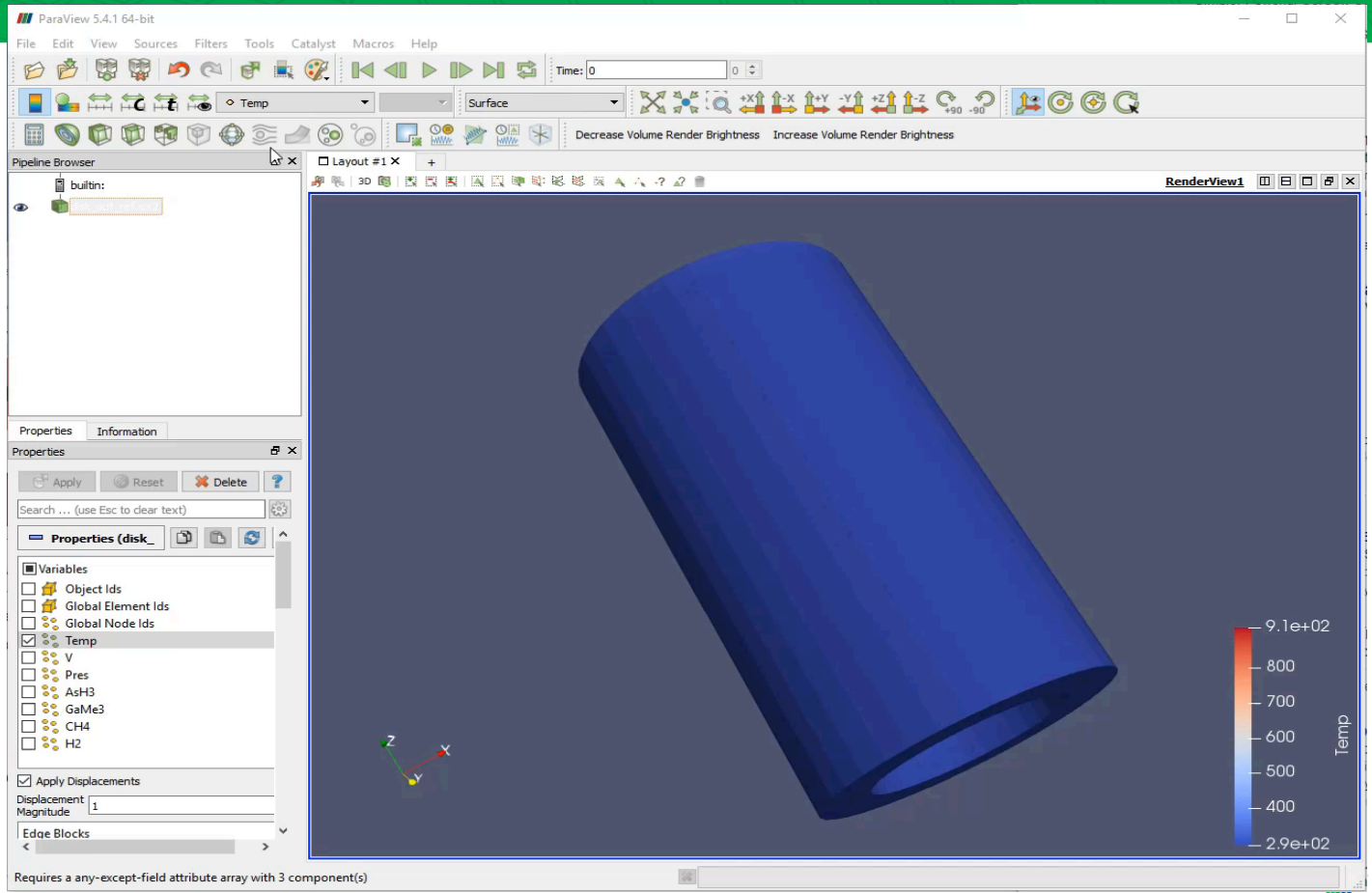


2. Use a VTK-m filter like any other



Slide Credit: Ken Moreland

Kitware

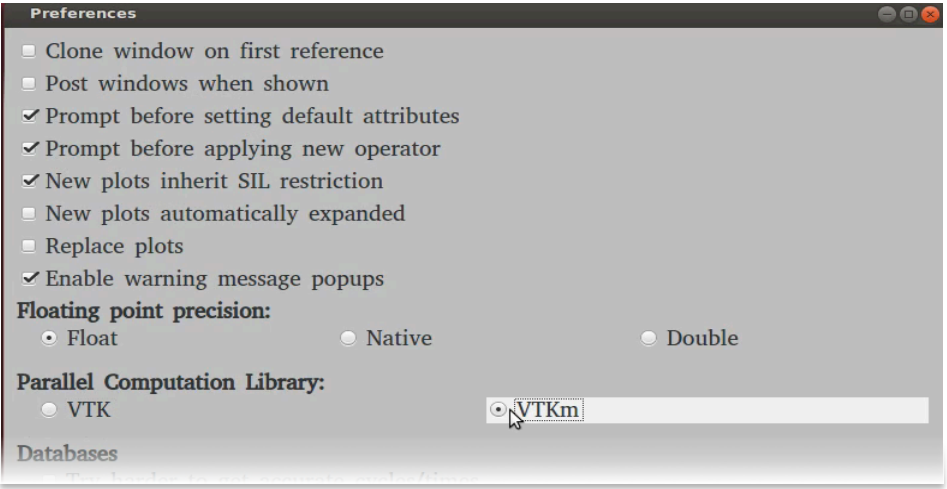


Slide Credit: Ken Moreland

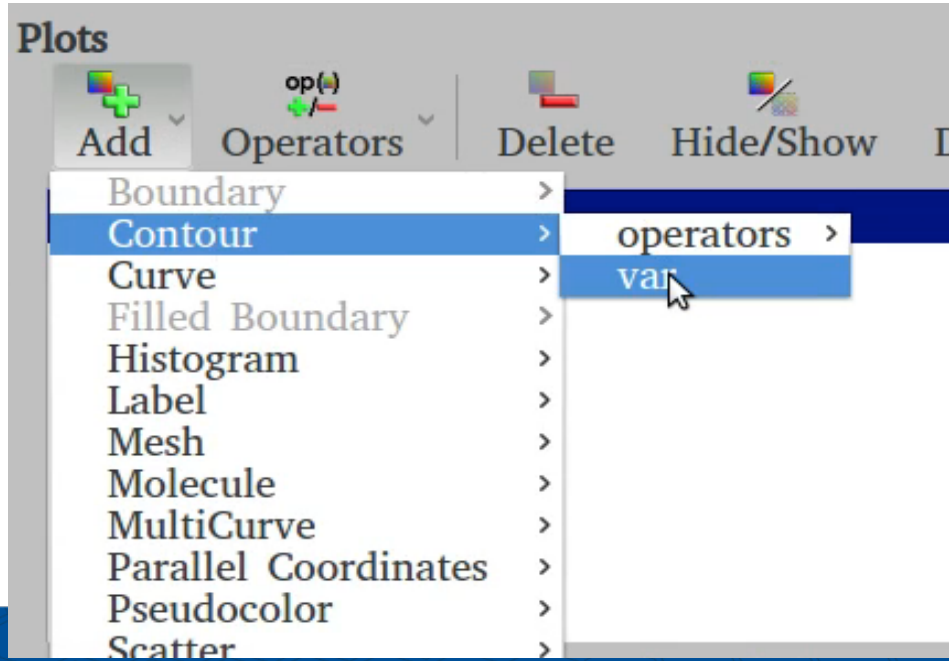
Kitware

VTK-m In VisIt

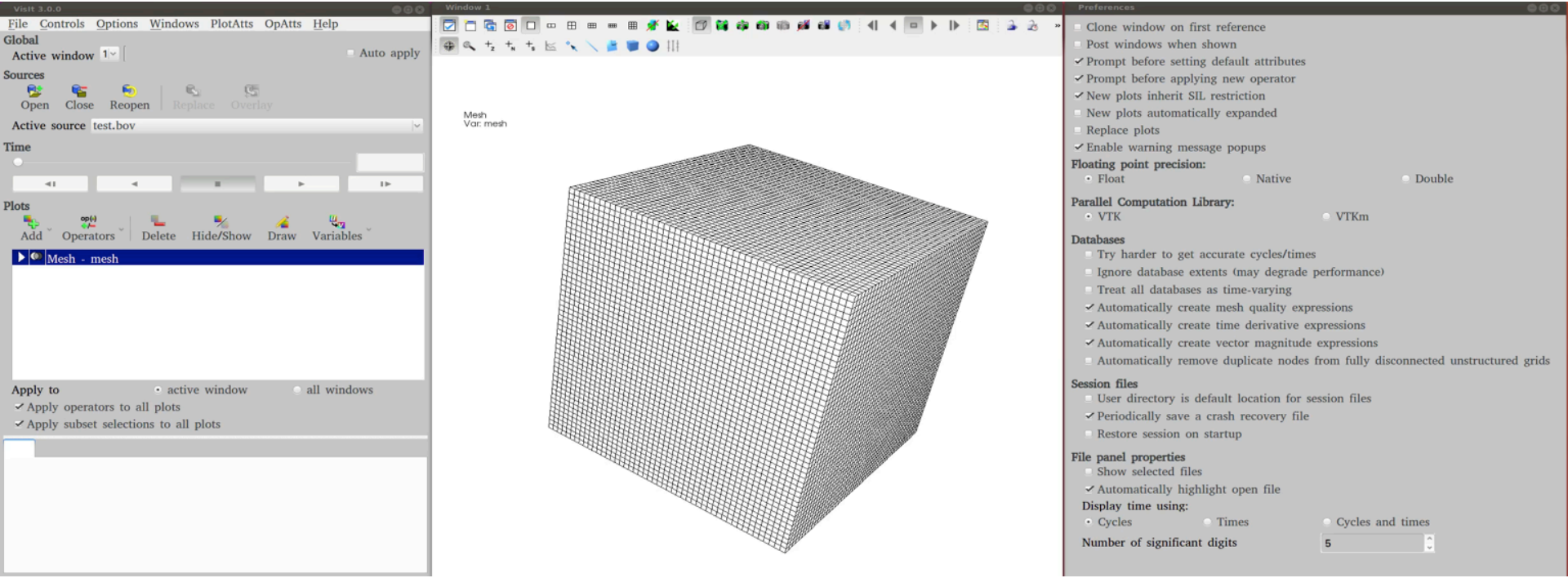
1. Turn on VTK-m in Preferences



2. Use VTK-m enabled plots as normal



Slide Credit: David Pugmire



Slide Credit: David Pugmire

Kitware

External Evolution

Kitware

Filters

- Cell Average
- Cell Measurements
- Clean Grid
- Clip by Field or Implicit Function
- Contour Trees
- External Faces
- Lagrangian
- Mask Points
- Point Average
- Point Elevation
- Probe
- Streamlines

Kitware

VTK-m Filters

- Extract Geometry, Points, Structured
- FieldToColors
- Gradient
- Histogram and Entropy
- Marching Cubes
  - Hex and Voxel Done
  - Other Cell Types In-Progress
- Split Sharp Edges
- Surface Normals
- Surface Simplification
- Tetrahedralize
- Threshold
- Triangulate
- Warp
- ZFP

Kitware

Worklet Control Signature

VTK-m no longer requires the list of allowed types for each worklet parameter

```
class Magnitude : public vtkm::worklet::WorkletMapField {
public:
  using ControlSignature = void(FieldIn<VecAll>, FieldOut<Scalar>);
  using ExecutionSignature = _2(_1);
```

```
class Magnitude : public vtkm::worklet::WorkletMapField {
public:
  using ControlSignature = void(FieldIn, FieldOut);
  using ExecutionSignature = _2(_1);
```

Kitware

Runtime Device Selection

VTK-m supports compilation of any number of device adapters in a single library.
Previously it was only possible to get runtime selection by jumping through hoops

```
struct LaunchMagnitude {
  template <typename DeviceAdapterTag, typename DesiredDevice, typename... Args>
  bool operator()(DeviceAdapterTag tag, DesiredDevice desired, Args &&... args) const {
    if(tag == desired)
    {
      vtkm::worklet::DispatcherMapField<Magnitude, DeviceAdapterTag> dispatcher;
      dispatcher.Invoke(std::forward<Args>(args)...);
      return true;
    }
    return false;
  }
};

  vtkm::cont::ArrayHandle<vtkm::Vec<float, 3>> input;
  vtkm::cont::ArrayHandle<float> output;
  vtkm::cont::DeviceAdapterTagCuda cuda;
  vtkm::cont::TryExecute(LaunchMagnitude{}, cuda, input, output);
```

Kitware

Runtime Device Execution

VTK-m has removed the Device template from all Dispatchers and instead builds all device versions and can easily switch between them

```
vtkm::cont::ArrayHandle<vtkm::Vec<float, 3>> input;
vtkm::cont::ArrayHandle<float> output;
vtkm::cont::DeviceAdapterTagCuda cuda;

vtkm::worklet::Invoker invoke(cuda);
invoke(Magnitude{}, input, output);
```

Kitware

Runtime Device Selection

ArrayHandle, Algorithms, Worklet, and Filter now all support runtime selection

```
auto runtime_device = vtkm::cont::make_DeviceAdapterId( 2 );
vtkm::cont::ArrayHandle<vtkm::Vec<float, 3>> input;
vtkm::cont::ArrayHandle<float> output;

{ // runtime selection
  vtkm::worklet::Invoker invoke(runtime_device);
  invoke(LightTask{}, input, output);
  vtkm::cont::Algorithm::Sort(runtime_device, output);
  invoke(HeavyTask{}, output);
}

{ // or mixed
  vtkm::worklet::Invoker openmp_exec(openmp);
  vtkm::worklet::Invoker cuda_exec(cuda);

  openmp_exec(LightTask{}, input, output);
  vtkm::cont::Algorithm::Sort(cuda, output);
  cuda_exec(HeavyTask{}, output);
}
```

Kitware

Runtime Device Tracking

Runtime selection supports the ability to use an Any device which selects the active device at runtime. Any supports graceful degradation for when a device crashes

```
auto runtime_device = vtkm::cont::make_DeviceAdapterId( ... );
{ // thread 1
  auto tracker = vtkm::cont::GetRuntimeDeviceTracker();
  tracker->ForceDevice(runtime_device);
  vtkm::worklet::Invoker invoke;
  invoke(LightTask{}, input, output);
  vtkm::cont::Algorithm::Sort(output);
  invoke(HeavyTask{}, output);
}

{ // thread 2
  auto tracker = vtkm::cont::GetRuntimeDeviceTracker();
  vtkm::worklet::Invoker invoke;
  invoke(LightTask{}, input, output);
  vtkm::cont::Algorithm::Sort(output);
  invoke(HeavyTask{}, output);
}
```

Kitware

Future Runtime Device Tracking

Since VTK-m defers location of execution to runtime this opens up future research work on task locality

- Should execution over small domains happen in serial?
- When should execution move to the memory space of the allocation?
  - Can we map this to multi-gpu machines and allocations?
- What to do when inputs are spread across multiple memory spaces?

Logging

For better reporting of runtime performance and errors VTK-m has a fully integrated logging framework. Allows us to log:

- Errors
- Warnings
- Dynamic Cast Failures
- Control Side Memory Allocations
- Execution Side Memory Allocations
- Memory Transfers
- Performance

Kitware

Logging

```
(7.977s) [410A7700]              Filter.hxx:284   Perf| . . { Filter: 'vtkm::filter::Gradient'
(7.977s) [410A7700]              Filter.hxx:304   Perf| . . . { Filter (MultiBlock): 'vtkm::filter::Gradient'
(7.977s) [410A7700]       DispatcherBase.h:670   Perf| . . . . { Invoking Worklet: '...StructuredPointGradient<vtkm::Vec<float, 3> >'
(7.978s) [410A7700]       CudaAllocator.cu:172   MemE| . . . . . Allocated CUDA array of 192.00 MiB (201326592 bytes) at 0x7f33f0000000.
(7.978s) [410A7700]ExecutionArrayInterface:150   MemT| . . . . . Copying host --> CUDA dev: 192.00 MiB (201326592 bytes)
(8.214s) [410A7700]       DispatcherBase.h:670   Perf| . . . . } 0.237068 s: Invoking Worklet: '...StructuredPointGradient<vtkm::Vec<float, 3> >'
(8.215s) [410A7700]              Filter.hxx:304   Perf| . . . } 0.237174 s: Filter (MultiBlock): 'vtkm::filter::Gradient'
(8.215s) [410A7700]              Filter.hxx:284   Perf| . . } 0.237196 s: Filter: 'vtkm::filter::Gradient'
```

```
finished processing a multi-block
terminate called after throwing an instance of 'std::system_error'
  what():  hi gtc!: Numerical argument out of domain

Loguru caught a signal: SIGABRT
Stack trace:
15      0x7fc1472af88f clone + 63
14      0x7fc1482d16db /lib/x86_64-linux-gnu/libpthread.so.0(+0x76db) [0x7fc1482d16db]
13      0x7fc147bf257f /usr/lib/x86_64-linux-gnu/libstdc++.so.6(+0xbd57f) [0x7fc147bf257f]
12      0x5610d41ccfd4 ./examples/multi_backend/MultiBackend(+0x42efd4) [0x5610d41ccfd4]
11      0x5610d41cd024 ./examples/multi_backend/MultiBackend(+0x42f024) [0x5610d41cd024]
10      0x5610d41cd0cb ./examples/multi_backend/MultiBackend(+0x42f0cb) [0x5610d41cd0cb]
9       0x5610d41cb1a4 ./examples/multi_backend/MultiBackend(+0x42d1a4) [0x5610d41cb1a4]
8       0x5610d41cbe70 ./examples/multi_backend/MultiBackend(+0x42de70) [0x5610d41cbe70]
7       0x5610d41ca634 ./examples/multi_backend/MultiBackend(+0x42c634) [0x5610d41ca634]
6       0x7fc147bc7c74 /usr/lib/x86_64-linux-gnu/libstdc++.so.6(+0x92c74) [0x7fc147bc7c74]
5       0x7fc147bc7a41 /usr/lib/x86_64-linux-gnu/libstdc++.so.6(+0x92a41) [0x7fc147bc7a41]
4       0x7fc147bc7a06 /usr/lib/x86_64-linux-gnu/libstdc++.so.6(+0x92a06) [0x7fc147bc7a06]
3       0x7fc147bc18b7 /usr/lib/x86_64-linux-gnu/libstdc++.so.6(+0x8c8b7) [0x7fc147bc18b7]
2       0x7fc1471ce801 abort + 321
1       0x7fc1471cce97 gsignal + 199
0       0x7fc1482dc890 /lib/x86_64-linux-gnu/libpthread.so.0(+0x12890) [0x7fc1482dc890]
2019-03-06 14:03:50.434 (   9.819s) [        466AE700]                      :0     FATL| Signal: SIGABRT
```

Kitware

Original Filter Policy Design

Filter Policies are how callers of VTK-m control what compile time type expansions will be done for:

- CellSets [Structured, Unstructured, …]
- Field Types [are they float, double, vec3f?]
- Field Storage [Basic, Counting, Implicit, …]
- Coordinates Types
- Coordinates Storage

Kitware

Original Filter Policy Design

```
template <typename Derived>
struct PolicyBase
{
  using FieldTypeList = vtkm::ListTag<vtkm::Int32,         vtkm::Int64,
                                      float,               double,
                                      vtkm::Vec<float, 3>, vtkm::Vec<double, 3>>;

  using FieldStorageList = vtkm::ListTag<vtkm::cont::StorageTagBasic>;


  using StructuredCellSetList =   vtkm::cont::CellSetListTagStructured;
  using UnstructuredCellSetList = vtkm::cont::CellSetListTagUnstructured;
  using AllCellSetList =          vtkm::ListJoin<vtkm::cont::CellSetListTagStructured,
                                                 vtkm::cont::CellSetListTagUnstructured>;

  using CoordinateTypeList = vtkm::ListTag<vtkm::Vec<float, 3>, vtkm::Vec<double, 3>>;
  using CoordinateStorageList = vtkm::ListTag< vtkm::cont::StorageTagBasic,
                                               vtkm::cont::UniformPointCoordinatesStorageTag,
                                               vtkm::cont::ArrayHandleCompositeF32StorageTag,
                                               vtkm::cont::ArrayHandleCompositeF64StorageTag,
                                               vtkm::cont::ArrayHandleCartesianProductF32StorageTag,
                                               vtkm::cont::ArrayHandleCartesianProductF64StorageTag>;

  using DeviceAdapterList = vtkm::ListTagBase<vtkm::cont::DeviceAdapterTagCuda,
                                              vtkm::cont::DeviceAdapterTagTBB,
                                              vtkm::cont::DeviceAdapterTagOpenMP,
                                              vtkm::cont::DeviceAdapterTagSerial
                                              >;
```

Kitware

New Filter Policy Design

```
template <typename Derived>
struct PolicyBase
{
  using FieldTypeList = vtkm::ListTag<vtkm::Int32,          vtkm::Int64,
                                      float,                double,
                                      vtkm::Vec<float, 3>, vtkm::Vec<double, 3>>;

  using StructuredCellSetList =   vtkm::cont::CellSetListTagStructured;
  using UnstructuredCellSetList = vtkm::cont::CellSetListTagUnstructured;
  using AllCellSetList =          vtkm::ListJoin<vtkm::cont::CellSetListTagStructured,
                                                 vtkm::cont::CellSetListTagUnstructured>;
};
```

Virtual Arrays

VTK-m has identified a need to have certain execution objects leverage virtual methods. Things such as array handle storage, implicit functions and coordinate systems now use virtuals.

```
CoordinateTypeList = vtkm::ListTag<vtkm::Vec<float, 3>, vtkm::Vec<double, 3>>;
CoordinateStorageList = vtkm::ListTag< vtkm::cont::StorageTagBasic,
                                       vtkm::cont::UniformPointCoordinatesStorageTag,
                                       vtkm::cont::ArrayHandleCompositeF32StorageTag,
                                       vtkm::cont::ArrayHandleCompositeF64StorageTag,
                                       vtkm::cont::ArrayHandleCartesianProductF32StorageTag,
                                       vtkm::cont::ArrayHandleCartesianProductF64StorageTag>;
```

7 types

```
vtkm::cont::ArrayHandleVirtualCoordinates coords;
  -> vtkm::cont::ArrayHandleVirtual< vtkm::Vec<float, 3> >
  -> vtkm::cont::ArrayHandleVirtual< vtkm::Vec<double, 3> >
  -> vtkm::cont::ArrayHandleUniformPointCoordinates
```

3 types

Kitware

New++ Filter Policy [In Design]

VTK-m currently only exactly matches FieldTypes. Going forward we are going to cast to best matching and provide explicit de-virtualization.

```
template <typename Derived>
struct PolicyBase
{ //cast to best match
  using FieldTypeList = vtkm::ListTag<vtkm::Int32,          vtkm::Int64,
                                      float,                double,
                                      vtkm::Vec<float, 3>, vtkm::Vec<double, 3>>;
  using FieldOptimize = vtkm::ListTag< vtkm::ArrayHandle<vtkm::Vec<double, 3>> >;

  using StructuredCellSetList =   vtkm::cont::CellSetListTagStructured;
  using UnstructuredCellSetList = vtkm::cont::CellSetListTagUnstructured;
  using AllCellSetList =          vtkm::ListJoin<vtkm::cont::CellSetListTagStructured,
                                                 vtkm::cont::CellSetListTagUnstructured>;
};
```

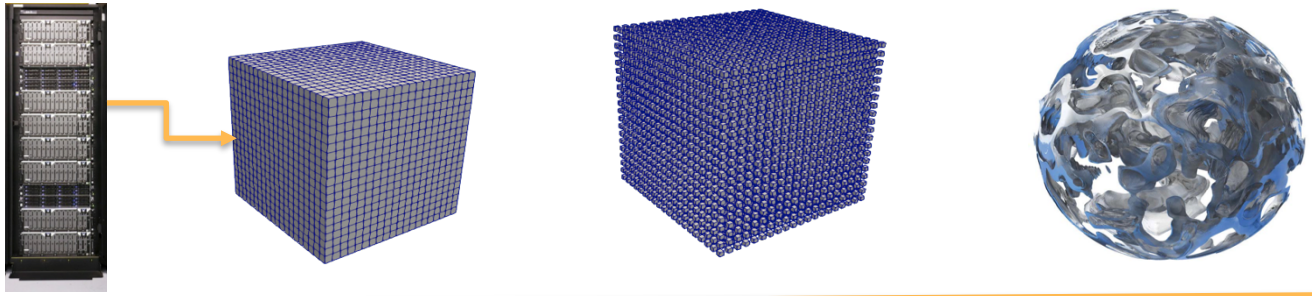
MultiBlock

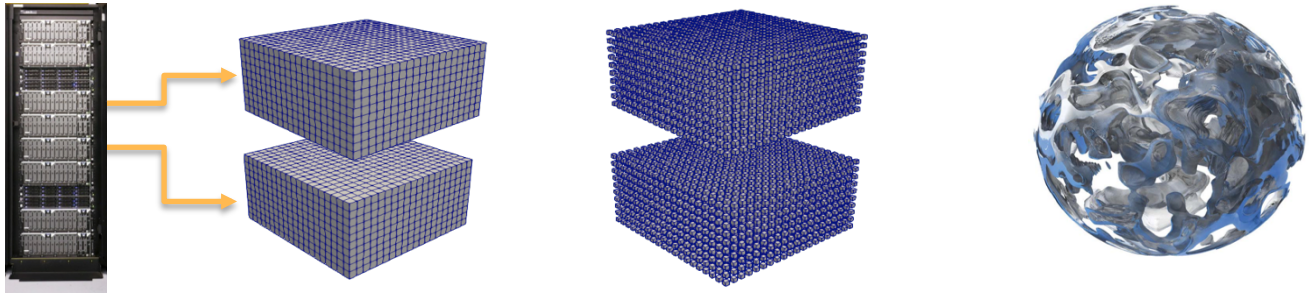
VTK-m MultiBlock is very similar to vtkPartitionedDataSet

- VTK-m MultiBlock entries can only be DataSets, no support for nested MultiBlocks
- In VTK-m a MultiBlock can span multiple nodes (MPI/DIY), but a block must be fully contained on a single node

VTK-m

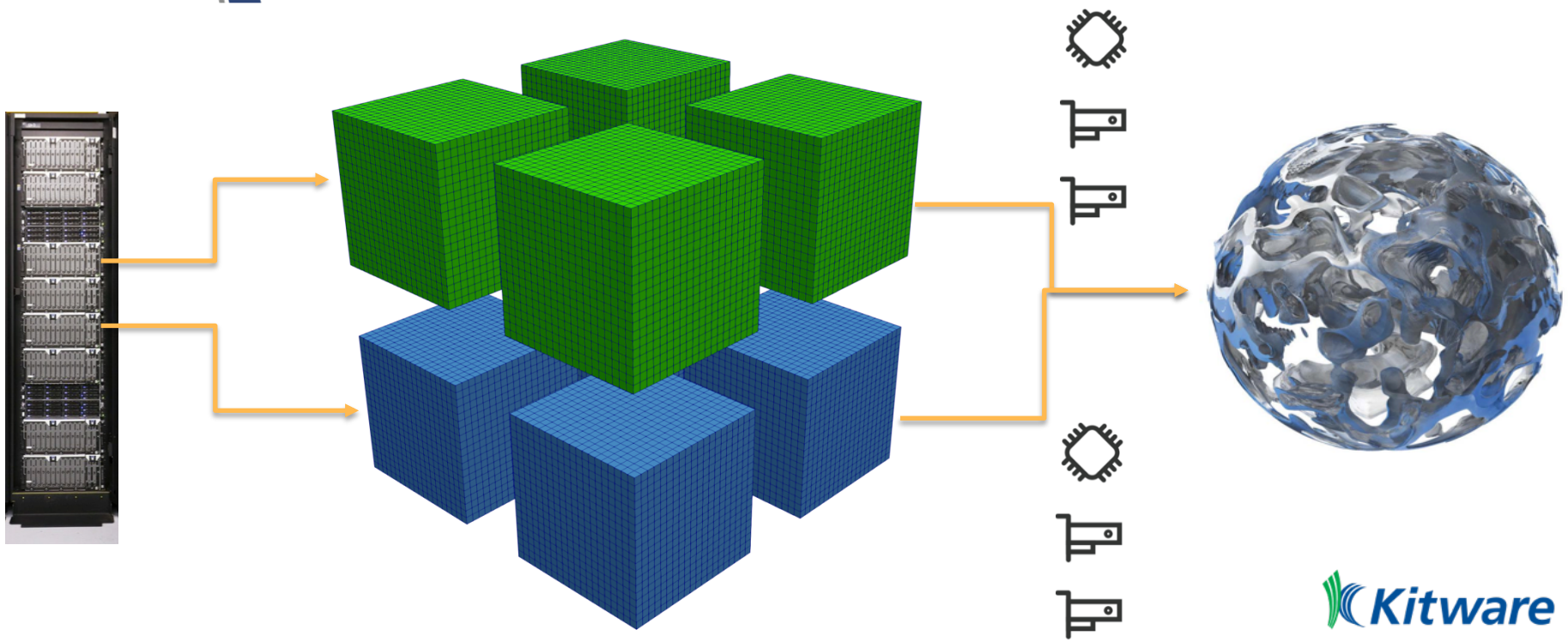
Hybrid Parallelism





Kitware

VTK-m Drive Towards Hybrid Async



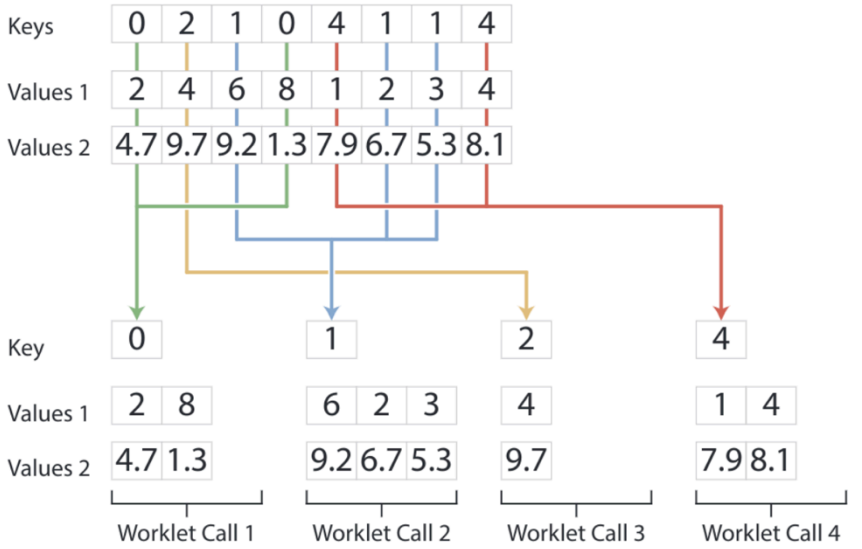
Kitware

```
auto filters = [=](const Executor& exec) {
  vtkm::worklet::Invoker invoke{exec};
  invoke(Magnitude{}, input, output);
  invoke(Sin{}, output);
};

vtkm::cont::MultiBlock mb = ...;
auto device = vtkm::cont::make_DeviceAdapterId(2);

//iterate block in parallel on CPU
auto computed = std::async(std::execution::par, mb.begin(), mb.end()
    [](auto block)
    {
      auto input = block.GetField("pressure");
      vtkm::cont::ArrayHandle<float> output;
      // launch all versions in async on a runtime selected device
      std::async(device.executor(), filters);
      block.AddField(output, "result");
    });

computed.get(); //all blocks executed
```

WorkletReduceByKey

VTK-m provides a custom reduce by key since we needed the following functionality:

- Multi value reduction
- Access to all values per key

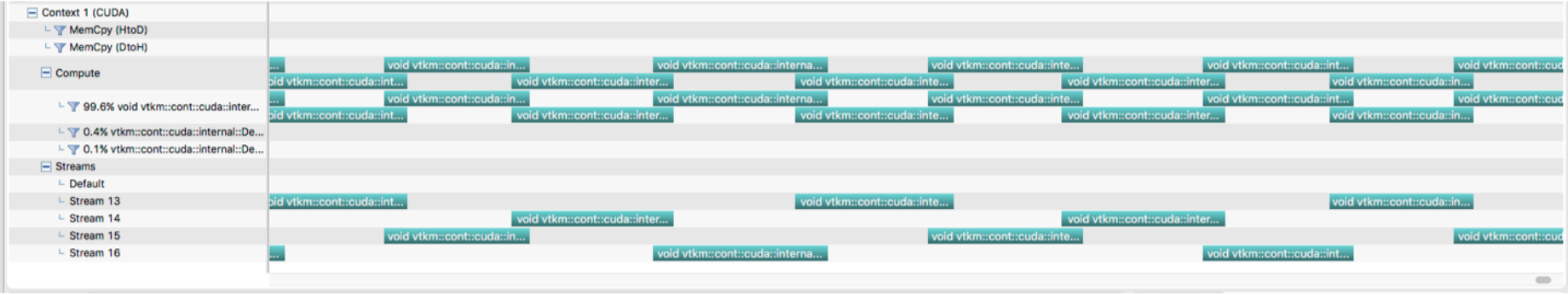


Internal Evolution

Kitware

CUDA Streams

When ever VTK-m executes using the CUDA device adapter all kernels and memory transfers now use per-thread default streams explicitly



This work allows for better in-situ integration, and for VTK-m to provide the option of coarse grained block level parallelism.

Kitware

CUDA

VTK-m ArrayHandle now properly handles users passing CUDA allocated pointers for input data.

- No extra data transfers or copies
- If UVM allocated can also be used with other devices

When VTK-m executes on Pascal+ hardware all device memory will be allocated using UVM.

- Includes hints to the UVM system if the memory is read, write, or r+w
- If the ArrayHandle doesn't have host data, will use the UVM memory
- Controllable with environment variables

Kitware

CUDA

VTK-m ArrayHandle reads now use __ldg loads automatically on any read only input

VTK-m tries for all cuda operations to happen asynchronously Allows for overlapping control and device

- Goal of reducing host / device synchronizations.
  - We use Thrust for parallel primitives (expect worklet launches)
  - We don’t sync after each worklet
  - We only use event syncs
  - We explicitly event sync only for host memory access
  - We batch small cuda memory free’s

Kitware

CUDA Lookup Tables

VTK-m uses lots of predefined lookup tables
These are challenging to write correctly when you want the same table to be used for host and device (E.3.13. Const-qualified variables && F.3.16.5. Constexpr variables)

```
// cuda 7.5 doesn't support static const or static constexpr variables
// that exist inside methods or classes, so in those cases we have to use
// just constexpr
#if __CUDACC_VER_MAJOR__ < 8
#define CONSTEXPR_ARRAY constexpr
// cuda 8-9? doesn't support static constexpr pointers/fixed size arrays
// that exist inside methods or classes, so in those cases we gracefully
// fall back to static const
#elif __CUDACC_VER_MAJOR__ < 10
#define CONSTEXPR_ARRAY static const
#else
#define CONSTEXPR_ARRAY static constexpr
#endif

__host__ __device__ int FaceLookUp(int x, int y)
{
  CONSTEXPR_ARRAY int faces[5][3] = {
    //offset into shapes face list, num faces, and num indices
    { 0, 6, 8 },  //hex
    { 6, 4, 4 },  //tet
    { 10, 5, 6 }, //wedge
    { 15, 5, 5 }, //pyramid
    { -1, 0, 0 }  //unsupported shape
  };
  return faces[x][y];
}
```

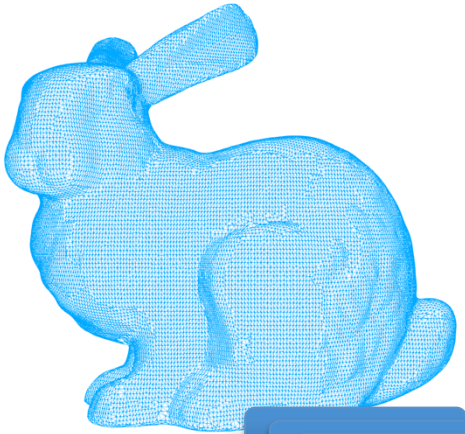
Kitware

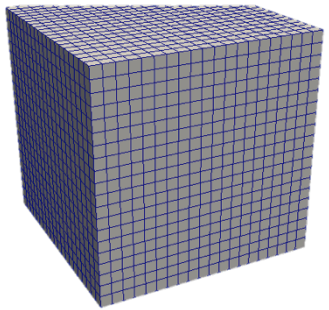
CUDA Lookup Tables

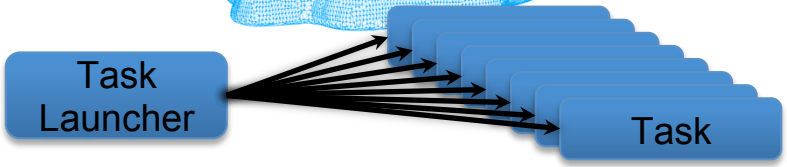
```
// cuda 7.5 doesn't support static const or static constexpr variables
// that exist inside methods or classes, so in those cases we have to use
// just constexpr
#if __CUDACC_VER_MAJOR__ < 8
#define CONSTEXPR_ARRAY constexpr
// cuda 8-9? doesn't support static constexpr pointers/fixed size arrays
// that exist inside methods or classes, so in those cases we gracefully
// fall back to static const
#elif __CUDACC_VER_MAJOR__ < 10
#define CONSTEXPR_ARRAY static const
#else
#define CONSTEXPR_ARRAY static constexpr
#endif

__host__ __device__ int FaceLookUp(int x, int y)
{
  CONSTEXPR_ARRAY int faces[5][3] = {
    //offset into shapes face list, num faces, and num indices
    { 0, 6, 8 },  //hex
    { 6, 4, 4 },  //tet
    { 10, 5, 6 }, //wedge
    { 15, 5, 5 }, //pyramid
    { -1, 0, 0 }  //unsupported shape
  };
  return faces[x][y];
}
```

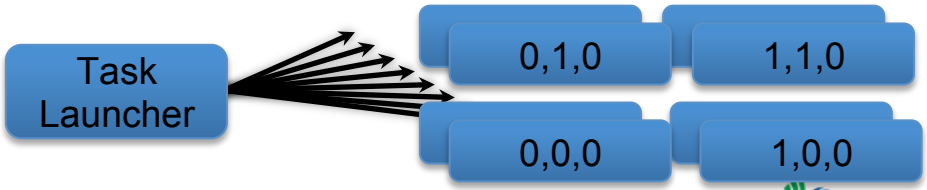
CUDA Worklet Execution

VTK-m Topology based worklets are always executed in the context of a topology.









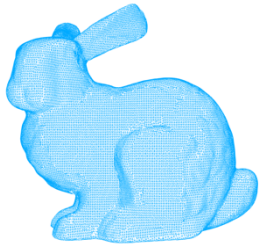
Kitware

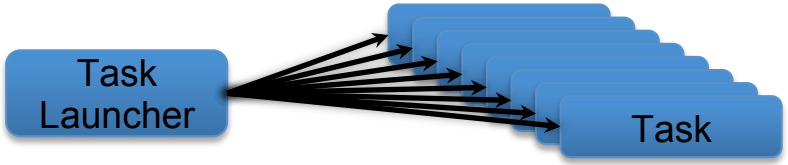
CUDA 1D Worklet Execution

VTK-m has explored using different strategies over the years for 1D execution.

- We use grid stride loops
  - We launch a fixed number of blocks and threads and stride over the total work
  - Number of blocks is based on a function of the number of SM's (32 per)
  - We use 128 threads per block

- We want as many register per thread as our worklets are 'large'



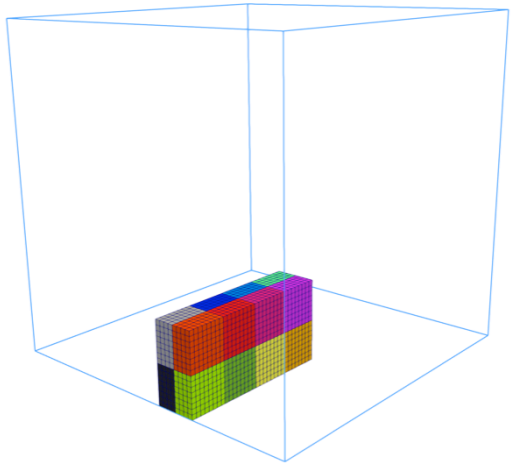


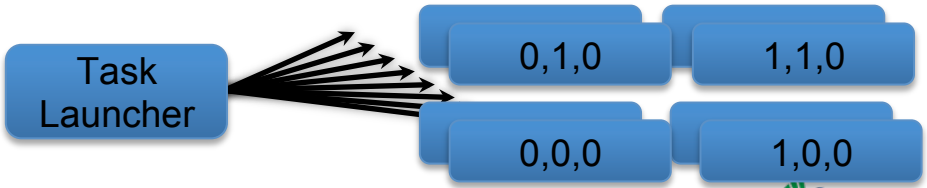
Kitware

CUDA 3D Worklet Execution

VTK-m uses a similar strategies over the years for 3D execution.

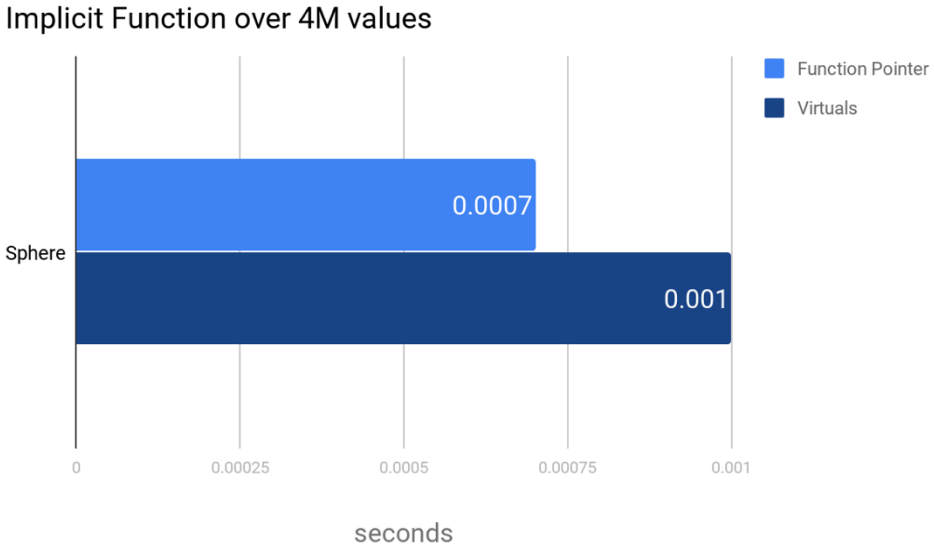
- We use grid stride loops
  - Number of blocks is based on a function of the number of SM's (32 per)
  - We use 256 threads per block in a <8,8,4> layout





Kitware

Virtual Methods




```
template <typename DerivedType, typename DeviceTag>
void move_to_device(DerivedType* host_side, DerivedType*& exec_side, DeviceTag
{
  // Allocate a buffer on the device
  DerivedType* state; cudaMalloc(&state, sizeof(DerivedType));
  // Copy host side members state
  cudaMemcpy(state, host_side, sizeof(DerivedType), cudaMemcpyHostToDevice);
  // Allocate memory for the object
  cudaMalloc(&exec_side, sizeof(DerivedType));
  // Initialize the device object using placement new
  detail::PlacementNewKernel<<<1, 1>>>(exec_side, state);
  // Clean up intermediate copy
  cudaFree(state);
}
```

CUDA: NVIDIA GP100 TBB: 2x Intel Xeon CPU E5-2620 v3 [24 cores]

Kitware

Atomic Performance

VTK-m originally avoided using atomics due to presumptions on performance. Starting in 2018 we have slowly moved algorithms over to atomics on a case by case basis

CellToPoint Table Gen	Time (s)				Mem (GiB)
Backend	**Serial**	**TBB**	**OpenMP**	**CUDA**	
VTK	2.535	(N/A)	(N/A)	(N/A)	2.711
VTK-m (Sort)	17.940	8.169	8.125	1.606*	8.166*
VTK-m (Atomic Histogram)	6.673	1.428	1.445	0.547	2.505

CUDA: Quadro K5100M CPU: Intel Core i7-4710MQ CPU @ 2.50GHz

Kitware

Conformance && Performance

Kitware

Testing

Nightly *20 builds*

		Update	Configure		Build		Test				
Site	Build Name	Revision	Error	Warn	Error	Warn	Not Run	Fail	Pass	Time	Start Time
aaargh.kitware	Linux-EL7_Intel-17.0.4	bdcb76	0	0	0	0	0	1	382	25s	6 hours ago
dejagore.kitware	Ubuntu-GCC-5.4	bdcb76	0	0	0	1 +1 -1	0	0	383	9s	12 hours ago
dejagore.kitware	Ubuntu-Clang-3.4-64bit	bdcb76	0	0	0	0	0	0	383	9s	12 hours ago
dejagore.kitware	Ubuntu-Clang-3.4-32bit	bdcb76	0	0	0	0	0	0	248	32s	12 hours ago
dejagore.kitware	Ubuntu-GCC-4.8	bdcb76	0	0	0	0	0	0	248	7s	12 hours ago
dragnipur.kitware	OSXHighSierra-AppleClang-9.1	bdcb76	0	0	0	0	0	0	248	9s	14 hours ago
delve.kitware	Windows8.1-VS-2015-CMake-3.8	bdcb76	0	0	0	0	0	0	248	2 1m 52s	15 hours ago
delve.kitware	Windows8.1-VS-2015-TBB	bdcb76	0	0	0	0	0	0	383	6 2m 47s	15 hours ago
dejagore	bdcb76ca-build282-[linux-static-debug+64bit_ids+clang+cuda_host_gcc_5+cuda_native]	bdcb76	0	0	0	0	0	0	387	2m 3s	16 hours ago
dejagore	bdcb76ca-build636-[linux-shared-debug+32bit_ids+gcc-4.8+logging+osmesa+tbb]	bdcb76	0	0	0	0	0	0	383	46s	16 hours ago
osheim	bdcb76ca-build757-[windows-static-release+logging+ninja-vs-2017+tbb]	bdcb76	0	0	0	0	0	0	383	14s	16 hours ago
delve	bdcb76ca-build508-[windows-shared-debug+32bit_ids+examples+logging+ninja-vs-2015+tbb]	bdcb76	0	0	0	0	0	0	383	2m 36s	16 hours ago
dragnipur	bdcb76ca-build92-[osx-shared-debug+32bit_ids+clang+examples+logging+ninja+tbb]	bdcb76	0	0	0	0	0	0	383	44s	16 hours ago
osheim	bdcb76ca-build1277-[windows-shared-debug+benchmark+examples+ninja-vs-2017+tbb]	bdcb76	0	0	0	0	0	0	383	2m 30s	17 hours ago
dragnipur	bdcb76ca-build95-[osx-static-release+64bit_ids+benchmark+clang+ninja+tbb]	bdcb76	0	0	0	0	0	0	383	8s	17 hours ago
delve	bdcb76ca-build485-[windows-shared-release+ninja-vs-2015+tbb]	bdcb76	0	0	0	0	0	0	383	12s	17 hours ago
renar	bdcb76ca-build87-[linux-static-release+64bit_ids+benchmark+cuda_volta+gcc-6+logging+mpi+ninja+openmp+tbb]	bdcb76	0	0	0	0	0	0	657	1m 17s	17 hours ago
renar	bdcb76ca-build96-[linux-static-debug+64bit_ids+examples+gcc-7+logging+mpi+ninja+openmp+tbb]	bdcb76	0	0	0	0	0	0	518	4m 41s	17 hours ago
adora	bdcb76ca-build81-[linux-static-release+benchmark+cuda_host_gcc_5+cuda_native+examples+gcc+ninja+tbb]	bdcb76	0	0	0	0	0	0	522	45s	17 hours ago
adora	bdcb76ca-build84-[linux-static-release+cuda_host_gcc_5+cuda_native+examples+gcc+logging+ninja]	bdcb76	0	0	0	0	0	0	387	36s	17 hours ago

Items per page All

itware

Testing

- Testing is used to catch serious changes in baseline performance

6 tests failed the time status check.

Name	Status	Time Status	Time	Details	History	Summary
CopyrightStatement	Passed	Failed	20s 170ms	Completed	Unstable	Stable
UnitTestBoundingIntervalHierarchySERIAL	Passed	Failed	22s 670ms	Completed	Stable	Stable
UnitTestMapperConnectivityTBB	Passed	Failed	15s 90ms	Completed	Stable	Stable
UnitTestMapperPointsSERIAL	Passed	Failed	38s 380ms	Completed	Stable	Stable
UnitTestTBBDeviceAdapter	Passed	Failed	1m 16s 840ms	Completed	Stable	Stable
UnitTestWaveletCompressorSERIAL	Passed	Failed	2m 7s 70ms	Completed	Stable	Stable

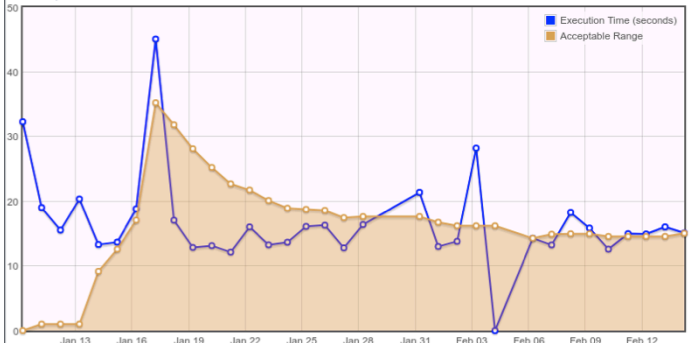
This test took longer to complete (15s 90ms) than the threshold allows (15s 30ms).

Processors 1

Show Command Line

Display graphs: Test Time

View Graph Data as JSON



Test output

Testing

- Testing is used to verify install layout
  - WIP: Building code against the installed vtk-m as part of the testing process

Test: TestInstallSetup (Passed)
Build: e0c0e4b3-build100-[linux-static-debug+64bit_ids+examples+gcc-7+logging+mpi+ninja+openmp+tbb] (renar) on 2019-02-15 04:03:40
Repository revision: e0c0e4b3dae46297a9c1f29fe90f63ddf90c33c3

Test Details: Completed

Test Timing: Passed

Processors 1

Show Command Line
Display graphs: Select...

Test output

```
-- MODE: INSTALL
-- Install configuration: "Debug"
-- Installing: /home/kitware/buildslave/root/vtk-m-renar-linux-static-debug_64bit_ids_examples_gcc-7_logging_mpi_ninja_openmp_tbb/build//CMakeFiles/_tmp_install/lib/cmake/vtkm-1.3/VTKmConfig.cmake
-- Installing: /home/kitware/buildslave/root/vtk-m-renar-linux-static-debug_64bit_ids_examples_gcc-7_logging_mpi_ninja_openmp_tbb/build//CMakeFiles/_tmp_install/lib/cmake/vtkm-1.3/VTKmConfigVersion.cmake
-- Installing: /home/kitware/buildslave/root/vtk-m-renar-linux-static-debug_64bit_ids_examples_gcc-7_logging_mpi_ninja_openmp_tbb/build//CMakeFiles/_tmp_install/share/vtkm-1.3/VTKmREADME.md
-- Installing: /home/kitware/buildslave/root/vtk-m-renar-linux-static-debug_64bit_ids_examples_gcc-7_logging_mpi_ninja_openmp_tbb/build//CMakeFiles/_tmp_install/share/vtkm-1.3/VTKmLICENSE.txt
-- Installing: /home/kitware/buildslave/root/vtk-m-renar-linux-static-debug_64bit_ids_examples_gcc-7_logging_mpi_ninja_openmp_tbb/build//CMakeFiles/_tmp_install/share/vtkm-1.3/cmake/FindTBB.cmake
-- Installing: /home/kitware/buildslave/root/vtk-m-renar-linux-static-debug_64bit_ids_examples_gcc-7_logging_mpi_ninja_openmp_tbb/build//CMakeFiles/_tmp_install/share/vtkm-1.3/cmake/FindOpenGL.cmake
-- Installing: /home/kitware/buildslave/root/vtk-m-renar-linux-static-debug_64bit_ids_examples_gcc-7_logging_mpi_ninja_openmp_tbb/build//CMakeFiles/_tmp_install/share/vtkm-1.3/cmake/FindOpenMP.cmake
-- Installing: /home/kitware/buildslave/root/vtk-m-renar-linux-static-debug_64bit_ids_examples_gcc-7_logging_mpi_ninja_openmp_tbb/build//CMakeFiles/_tmp_install/share/vtkm-1.3/cmake/VTKmCPUVectorization.cmake
-- Installing: /home/kitware/buildslave/root/vtk-m-renar-linux-static-debug_64bit_ids_examples_gcc-7_logging_mpi_ninja_openmp_tbb/build//CMakeFiles/_tmp_install/share/vtkm-1.3/cmake/VTKmDetectCUDAVersion.cu
-- Installing: /home/kitware/buildslave/root/vtk-m-renar-linux-static-debug_64bit_ids_examples_gcc-7_logging_mpi_ninja_openmp_tbb/build//CMakeFiles/_tmp_install/share/vtkm-1.3/cmake/VTKmDeviceAdapters.cmake
-- Installing: /home/kitware/buildslave/root/vtk-m-renar-linux-static-debug_64bit_ids_examples_gcc-7_logging_mpi_ninja_openmp_tbb/build//CMakeFiles/_tmp_install/share/vtkm-1.3/cmake/VTKmExportHeaderTemplate.h.
-- Installing: /home/kitware/buildslave/root/vtk-m-renar-linux-static-debug_64bit_ids_examples_gcc-7_logging_mpi_ninja_openmp_tbb/build//CMakeFiles/_tmp_install/share/vtkm-1.3/cmake/VTKmRenderingContexts.cmake
-- Installing: /home/kitware/buildslave/root/vtk-m-renar-linux-static-debug_64bit_ids_examples_gcc-7_logging_mpi_ninja_openmp_tbb/build//CMakeFiles/_tmp_install/share/vtkm-1.3/cmake/VTKmWrappers.cmake
-- Installing: /home/kitware/buildslave/root/vtk-m-renar-linux-static-debug_64bit_ids_examples_gcc-7_logging_mpi_ninja_openmp_tbb/build//CMakeFiles/_tmp_install/lib/cmake/vtkm-1.3/VTKmTargets.cmake
-- Installing: /home/kitware/buildslave/root/vtk-m-renar-linux-static-debug_64bit_ids_examples_gcc-7_logging_mpi_ninja_openmp_tbb/build//CMakeFiles/_tmp_install/lib/cmake/vtkm-1.3/VTKmTargets-debug.cmake
-- Installing: /home/kitware/buildslave/root/vtk-m-renar-linux-static-debug_64bit_ids_examples_gcc-7_logging_mpi_ninja_openmp_tbb/build//CMakeFiles/_tmp_install/include/vtkm-1.3/vtkm/Version.h
-- Installing: /home/kitware/buildslave/root/vtk-m-renar-linux-static-debug_64bit_ids_examples_gcc-7_logging_mpi_ninja_openmp_tbb/build//CMakeFiles/_tmp_install/include/vtkm-1.3/vtkm/Assert.h
```

Testing

- Testing will be used to monitor compile times leverages Ninja ability to report per TU compilation times

Test: BuildTime (Failed)
Build: d48eaf5c-build68-[linux-static-debug+64bit_ids+examples+gcc-7+logging+mpi+ninja+openmp+tbb]-buildtime_test (renar) on 2019-02-11 15:01:28
Repository revision: d48eaf5ca9f530fc600515b7e1ddf562fe5326e9

Test Details: Completed (Failed)

Test Timing: Passed

Exit Value 1
Processors 1

Show Command Line
Display graphs: Select...

Test output

```
-- ------------------------+------+-----------------------------
-- Time                    | Fail | Command
-- ------------------------+------+-----------------------------
-- 000d 00h 02m 02s 671ms | **** | vtkm/cont/testing/CMakeFiles/UnitTests_vtkm_cont_testing_mpi.dir/UnitTestSerializationArrayHandle.cxx.o
-- 000d 00h 01m 29s 059ms | **** | vtkm/worklet/testing/CMakeFiles/UnitTests_vtkm_worklet_testing.dir/UnitTestCellSetConnectivity.cxx.o
-- 000d 00h 01m 26s 522ms | **** | vtkm/filter/testing/CMakeFiles/UnitTests_vtkm_filter_testing.dir/UnitTestMarchingCubesFilter.cxx.o
-- 000d 00h 01m 22s 904ms | **** | vtkm/filter/testing/CMakeFiles/UnitTests_vtkm_filter_testing.dir/UnitTestSplitSharpEdgesFilter.cxx.o
-- 000d 00h 01m 21s 207ms | **** | vtkm/filter/testing/CMakeFiles/UnitTests_vtkm_filter_testing.dir/UnitTestGradient.cxx.o
-- 000d 00h 01m 19s 332ms | **** | vtkm/filter/testing/CMakeFiles/UnitTests_vtkm_filter_testing.dir/UnitTestClipWithFieldFilter.cxx.o
-- 000d 00h 01m 10s 381ms | **** | vtkm/filter/testing/CMakeFiles/UnitTests_vtkm_filter_testing.dir/UnitTestProbe.cxx.o
-- 000d 00h 01m 07s 516ms | **** | vtkm/worklet/testing/CMakeFiles/UnitTests_vtkm_worklet_testing.dir/UnitTestWorkletMapField.cxx.o
-- 000d 00h 01m 05s 897ms | **** | vtkm/filter/testing/CMakeFiles/UnitTests_vtkm_filter_testing.dir/UnitTestContourTreeUniformAugmentedFilter.cxx.o
-- 000d 00h 01m 04s 863ms | **** | vtkm/cont/tbb/testing/CMakeFiles/UnitTests_vtkm_cont_tbb_testing.dir/UnitTestTBBArrayHandleFancy.cxx.o
-- 000d 00h 01m 04s 771ms | **** | vtkm/cont/openmp/testing/CMakeFiles/UnitTests_vtkm_cont_openmp_testing.dir/UnitTestOpenMPArrayHandleFancy.cxx.o
-- 000d 00h 01m 03s 281ms | **** | vtkm/cont/serial/testing/CMakeFiles/UnitTests_vtkm_cont_serial_testing.dir/UnitTestSerialArrayHandleFancy.cxx.o
```

Device Level Benchmarks

VTK-m has a collection of device adapter level benchmarks used for micro performance comparisons.

- Allows developers to test new implementations for parallel primitives
- Allows VTK-m to get a baseline for new hardware
- Allows device adapters to be compared against each other

AtomicArray | CopySpeeds | DeviceAdapter

Kitware

Device Level Benchmarks

Comparison:

Speedup	Warn	OpenMP 42cores SMT2	CUDA 1gpu -transfer	Benchmark (Type)
8.286		0.057378 +- 0.010280	0.006925 +- 0.003243	LowerBounds (1024MiB, 5% config, vtkm::Float64)
9.743		0.063469 +- 0.009263	0.006514 +- 0.003137	LowerBounds (1024MiB, 5% config, vtkm::Int64)
39.412		0.405742 +- 0.054356	0.010295 +- 0.003476	LowerBounds (1024MiB, 25% config, vtkm::Float64)
30.929		0.366755 +- 0.040363	0.011858 +- 0.004388	LowerBounds (1024MiB, 25% config, vtkm::Int64)
54.974		1.085270 +- 0.119102	0.019742 +- 0.005618	LowerBounds (1024MiB, 75% config, vtkm::Float64)
55.944		1.167100 +- 0.083073	0.020862 +- 0.007075	LowerBounds (1024MiB, 75% config, vtkm::Int64)
8.972		0.070113 +- 0.045381	0.007815 +- 0.003804	Reduce (1024MiB, vtkm::Float64)
8.038		0.059702 +- 0.034040	0.007428 +- 0.003453	Reduce (1024MiB, vtkm::Int64)
9.820		0.076123 +- 0.040566	0.007752 +- 0.003664	Reduce (1024MiB, vtkm::Pair< vtkm::Int64, vtkm::Float32 >)
5.478		0.112917 +- 0.021229	0.020612 +- 0.004912	ReduceByKey (1024MiB, 25% distinct keys) [vtkm::Id, vtkm::Vec< vtkm::Float64, 3 >]
6.602		0.171744 +- 0.060392	0.026012 +- 0.008613	ReduceByKey (1024MiB, 25% distinct keys) [vtkm::Id, vtkm::Vec< vtkm::Float32, 3 >]
2.385	!!	0.045134 +- 0.004877	0.018927 +- 0.009338	ScanExclusive (1024MiB, vtkm::Float64)
1.910	!!!	0.041481 +- 0.003169	0.021718 +- 0.006635	ScanExclusive (1024MiB, vtkm::Int64)
2.329	!!	0.044808 +- 0.002846	0.019236 +- 0.009786	ScanInclusive (1024MiB, vtkm::Float64)
1.992	!!!	0.044884 +- 0.002025	0.022528 +- 0.010406	ScanInclusive (1024MiB, vtkm::Int64)
7.400		0.599648 +- 0.030927	0.081039 +- 0.006644	Sort (1024MiB, vtkm::Float64)
9.198		0.677927 +- 0.060614	0.073704 +- 0.005635	Sort (1024MiB, vtkm::Int64)

IBM Power System AC922 node (SUMMIT) GPU: Volata V100 CPU: 2x Power9 [42 cores, SMT2]

Algorithm Level Benchmarks

VTK-m has a collection of filter and worklet level benchmarks. These are generally used to verify whole algorithm or application performance.

- Allows developers to test new implementations for algorithms
- Allows VTK-m to get a baseline for new hardware

FieldWorklets | ToplogyWorklets | Filters

Kitware

VTK-m Filter Benchmarks

Comparison on 512*512*512 Structured Grid:

Speedup	Warn	OpenMP 42cores SMT2	CUDA 1gpu +uvm 8x8x4	Benchmark (Type)
0.878	!!!!	0.018225 +- 0.000898	0.020755 +- 0.000217	AvgPointToCell (float)
10.751		0.423592 +- 0.018624	0.039400 +- 0.000179	AvgCellToPoint (float)
4.237		0.396348 +- 0.012257	0.093539 +- 0.000076	ExternalFaces (float)
2.772	!!	0.529174 +- 0.012926	0.190922 +- 0.006715	ExternalFaces merge points (float)
3.504	!	0.130657 +- 0.002057	0.037284 +- 0.000093	Gradient Gradient ColumnOrder (float)
30.778		1.128130 +- 0.014497	0.036654 +- 0.000042	Gradient Divergence ColumnOrder (vec<float,3>)
7.395		1.181490 +- 0.025548	0.159759 +- 0.002150	Gradient Gradient ColumnOrder (vec<float,3>)
8.802		1.085760 +- 0.030149	0.123357 +- 0.003849	Gradient Gradient RowOrder (vec<float,3>)
5.278		0.105243 +- 0.001999	0.019941 +- 0.000033	Gradient PointGradient ColumnOrder (vec<float,3>)
0.998	!!!!	0.237009 +- 0.007141	0.237514 +- 0.003443	Gradient PointGradient Gradient+ALL ColumnOrder (vec<float,3>)
2.489	!!	0.207679 +- 0.008602	0.083439 +- 0.000907	MarchingCubes contours=1 mergePoints=0 normals=1 fastNormals=0 (float)
1.635	!!!	0.141355 +- 0.012551	0.086445 +- 0.006428	MarchingCubes contours=1 mergePoints=0 normals=1 fastNormals=1 (float)
1.497	!!!	0.174703 +- 0.009029	0.116684 +- 0.007648	MarchingCubes contours=1 mergePoints=1 normals=0 fastNormals=0 (float)
4.499		0.528579 +- 0.033417	0.117500 +- 0.001521	MarchingCubes contours=3 mergePoints=0 normals=1 fastNormals=0 (float)
2.397	!!	0.306236 +- 0.004642	0.127771 +- 0.009548	MarchingCubes contours=3 mergePoints=0 normals=1 fastNormals=1 (float)
4.591		1.045640 +- 0.059820	0.227758 +- 0.006864	MarchingCubes contours=3 mergePoints=1 normals=0 fastNormals=0 (float)
4.399		0.119523 +- 0.005674	0.027169 +- 0.000504	Threshold (float)
1.793	!!!	0.072578 +- 0.002941	0.040469 +- 0.000071	WarpScalar (float)
1.628	!!!	0.061275 +- 0.001162	0.037628 +- 0.000060	WarpVector (float)

Kitware

IBM Power System AC922 node (SUMMIT) GPU: Volata V100 CPU: 2x Power9 [42 cores, SMT2]

Filter Benchmarks

Comparison on 512*512*512 Structured Grid:

Speedup	Warn	OpenMP 42cores SMT2	CUDA 1gpu -uvm 8x8x4	Benchmark (Type)
1.541	!!!	0.018225 +- 0.000898	0.011827 +- 0.000210	AvgPointToCell (float)
10.556		0.423592 +- 0.018624	0.040127 +- 0.000559	AvgCellToPoint (float)
6.757		0.396348 +- 0.012257	0.058658 +- 0.004308	ExternalFaces (float)
5.924		0.529174 +- 0.012926	0.089327 +- 0.010585	ExternalFaces merge points (float)
8.662		0.130657 +- 0.002057	0.015084 +- 0.000595	Gradient Gradient ColumnOrder (float)
49.045		1.128130 +- 0.014497	0.023002 +- 0.000018	Gradient Divergence ColumnOrder (vec<float,3>)
13.267		1.181490 +- 0.025548	0.089058 +- 0.000695	Gradient Gradient ColumnOrder (vec<float,3>)
19.780		1.085760 +- 0.030149	0.054891 +- 0.000072	Gradient Gradient RowOrder (vec<float,3>)
8.194		0.105243 +- 0.001999	0.012845 +- 0.000038	Gradient PointGradient ColumnOrder (vec<float,3>)
1.780	!!!	0.237009 +- 0.007141	0.133153 +- 0.000451	Gradient PointGradient Gradient+ALL ColumnOrder (vec<float,3>)
8.728		0.207679 +- 0.008602	0.023796 +- 0.000241	MarchingCubes contours=1 mergePoints=0 normals=1 fastNorms=0 float)
5.459		0.141355 +- 0.012551	0.025894 +- 0.000185	MarchingCubes contours=1 mergePoints=0 normals=1 fastNorms=1 float)
4.035		0.174703 +- 0.009029	0.043297 +- 0.016281	MarchingCubes contours=1 mergePoints=1 normals=0 fastNorms=0 float)
11.626		0.528579 +- 0.033417	0.045466 +- 0.000172	MarchingCubes contours=3 mergePoints=0 normals=1 fastNorms=0 float)
7.105		0.306236 +- 0.004642	0.043102 +- 0.000190	MarchingCubes contours=3 mergePoints=0 normals=1 fastNorms=1 float)
7.629		1.045640 +- 0.059820	0.137057 +- 0.019126	MarchingCubes contours=3 mergePoints=1 normals=0 fastNorms=0 float)
8.819		0.119523 +- 0.005674	0.013553 +- 0.000185	Threshold (float)
6.867		0.072578 +- 0.002941	0.010570 +- 0.000057	WarpScalar (float)
6.068		0.061275 +- 0.001162	0.010097 +- 0.000050	WarpVector (float)

IBM Power System AC922 node (SUMMIT) GPU: Volata V100 CPU: 2x Power9 [42 cores, SMT2]

Filter Benchmarks [Old 3D Scheduling]

Comparison on 512*512*512 Structured Grid:

Speedup	Warn	OpenMP 42cores SMT2	CUDA 1gpu -uvm 64x2x1	Benchmark (Type)
0.181	!!!!	0.018225 +- 0.000898	0.100918 +- 0.000452	AvgPointToCell (float)
1.670	!!!	0.423592 +- 0.018624	0.253668 +- 0.000493	AvgCellToPoint (float)
0.689	!!!!	0.396348 +- 0.012257	0.575359 +- 0.000800	ExternalFaces (float)
0.877	!!!!	0.529174 +- 0.012926	0.603062 +- 0.001143	ExternalFaces merge points (float)
1.028	!!!	0.130657 +- 0.002057	0.127148 +- 0.001399	Gradient Gradient ColumnOrder (float)
3.825		1.128130 +- 0.014497	0.294906 +- 0.000184	Gradient Divergence ColumnOrder (vec<float,3>)
3.041	!	1.181490 +- 0.025548	0.388490 +- 0.002319	Gradient Gradient ColumnOrder (vec<float,3>)
3.066	!	1.085760 +- 0.030149	0.354109 +- 0.001686	Gradient Gradient RowOrder (vec<float,3>)
0.651	!!!!	0.105243 +- 0.001999	0.161743 +- 0.000041	Gradient PointGradient ColumnOrder (vec<float,3>)
0.767	!!!!	0.237009 +- 0.007141	0.309168 +- 0.000855	Gradient PointGradient Gradient+ALL ColumnOrder (vec<float,3>)
2.016	!!	0.207679 +- 0.008602	0.103004 +- 0.000463	MarchingCubes contours=1 mergePoints=0 normals=1 fastNormals=0 (float)
1.350	!!!	0.141355 +- 0.012551	0.104722 +- 0.000337	MarchingCubes contours=1 mergePoints=0 normals=1 fastNormals=1 (float)
1.490	!!!	0.174703 +- 0.009029	0.117260 +- 0.000287	MarchingCubes contours=1 mergePoints=1 normals=0 fastNormals=0 (float)
3.538	!	0.528579 +- 0.033417	0.149390 +- 0.000685	MarchingCubes contours=3 mergePoints=0 normals=1 fastNormals=0 (float)
2.095	!!	0.306236 +- 0.004642	0.146197 +- 0.000220	MarchingCubes contours=3 mergePoints=0 normals=1 fastNormals=1 (float)
4.506		1.045640 +- 0.059820	0.232070 +- 0.001858	MarchingCubes contours=3 mergePoints=1 normals=0 fastNormals=0 (float)
1.473	!!!	0.119523 +- 0.005674	0.081163 +- 0.000134	Threshold (float)
6.833		0.072578 +- 0.002941	0.010622 +- 0.000058	WarpScalar (float)
6.048		0.061275 +- 0.001162	0.010132 +- 0.000053	WarpVector (float)

IBM Power System AC922 node (SUMMIT) GPU: Volata V100 CPU: 2x Power9 [42 cores, SMT2]

Thank You!

Robert Maynard

robert.maynard@kitware.com

@robertjmaynard

This research was supported by the Exascale Computing Project (http://www.exascaleproject.org), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

Project Number: 17-SC-20-SC

Checkout out VTK-m @ gitlab.kitware.com/vtk/vtk-m and Kitware @ www.kitware.com

Please complete the Presenter Evaluation sent to you by email or through the GTC Mobile App.

Your feedback is important!

Kitware