SAND2014-15790PE

VTK-M: A FOUNDATION

Kenneth Moreland, et al.

SDAV Vis Teleconference
Tuesday 15, 2014

VTK-m Project

Target Multi and Many core architectures.

Combines the strengths of multiple projects:

EAVL, DAX, and PISTON

Reduce the complexity of writing highly concurrent code.

Funding

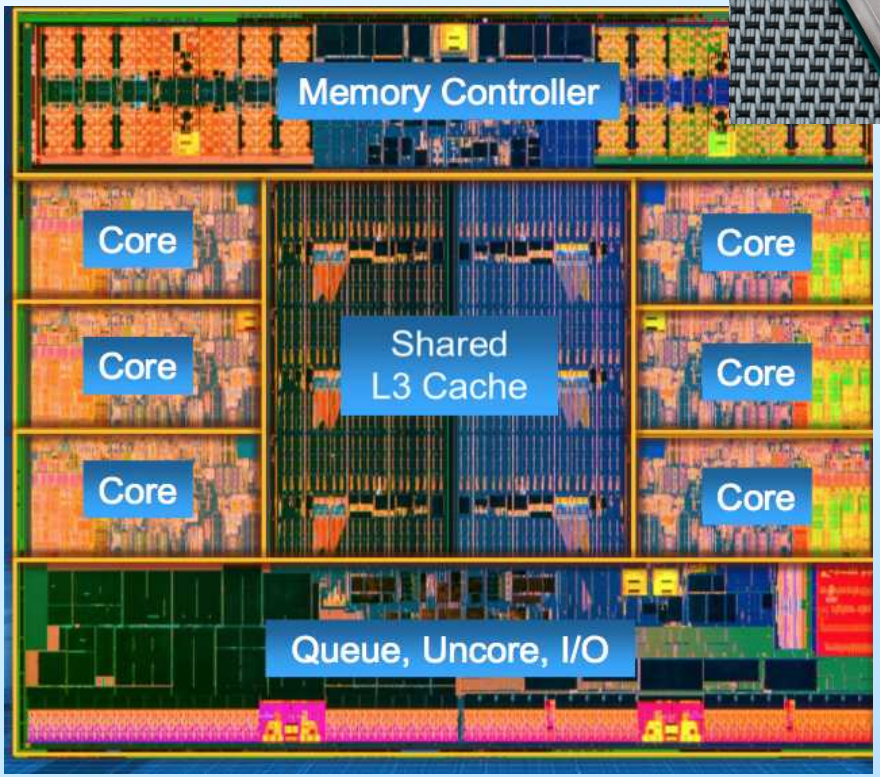
- VTK-m development made part of recent XVis proposal to ASCR Scientific Data Management, Analysis and Visualization at Extreme Scale 2 call
- Recommended for award
  - Funding expected to start in September

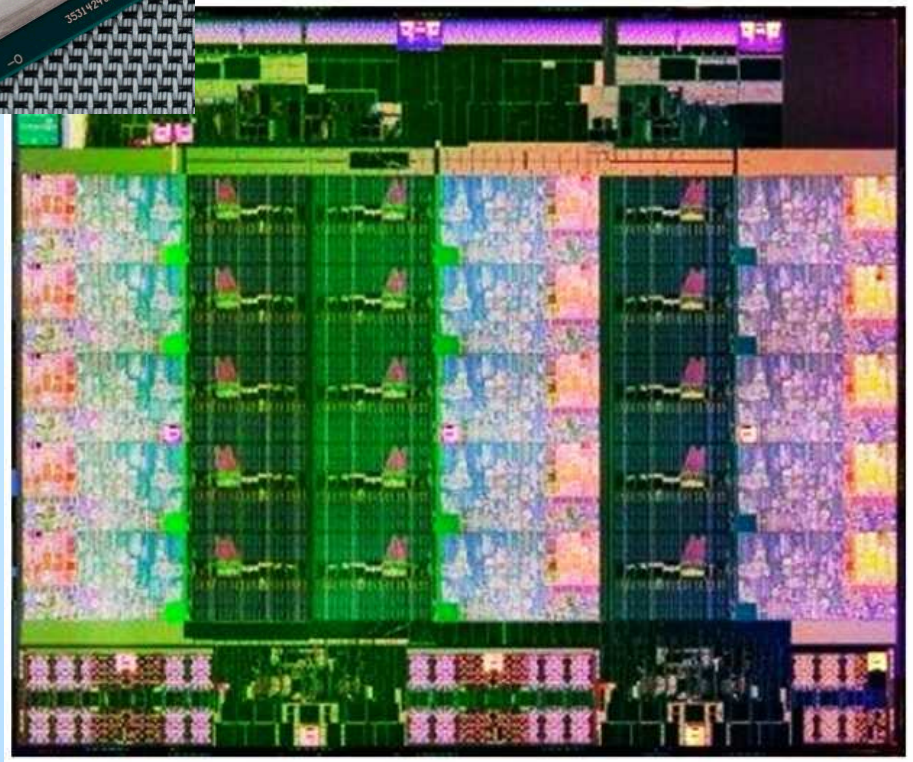
Multi and Many Core

What exactly are we talking about

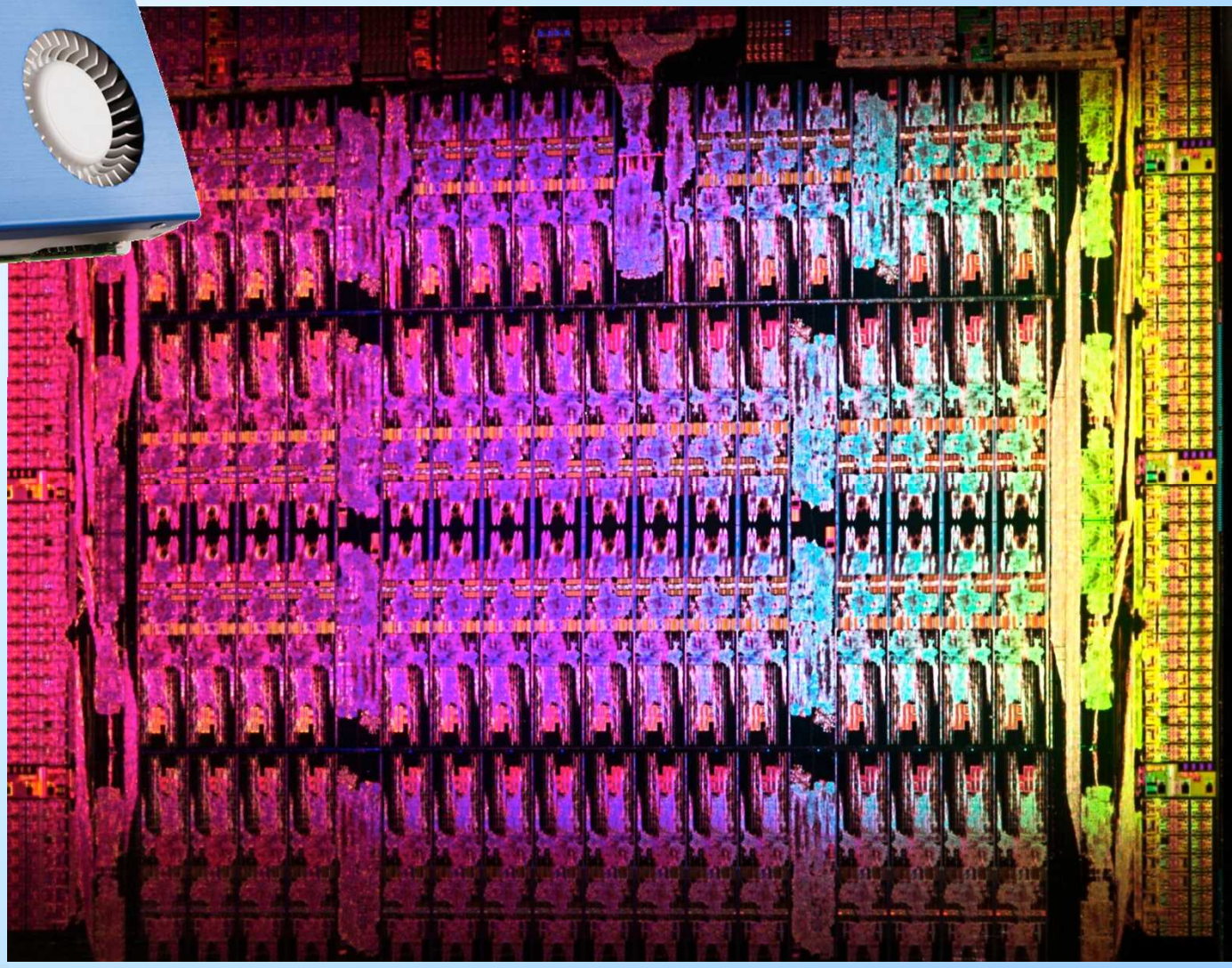
Multi Core


INTEL CONFIDENTIAL
QF7Z ES 3.60GHZ
COSTA RICA
3320A806


Memory Controller
Core
Core
Core
Core
Core
Core
Shared
L3 Cache
Queue, Uncore, I/O



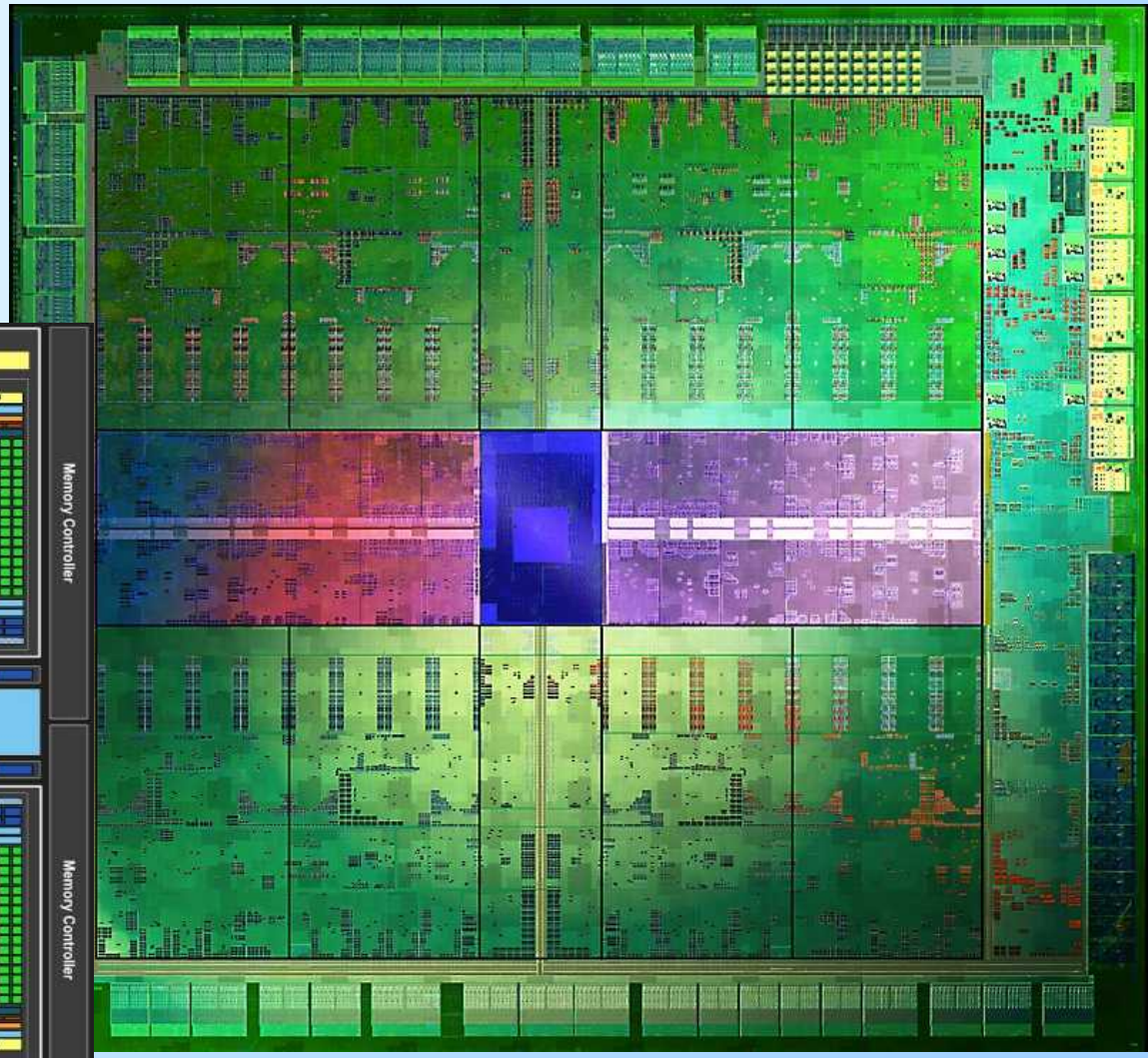
Multi Core



Many Core

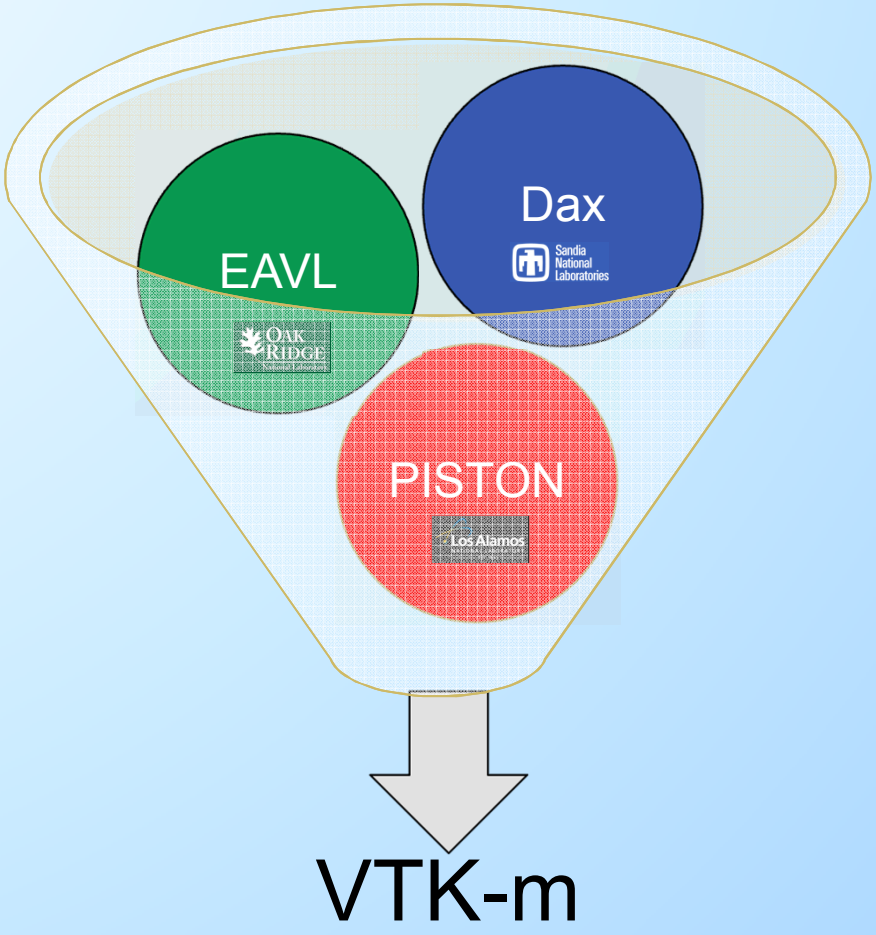

NVIDIA TESLA


GPC
Raster Engine
SMX
Polymorph Engine 2.0
Memory Controller
L2 Cache

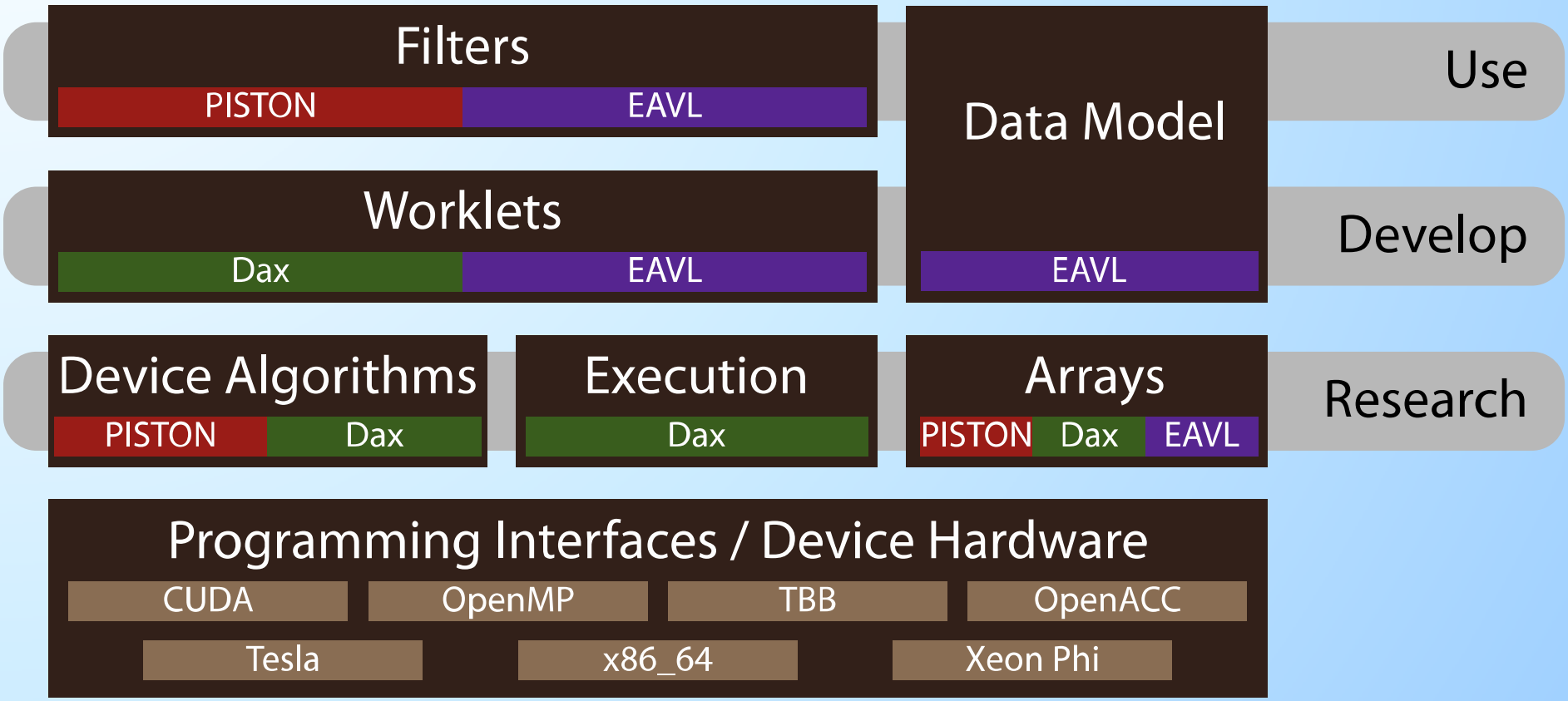


VTK-m: A Foundation

Combines the strengths of multiple projects



VTK-m Architecture

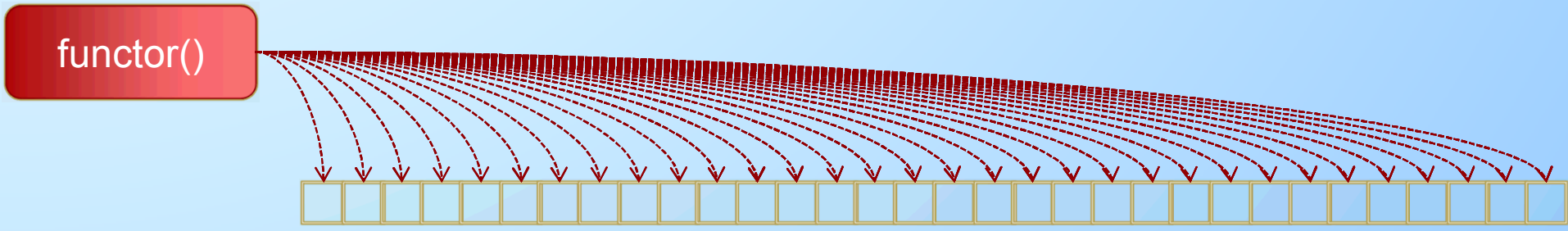


Highly concurrent code

High Level Concept

Decompose algorithms into sections that can run small section of data.

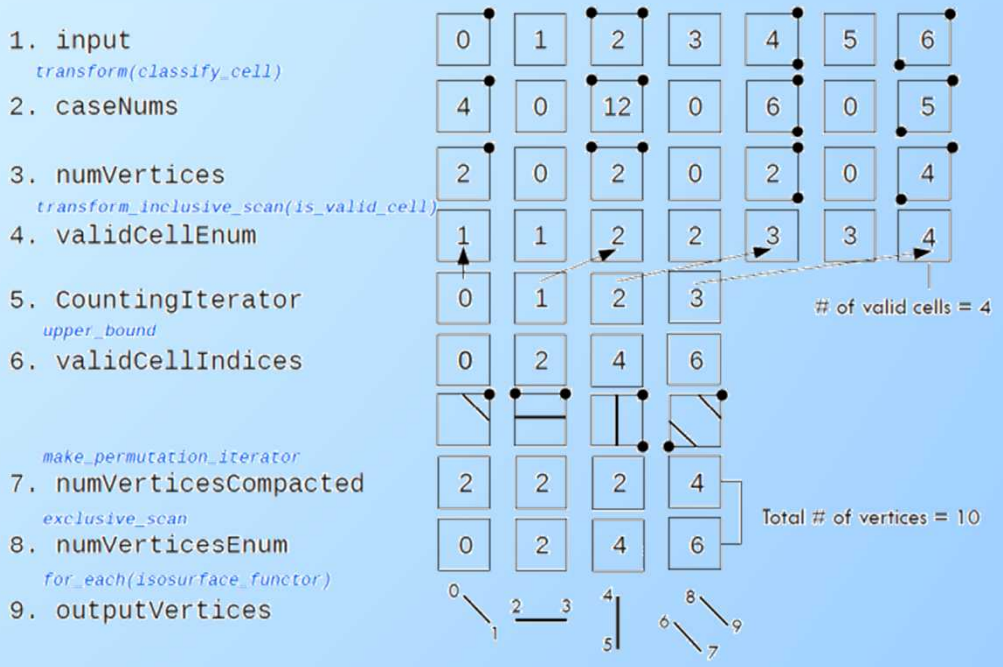
The approach is essentially that same as presented by Baker and colleagues, functional mapping [Baker, et al. 2010]



Encapsulate Parallel Operations

Visualization algorithms can be composed of base operations like scan, sort, for each, reduce, etc.

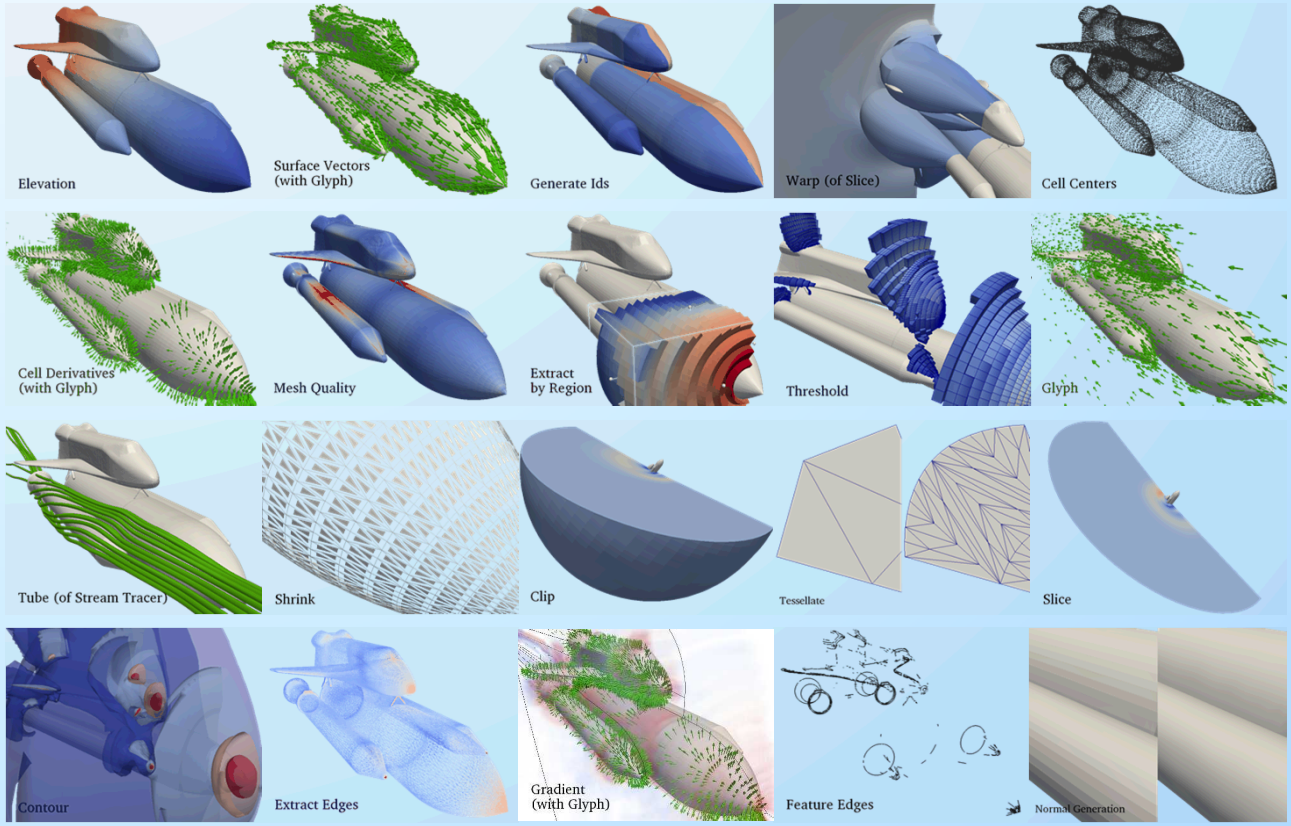
Based on work of Belloch [1990]



Lo, Swell, Ahrens. “PISTON: A Portable Cross-Platform Framework for Data-Parallel Visualization Operations,” 2012

Simplifying Algorithm Development

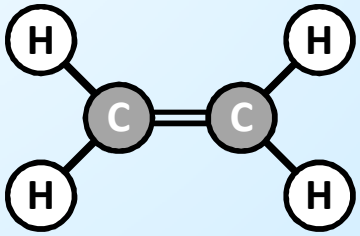
Many visualization algorithms share algorithm structure. These patterns can be directly implemented in the toolkit.

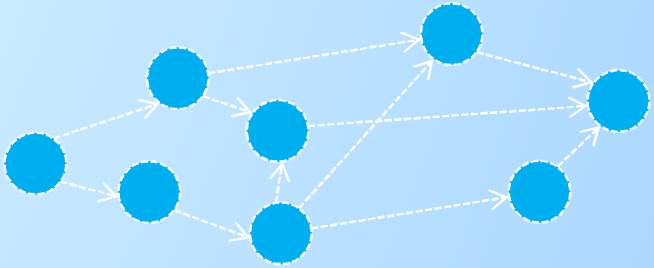


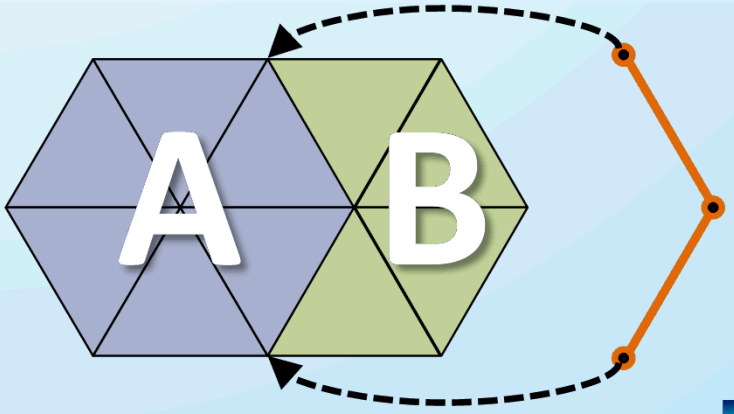
Moreland, Geveci, Ma, Maynard. "A Classification of Scientific Visualization Algorithms for Massive Threading," 2013

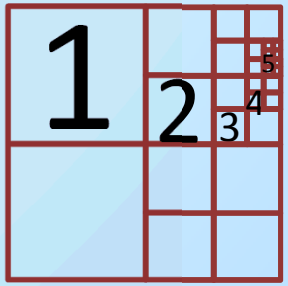
Powerful, Flexible Data Structures

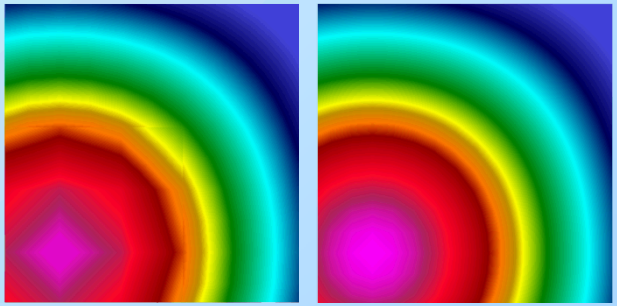
Need to represent many structure types but with consistent access that works on all devices

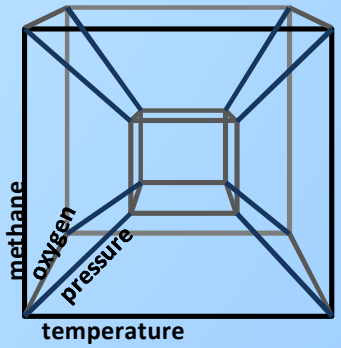










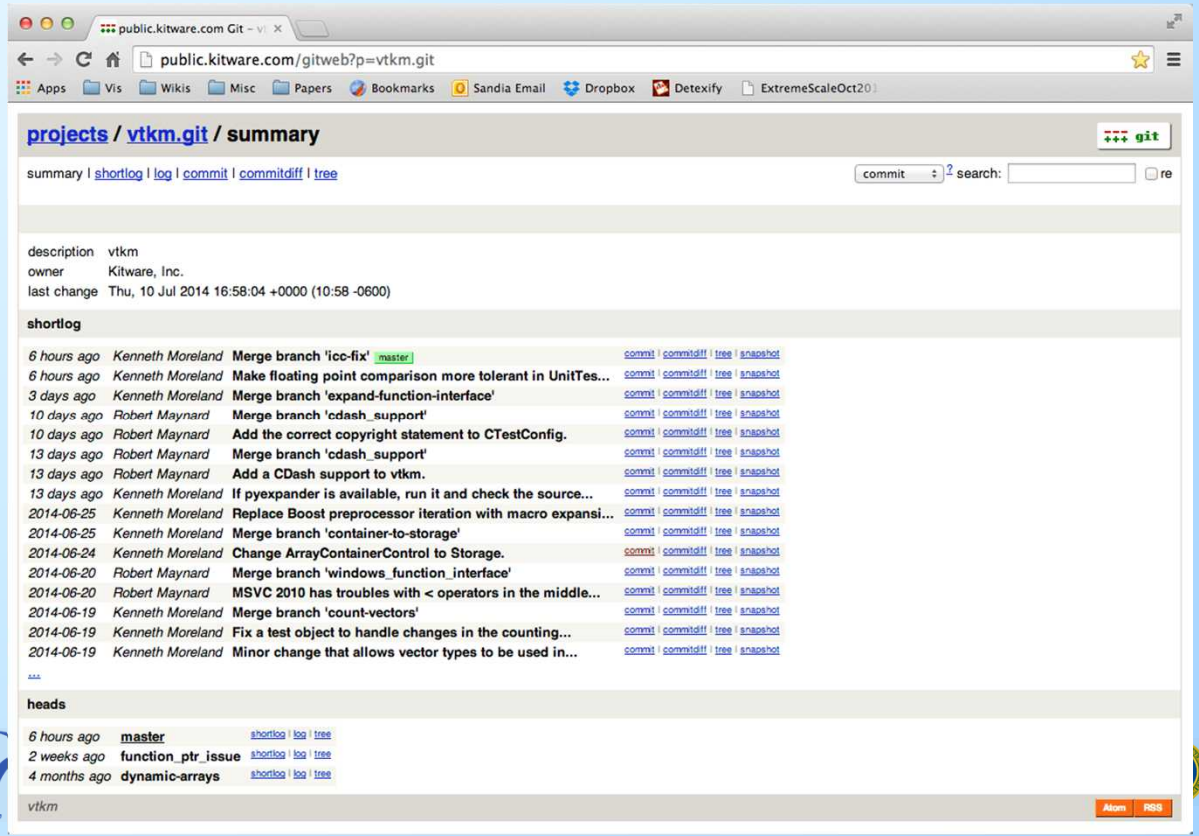


VTK-m Code Repository

http://public.kitware.com/gitweb?p=vtkm.git

Currently contains basic foundations to combine features from Dax, PISTON, EAVL

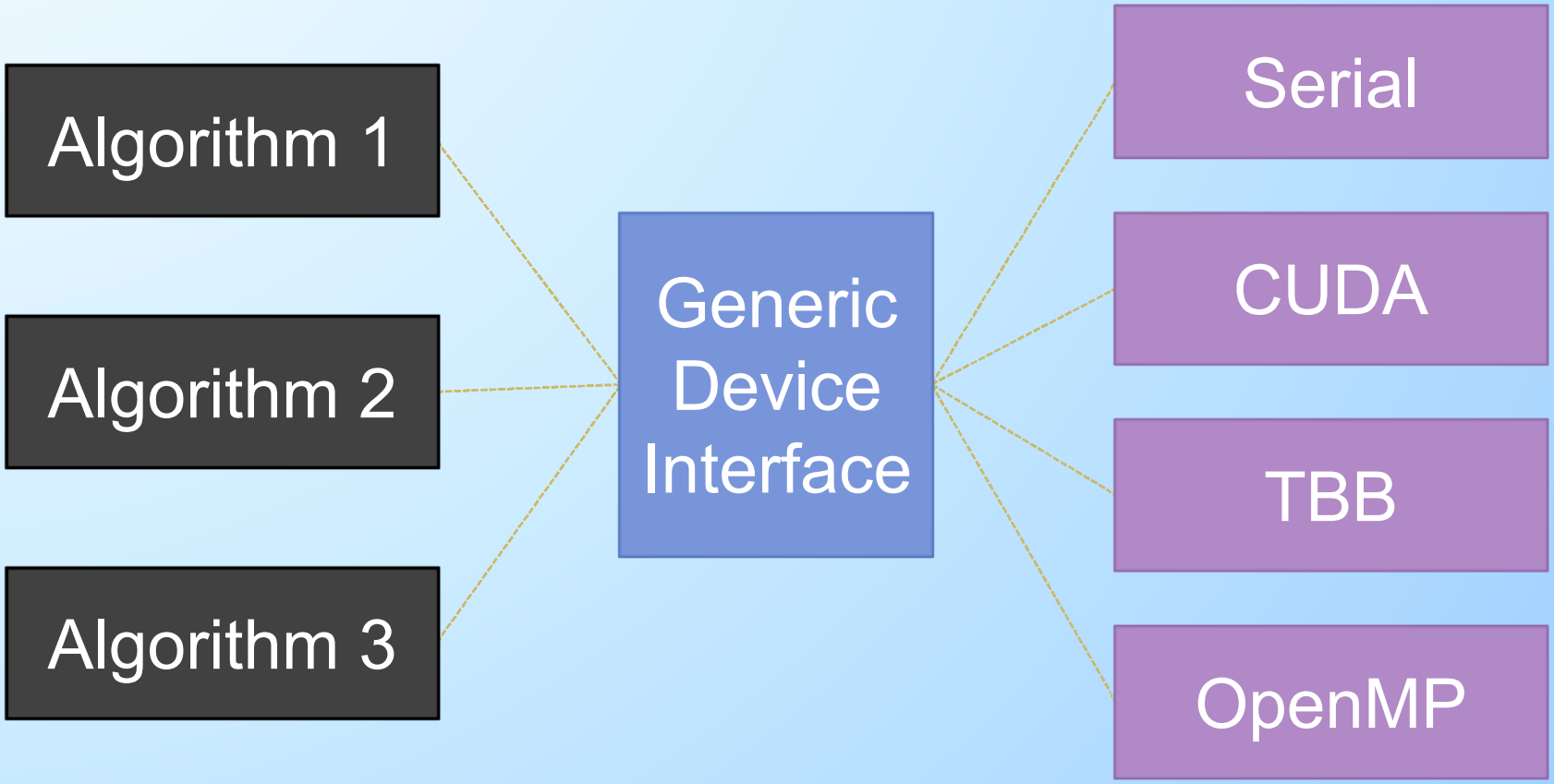
Expect to hit ground running when XVis starts



Interchangeable Devices

Cannot rewrite every algorithm for every device

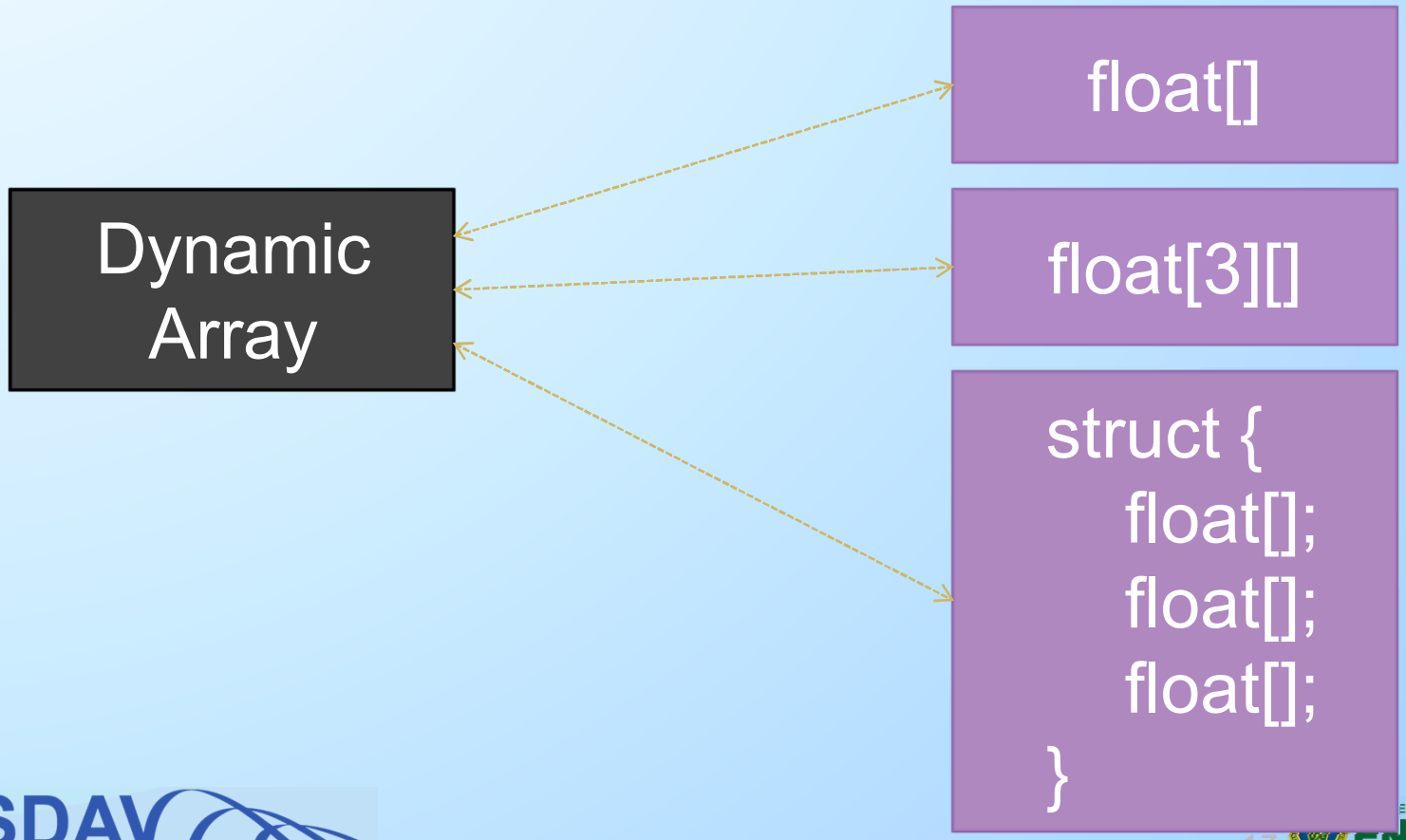
Use an abstract device to minimize porting



Enable Polymorphic Types

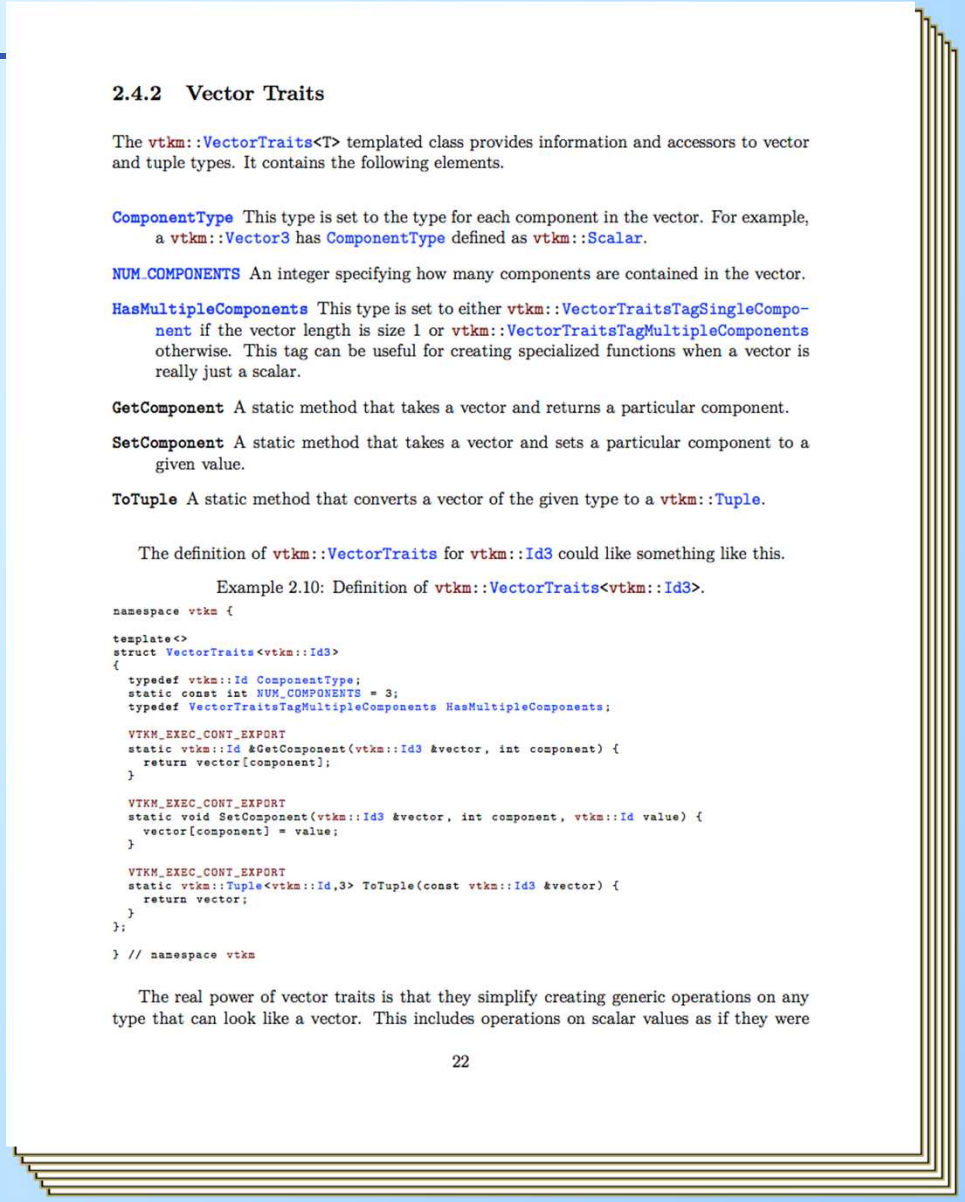
Efficient code likes templates for direct data access (especially on GPUs)

Data types are often unknown until run time



Documentation

- Writing documentation for VTK-m as we go
  - Important for communication
- All existing code well documented with examples (currently about 60 pages)



2.4.2 Vector Traits

The vtkm::VectorTraits<T> templated class provides information and accessors to vector and tuple types. It contains the following elements.

ComponentType This type is set to the type for each component in the vector. For example, a vtkm::Vector3 has ComponentType defined as vtkm::Scalar.

NUM_COMPONENTS An integer specifying how many components are contained in the vector.

HasMultipleComponents This type is set to either vtkm::VectorTraitsTagSingleComponent if the vector length is size 1 or vtkm::VectorTraitsTagMultipleComponents otherwise. This tag can be useful for creating specialized functions when a vector is really just a scalar.

GetComponent A static method that takes a vector and returns a particular component.

SetComponent A static method that takes a vector and sets a particular component to a given value.

ToTuple A static method that converts a vector of the given type to a vtkm::Tuple.

The definition of vtkm::VectorTraits for vtkm::Id3 could like something like this.

Example 2.10: Definition of vtkm::VectorTraits<vtkm::Id3>.

```
namespace vtkm {

template<>
struct VectorTraits<vtkm::Id3>
{
  typedef vtkm::Id ComponentType;
  static const int NUM_COMPONENTS = 3;
  typedef VectorTraitsTagMultipleComponents HasMultipleComponents;

  VTKM_EXEC_CONT_EXPORT
  static vtkm::Id &GetComponent(vtkm::Id3 &vector, int component) {
    return vector[component];
  }

  VTKM_EXEC_CONT_EXPORT
  static void SetComponent(vtkm::Id3 &vector, int component, vtkm::Id value) {
    vector[component] = value;
  }

  VTKM_EXEC_CONT_EXPORT
  static vtkm::Tuple<vtkm::Id,3> ToTuple(const vtkm::Id3 &vector) {
    return vector;
  }
};

} // namespace vtkm
```

The real power of vector traits is that they simplify creating generic operations on any type that can look like a vector. This includes operations on scalar values as if they were

22