

# Development of VTK-m During ECP

HPC Best Practices Webinar

Kenneth Moreland  
Oak Ridge National Laboratory

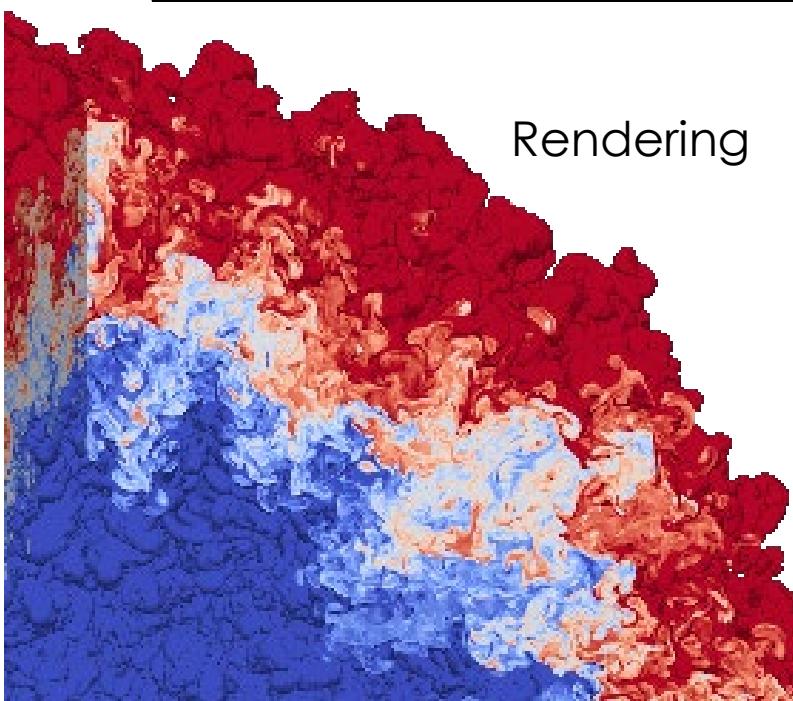
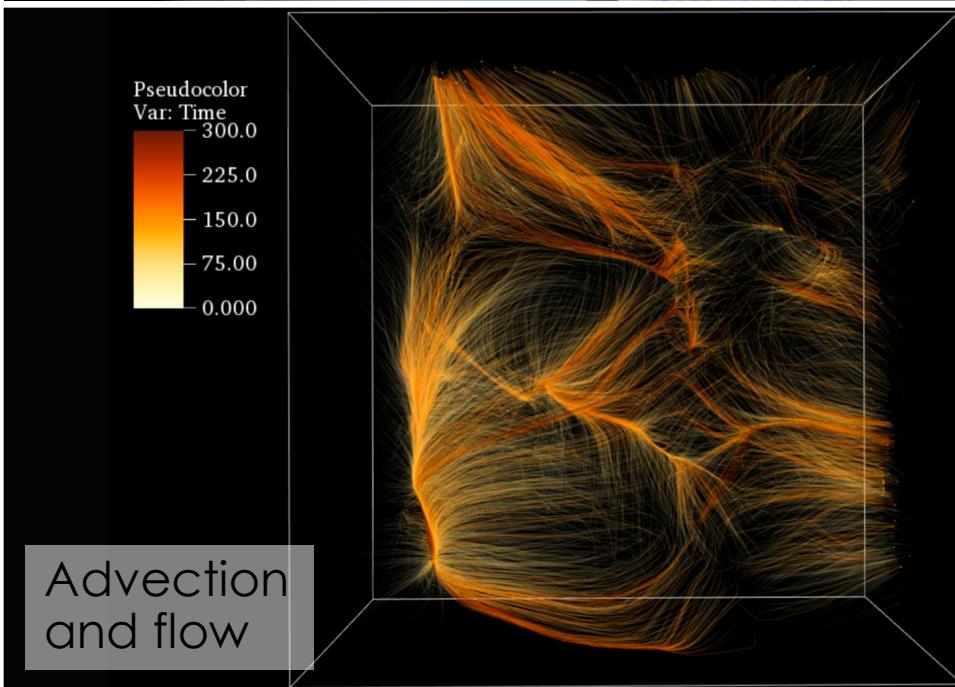
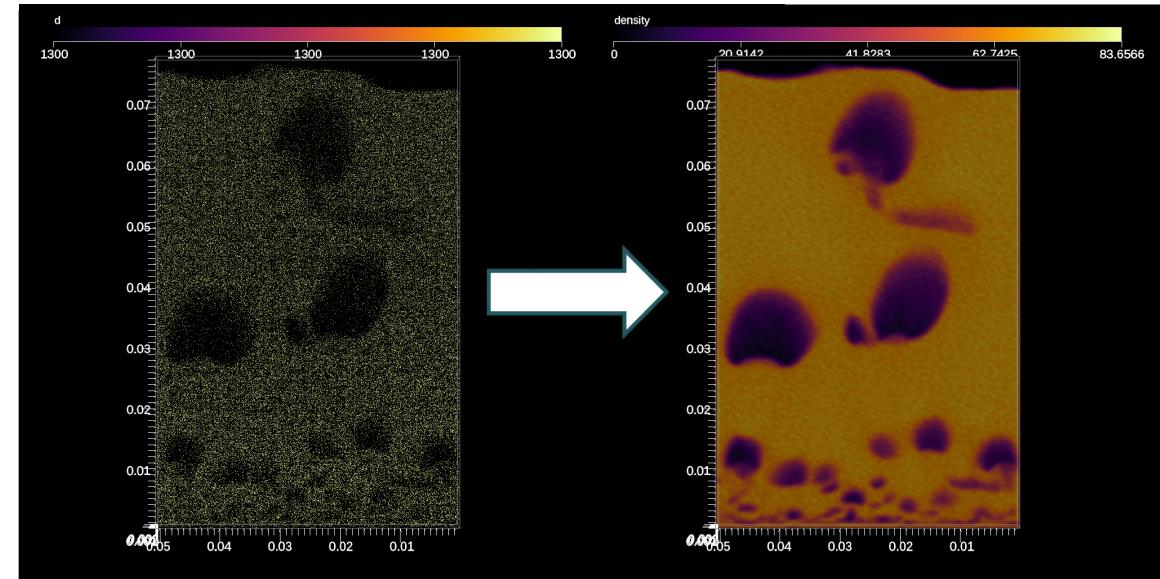
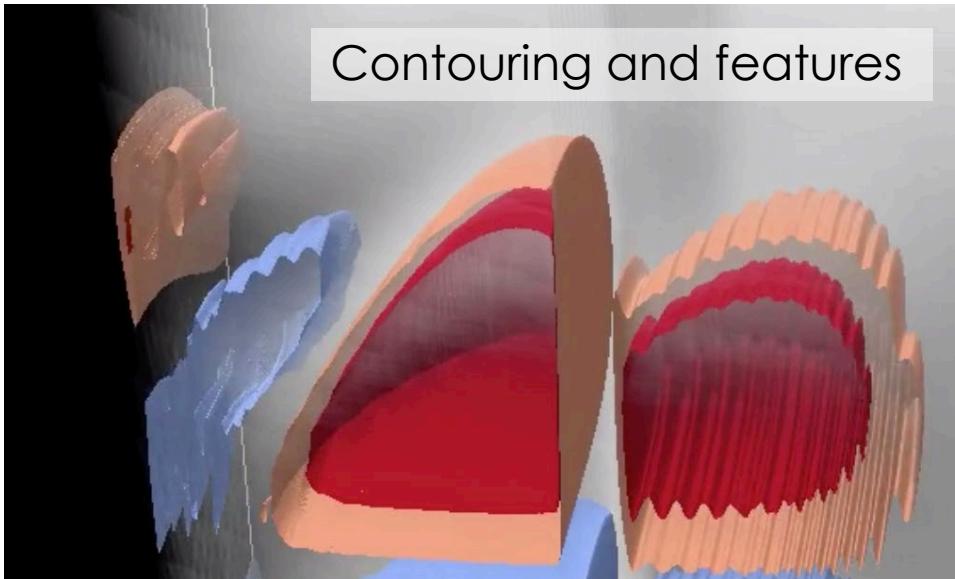
September 25, 2024

ORNL is managed by UT-Battelle LLC for the US Department of Energy



# VTK-m: Visualization on Accelerators

Particle Density



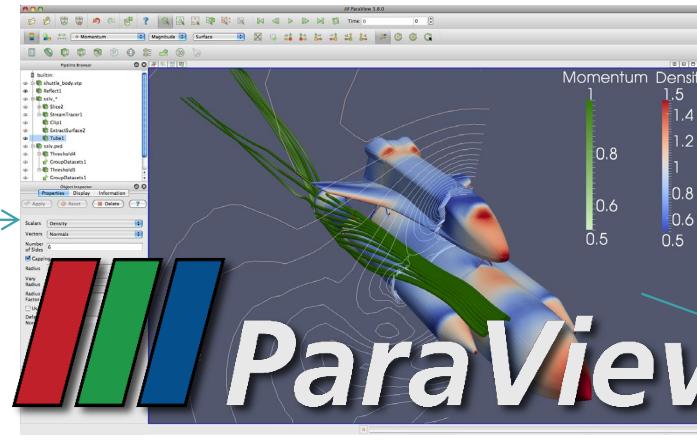
And  
much  
more...

Simulations

In Situ Vis Library  
(Integration with Sim)



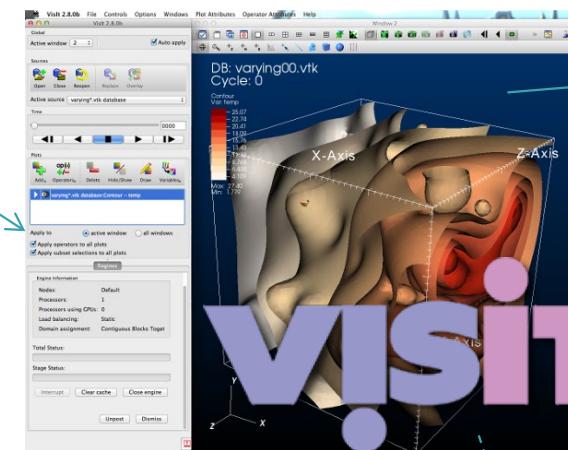
GUI / Parallel Management



Base Vis Library  
(Algorithm Implementation)



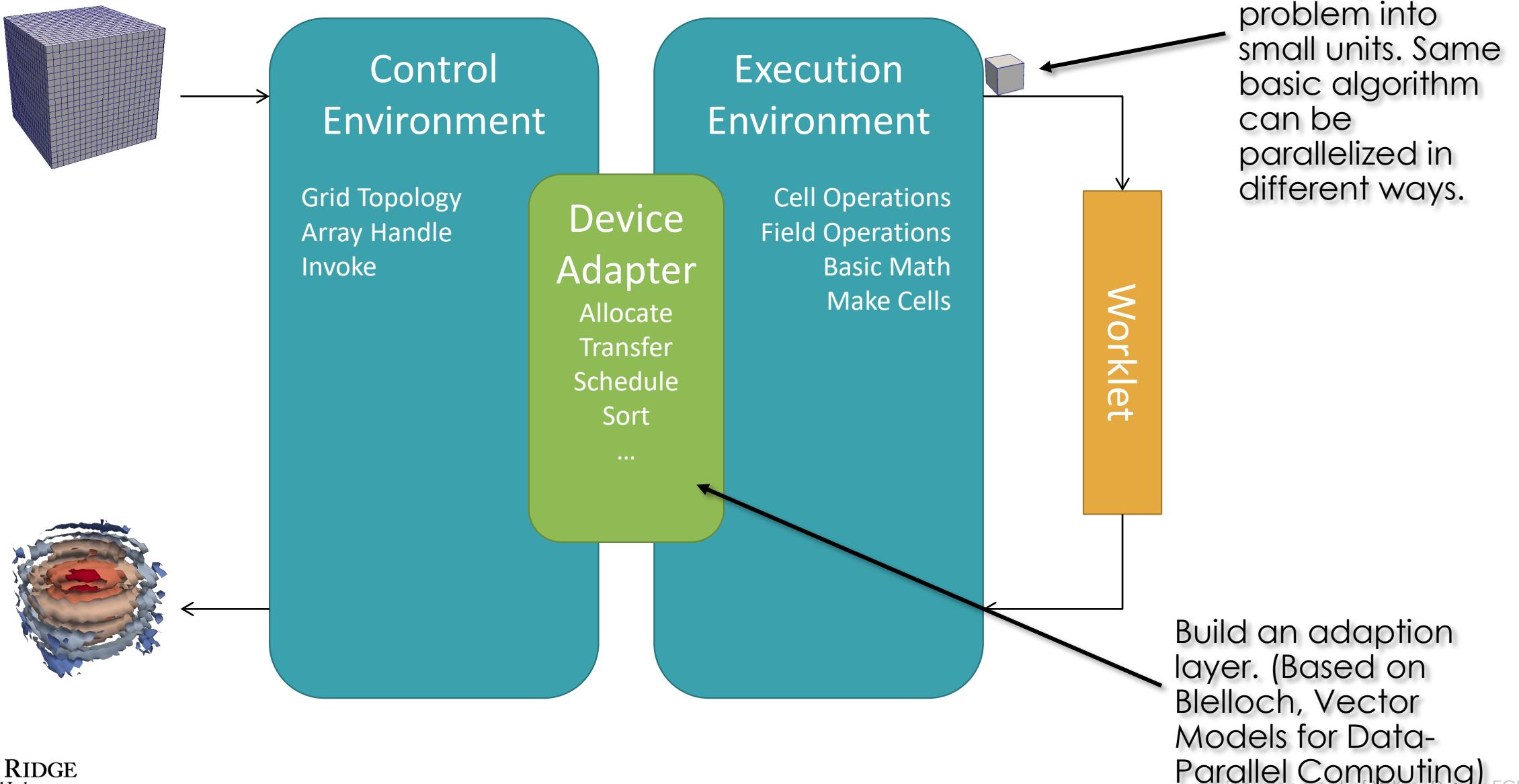
Libsim



Multithreaded Algorithms  
Processor Portability



# VTK-m Designed with Portability in Mind

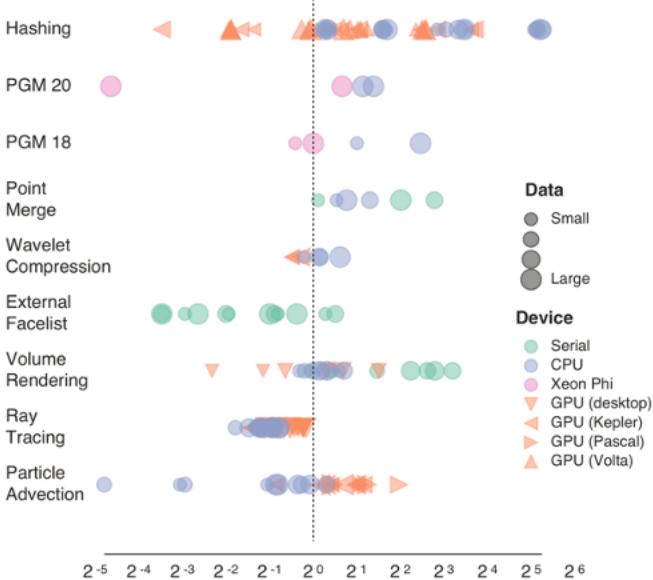


# What went well

- Prepare for portability from the start
  - Recognize there is a lot of variability between hardware
  - Focus on one major push
- There are software libraries to help with this process
  - Kokkos, RAJA, ...
- Recognize an overhead for portability

# VTK-m Demonstrates Performance Portability

- Science Problem
  - Leadership class facilities are using a variety of processor technologies.
    - Hardware for the Exascale Computing Project (ECP) takes advantage of processors from a variety of vendors — Intel, NVIDIA, AMD — each with their own programming language and API.
  - There are many scientific visualization algorithms that need to work well on these accelerator processors.
    - It is infeasible to update every algorithm for every device.
- Technical Solution
  - The VTK-m software framework provides abstractions that make it possible to implement an algorithm once and port it to any of the devices used by ECP.
  - A recent literature review of algorithms implemented in VTK-m and implemented for a particular hardware device show that the portable VTK-m implementation works as well as the specialized implementations.
- Science Impact
  - VTK-m makes it possible to support scientific visualization on the heterogeneous architectures of today's leadership class facilities.



Algorithm	CPUs	GPUs	X. Phi	Serial	Total
External facelist	-	-	-	0.34	0.34
PGM 18	3.32	-	0.87	-	1.69
PGM 20	2.39	-	0.25	-	0.78
Particle advection	0.38	1.53	-	-	0.76
Point merge	1.82	-	-	3.10	2.38
Ray tracing	0.47	0.55	-	-	0.51
Volume rendering	1.13	0.83	-	3.10	1.43
Wavelet compression	1.13	0.75	-	-	0.92
Hashing	5.97	1.45	-	-	2.94
Total	1.45	0.95	0.47	1.48	<b>1.14</b>

This plot and table demonstrate the speedup from implementing a visualization algorithm in VTK-m vs. directly for a particular type of hardware. The geometric means indicate a speedup near 1, meaning that the VTK-m implementations, perform about the same as code optimized for a specific platform. From Moreland, et al. (2021). Minimizing Development Costs for Efficient Many-Core Visualization Using MCD<sup>3</sup>. *Parallel Computing*, 108(102834). doi:10.1016/j.parco.2021.102834



UNIVERSITY OF OREGON



OAK RIDGE  
National Laboratory



Kitware



Lawrence Livermore National Laboratory

# Software Development Management



# The Beginnings

- VTK-m started as an Office of Science research project
  - Organized across 3 labs, 1 university, and an industry partner
- Starting plan that VTK-m would become production software
- Had the advantage that VTK-m was originally derived from 3 existing software projects
  - Quickly established a base framework to structure contributions
- Established some rough contributing guidelines early
  - It is fun to revisit slides from the kickoff describing what was expected

# VTK-m Kickoff Slides (November 2014)

## VTK-m Development Method



- Currently a small-ish but widely distributed software team
  - 6 – 10 part-time developers across 4 – 6 institutions
- Need enough formality to avoid wild west shootouts
  - Not so much to stifle early research and development
- Focus on communication, cooperation, and testing
  - Weekly Development Meetings
  - Design Documents
  - Branchy Workflow
  - Coding Conventions
  - Unit Tests



## Weekly Development Meetings



- Once a week 1 hour meeting for developers
  - Videocon or telecon (technology permitting)
- Opportunity to communicate current development progress
  - Roundtable of ongoing/about to be merged/merged source
  - Forum for design discussions
  - Heartbeat of project
- Easy and informal (e.g. not like SDAV Vis telecons)
  - Don't prepare slides
  - It's OK to say "Nothing to report" (happens all the time)
  - It's OK to miss meetings due to conflicts with travel, vacation, etc.
  - It's OK to have fun
- Will establish good time later

# VTK-m Kickoff Slides (November 2014)

## Coding Conventions

- Coding conventions help developers agree on a style that makes the overall code easier to read and easier to interface.
- VTK-m has an explicit set of coding conventions.
  - Documented in the User's Guide.
  - Based off of explicit and implicit conventions for VTK.
    - Exception braces are not hanging.
- Although formalized, the coding conventions are not dogma.
  - The conventions can be broken if there is a good reason.
  - Conventions can be considered authority on developer disagreements.



### Chapter 9

#### Coding Conventions

Several developers contribute to VTK-m and we welcome others who are interested to also contribute to the project. To ensure readability and consistency in the code, we have adopted the following coding conventions. Many of these conventions are adapted from the coding conventions of the VTK project. This is because many of the developers are familiar with VTK coding and because we expect VTK-m to have continual interaction with VTK.

- All code contributed to VTK-m must be compatible with VTK-m's BSD license.
- Copyright notices should appear at the top of all source, configuration, and text files. The statement should have the following form:

```
=====
// Copyright (c) Kitware, Inc.
// All rights reserved.
// See LICENSE.txt for details.
// This software is distributed WITHOUT ANY WARRANTY; without even
// the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
// PURPOSE. See the above copyright notice for more information.
//
// Copyright 2014 Sandia Corporation.
// Copyright 2014 UT-Battelle, LLC.
// Copyright 2014 Los Alamos National Security
//
// Under the terms of Contract DE-AC04-94AL85000 with Sandia Corporation,
// the U.S. Government retains certain rights in this software.
//
// Under the terms of Contract DE-AC52-06NA25396 with Los Alamos National
// Laboratory (LANL), the U.S. Government retains certain rights in
// this software.
=====
```

The CopyrightStatement test checks all files for a similar statement. The test will print out a suggested text that can be copied and pasted to any file that has a missing copyright statement (with appropriate replacement of comment prefix). Exceptions to this copyright statement (for example, third-party files with different but compatible statements) can be added to LICENSE.txt.

# VTK-m Kickoff Slides (November 2014)

## Unit Tests



- All code should have an associated regression test
  - Direct support for unit tests in build system
  - Tests automatically run in nightly dashboard once in master
- There is almost a one-to-one correlation between source code file and test
  - Yes, that's a lot of tests.
  - Already 48 tests now.
- Look for in a code review.
- Saves time in the long run.

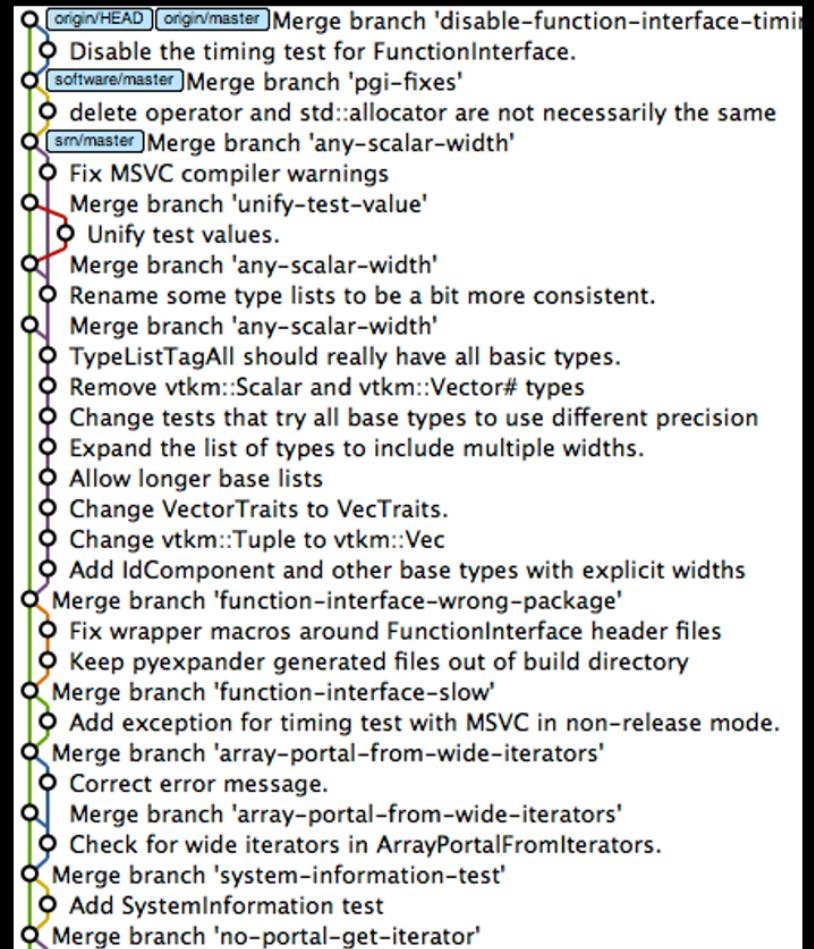
```
kmorel:0> ctest
Test project /Users/kmorel/src/builds/vtkm
  Start 1: CopyrightStatement
  1/48 Test #1: CopyrightStatement ..... Passed    2.21 sec
  Start 2: SystemInformation
  2/48 Test #2: SystemInformation ..... Passed    0.04 sec
  Start 3: UnitTestExtent
  3/48 Test #3: UnitTestExtent ..... Passed    0.77 sec
  Start 4: UnitTestListTag
  4/48 Test #4: UnitTestListTag ..... Passed    0.00 sec
  Start 5: UnitTestPair
  5/48 Test #5: UnitTestPair ..... Passed    0.01 sec
  Start 6: UnitTestTesting
  6/48 Test #6: UnitTestTesting ..... Passed    0.00 sec
  Start 7: UnitTestTypeListTag
  7/48 Test #7: UnitTestTypeListTag ..... Passed    0.00 sec
  Start 8: UnitTestTypes
  8/48 Test #8: UnitTestTypes ..... Passed    0.01 sec
  Start 9: UnitTestTypeTraits
  9/48 Test #9: UnitTestTypeTraits ..... Passed    0.00 sec
  Start 10: UnitTestVecTraits
  10/48 Test #10: UnitTestVecTraits ..... Passed    0.00 sec
  Start 11: UnitTestConfigureFor32
  11/48 Test #11: UnitTestConfigureFor32 ..... Passed    0.00 sec
  Start 12: UnitTestConfigureFor64
  12/48 Test #12: UnitTestConfigureFor64 ..... Passed    0.00 sec
  Start 13: UnitTestFunctionInterface
  13/48 Test #13: UnitTestFunctionInterface ..... Passed    0.00 sec
  Start 14: UnitTestArrayManagerExecutionShareWithControl
  14/48 Test #14: UnitTestArrayManagerExecutionShareWithControl ...
  Start 15: UnitTestArrayPortalFromIterators
  15/48 Test #15: UnitTestArrayPortalFromIterators ..... Passed    0.01 sec
  Start 16: UnitTestDynamicTransform
  16/48 Test #16: UnitTestDynamicTransform ..... Passed    0.00 sec
```

# VTK-m Kickoff Slides (November 2014)

## Branchy Workflow



- A simple git workflow to organize development by topics.
  - Git history loops that branch off master, develop a topic, and merge back in.
- Lots of advantages (I use it even when not collaborating).
  - Promotes breaking work into logical sections (as opposed to large monolithic changes).
  - Allows you to simultaneously work on independent features.
  - Minimizes developers stepping on each other's foot.
    - Reduces number of times master branch changes.
    - Code not visible until ready.
  - Gives opportunity to clean history.
  - Easier to share not-ready work with yourself and others.



# ECP Project Management

# Branchy Workflow Solidified in GitLab Merge Requests

- Only way to contribute to VTK-m is through a GitLab Merge Request
- Reviews expected on all MRs
- clang-format enforces conventions
- All tests run for every merge request (CI)
- Contributors don't need "write access" to contribute
  - Anyone with a GitLab account can fork the repo and submit an MR
  - A developer can merge other's MRs
  - Safe way to onboard new developers

The screenshot shows a GitLab Merge Request page for a pull request titled "Distributed Isosurface extraction from branch decomposition". The page includes the following details:

- Overview:** 58 commits, 7 pipelines, 48 changes.
- Description:** Given the distributed hierarchical contour tree and its branch decomposition, we compute the volume of each branch and select the branches with high volumes (i.e., Top K). Then, we extract the meshes on the contours nearby the saddles of these top-volume branches.
- Reviewers:** Gunther Weber.
- Labels:** None.
- Milestone:** None.
- Time tracking:** No estimate or time spent.
- Participants:** 5 participants shown with their profile icons.

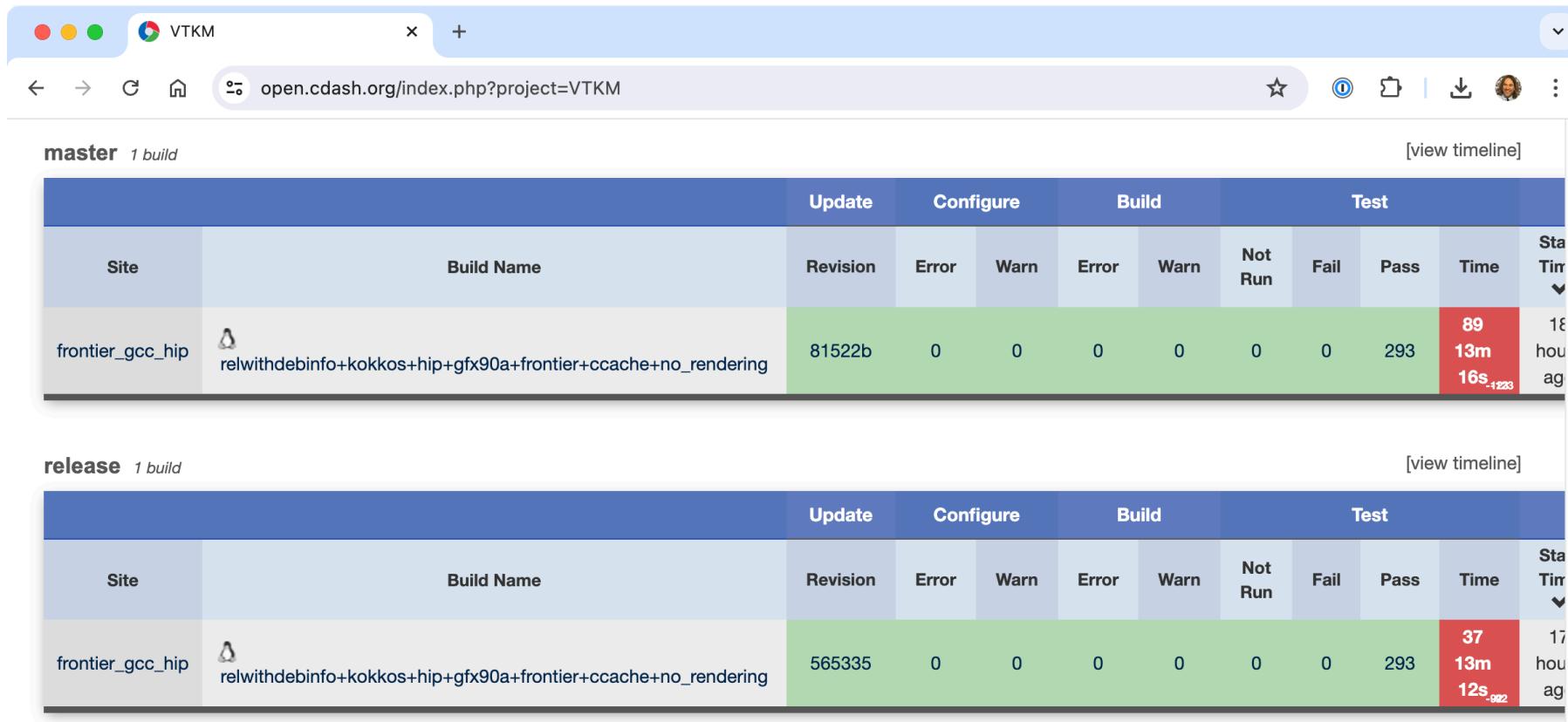
The merge request has passed the pipeline and is ready to be merged. The "Merge" button is visible at the bottom of the main content area.

# Later Requirement: Safe Deprecation

- Early revisions regularly broke the API
  - Necessary for the experimental development we were doing
  - As we moved to production and adoption, this became unacceptable
- Needed a way to change the API w/o breaking it in minor revisions
- Solution: deprecation mechanism
  - Use [ [deprecated] ] modifier (in C macros) to mark old functionality
    - Use of deprecated features gives compiler warning that points to replacement
  - Any new functionality needs a new name or overload

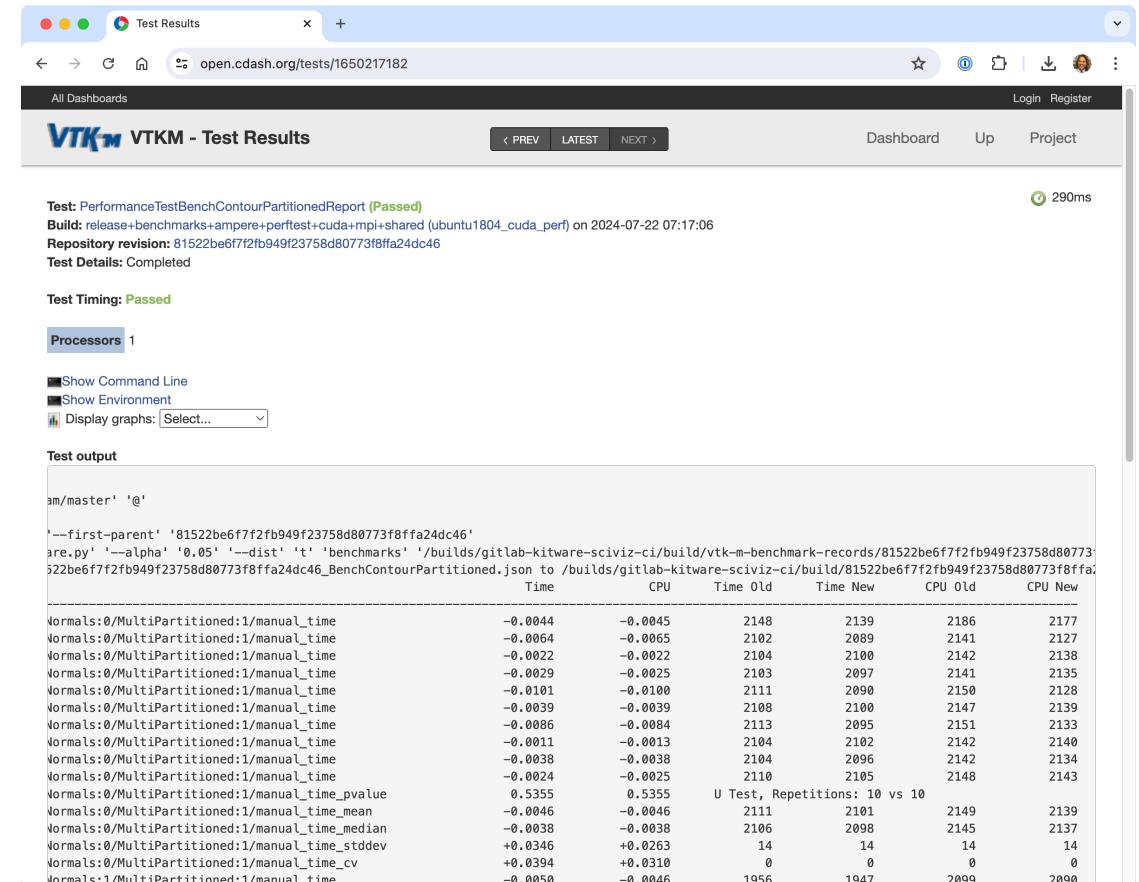
# CI Improvement: Build and Run on OLCF

- ECP worked to provide access to Spock, Crusher, and Frontier
- There are still barriers; the implementation needs babysitting



# CI Improvement: Performance Testing

- Most tests just verify the result is correct, but does not ensure that performance does not drop
- We recently added a special job that runs a select set of benchmarks
- Performs a null-hypothesis test against previous run execution time with a threshold of 95% failure
- Not perfect
  - A wide tolerance to avoid false positives
  - The number of things tested is limited
  - Times of code in containers on shared devices can be janky



# Helpful Features



# Early Feature: Cross Platform Support

- Builds across systems supported by CMake
- Although GPUs were a focus, intentionally focused on others
  - Worked on CPUs, even in serial
- **Can develop on laptop, deploy on HPC**
- Careful structure to protect most code from parallel race conditions
  - Majority of debugging can happen on deterministic serial devices

# Comprehensive Documentation



# Comprehensive Documentation



VTK-m comes with many filters, which are documented in Chapter 9. For this example, we will demonstrate the use of the `vtkm::filter::MeshQuality` filter, which is defined in the `vtkm/filter/MeshQuality.h` header file. The `MeshQuality` filter will compute for each cell in the input data a quantity representing some metric of the cell's shape. Several metrics are available, and in this example we will find the area of each cell.

Like all filters, `MeshQuality` contains an `Execute` method that takes an input `DataSet` and produces an output `DataSet`. It also has several methods used to set up the parameters of the execution. Section 9.1.9 provides details on all the options of `MeshQuality`. Suffice it to say that in this example we instruct the filter to find the area of each cell, which it will output to a field named "area."

Example 3.3: Running a filter.

```
1 | vtkm::filter::mesh_info::MeshQuality cellArea(  
2 |   vtkm::filter::mesh_info::CellMetric::Area);  
3 | vtkm::cont::DataSet outData = cellArea.Execute(inData);
```

## 3.4 Rendering an Image

Although it is possible to leverage external rendering systems, VTK-m comes with its own self-contained image rendering algorithms. These rendering classes are completely implemented with the parallel features provided by VTK-m, so using rendering in VTK-m does not require any complex library dependencies.

Even a simple rendering scene requires setting up several parameters to establish what is to be featured in the image including what data should be rendered, how that data should be represented, where objects should be placed in space, and the qualities of the image to generate. Consequently, setting up rendering in VTK-m involves many steps. Chapter 10 goes into much detail on the ways in which a rendering scene is specified. For now, we just briefly present some boilerplate to achieve a simple rendering.

Example 3.4: Rendering data.

```
1 | vtkm::rendering::Actor actor(  
2 |   outData.GetCellSet(), outData.GetCoordinateSystem(), outData.GetField("area"));  
3 |
```

# Comprehensive Documentation



The VTK-m User's Guide — [vtk-m.readthedocs.io/en/latest/index.html](https://vtk-m.readthedocs.io/en/latest/index.html)

The VTK-m User's Guide

VTK-M USER'S GUIDE

- 1. Getting Started
- 2. Using VTK-m
- 3. Developing Algorithms
- 4. Advanced Development
- 5. Core Development
- 6. Appendix

Upstream: A celebration of open source. Watch now!

Join the VTK-m Community at <http://m.vtk.org>.

**VTK-m User's Guide**

- 1. Getting Started
  - 1.1. Introduction
  - 1.2. Building and Installing VTK-m
  - 1.3. Quick Start
- 2. Using VTK-m
  - 2.1. Base Types
  - 2.2. VTK-m Version
  - 2.3. Initialization
  - 2.4. Data Sets
  - 2.5. File I/O
  - 2.6. Running Filters
  - 2.7. Provided Filters

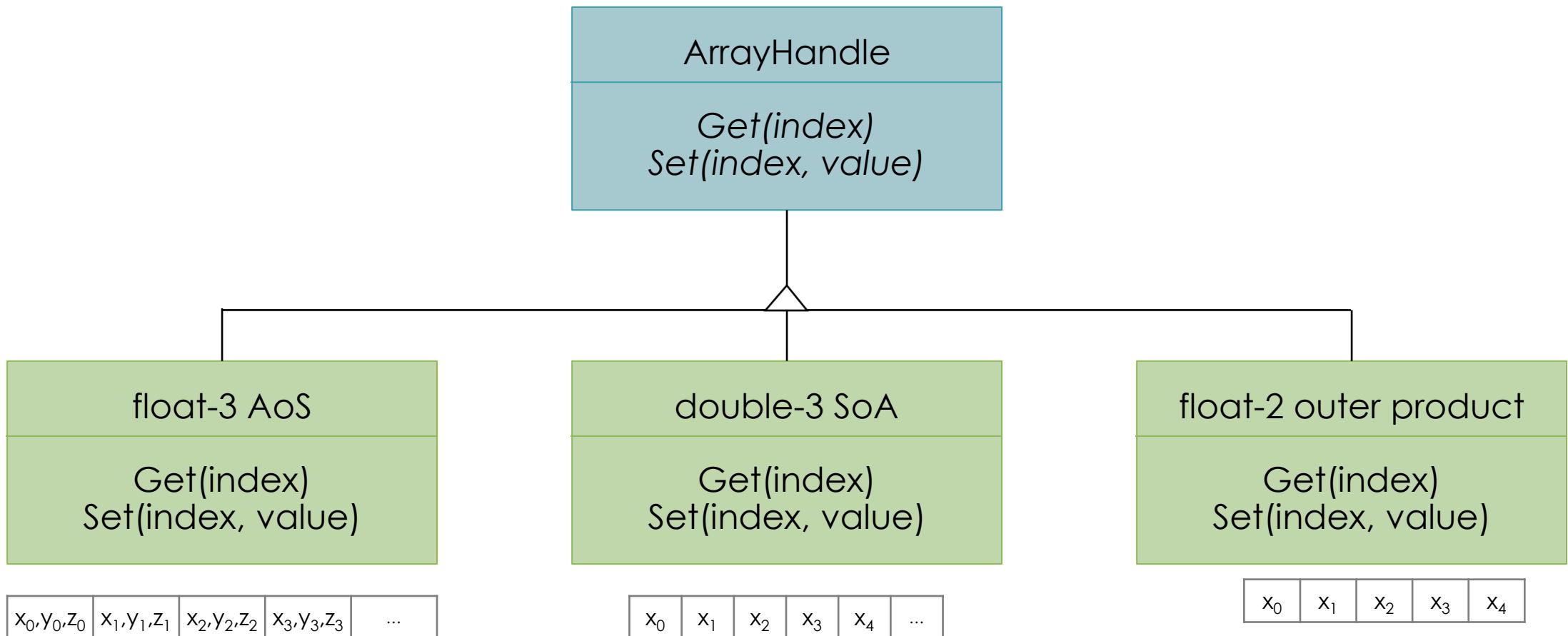
# Porting Ordeals



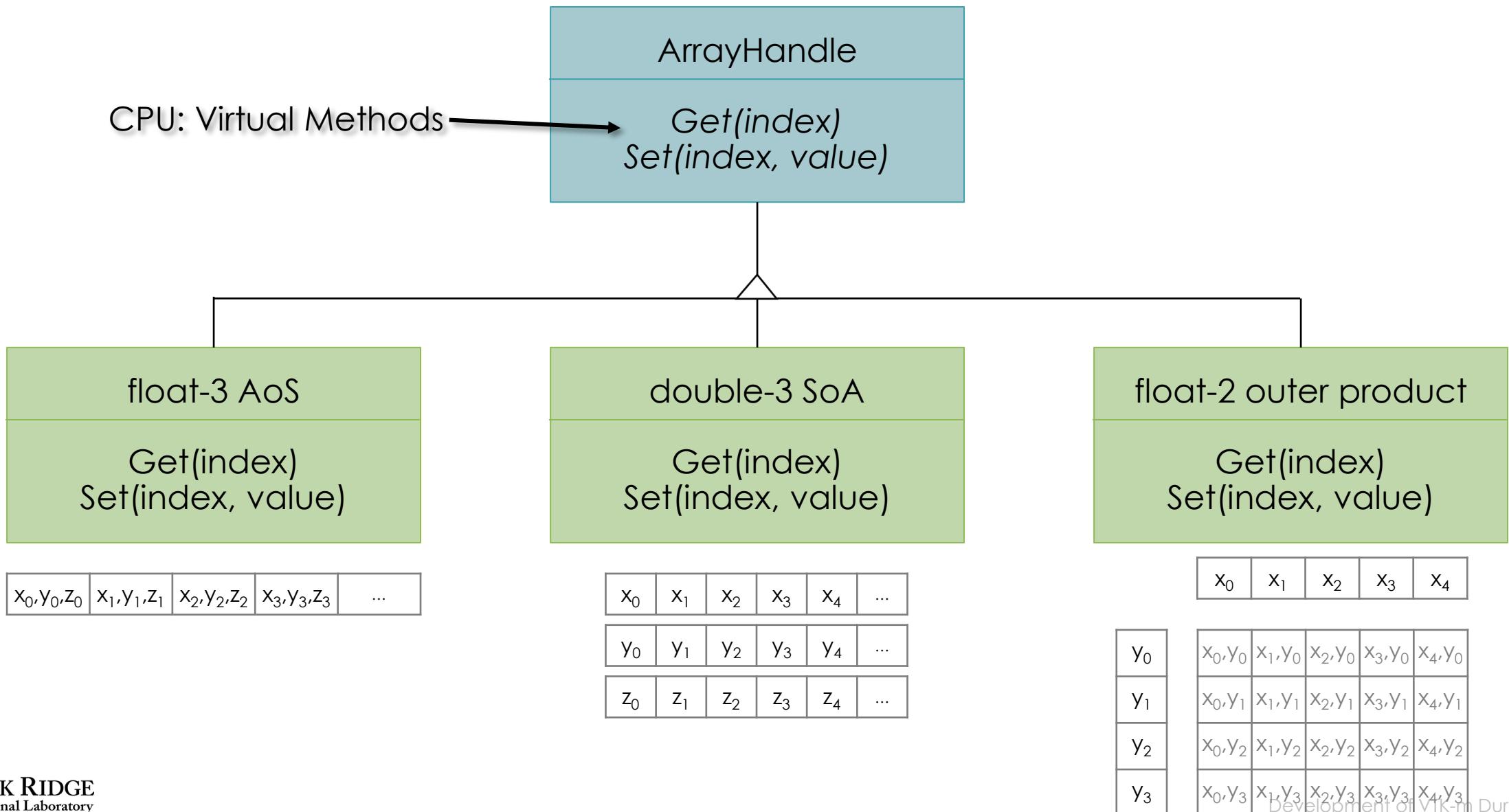
# Setting up for Success

- It is best to prepare for porting at the start
- Use software libraries for this purpose (e.g. Kokkos, RAJA)
  - Share the burden of porting
- On a brand-new system, get ready to encounter things that are broken or dysfunctional
  - These are very difficult to find because the problem is not in your code
- Have a working relationship with engineers from the vendor
  - It may not be possible to find errors without them
  - Even if you find errors, some may be beyond your capability to fix

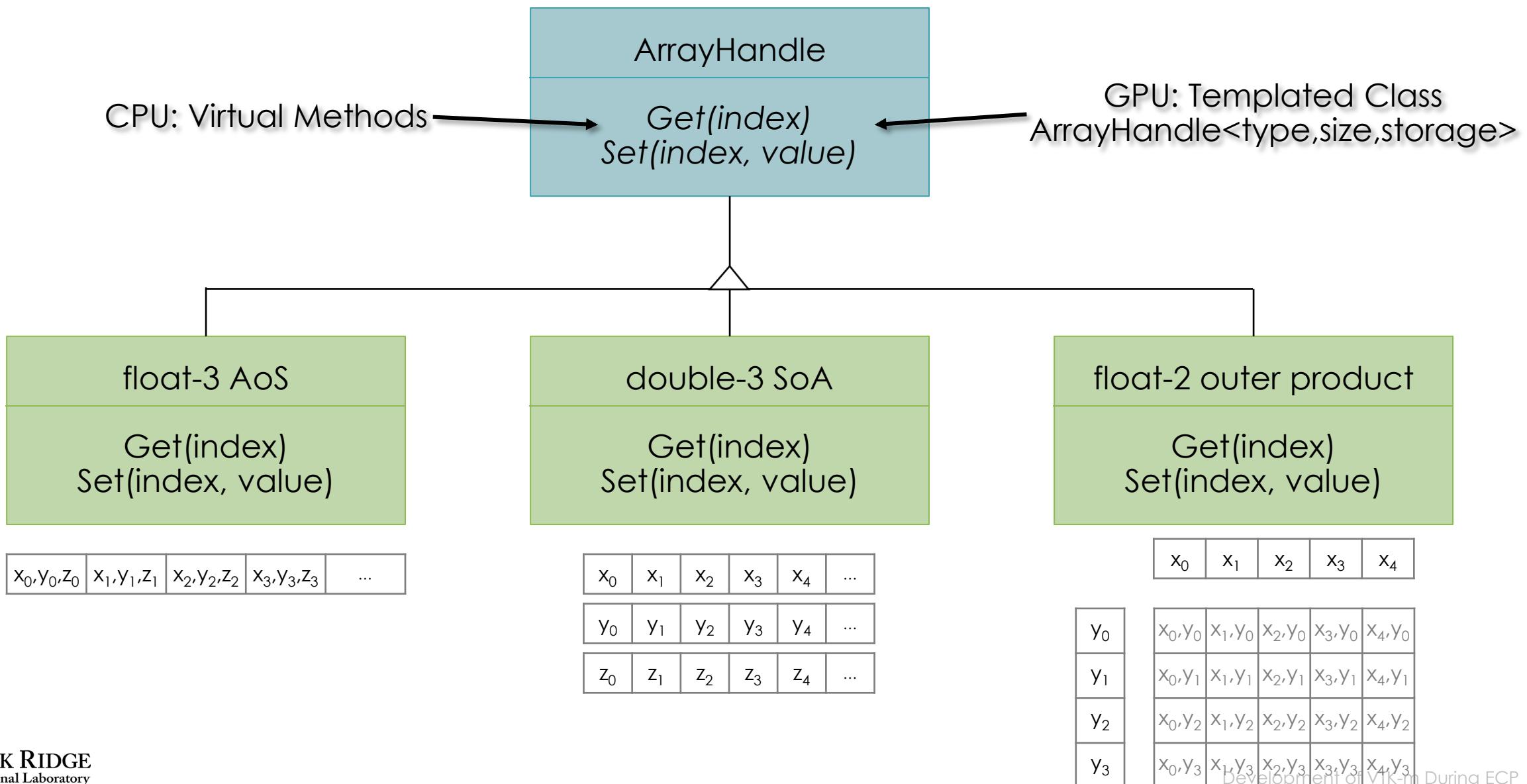
# What Went Awry: The Virtual Methods Saga



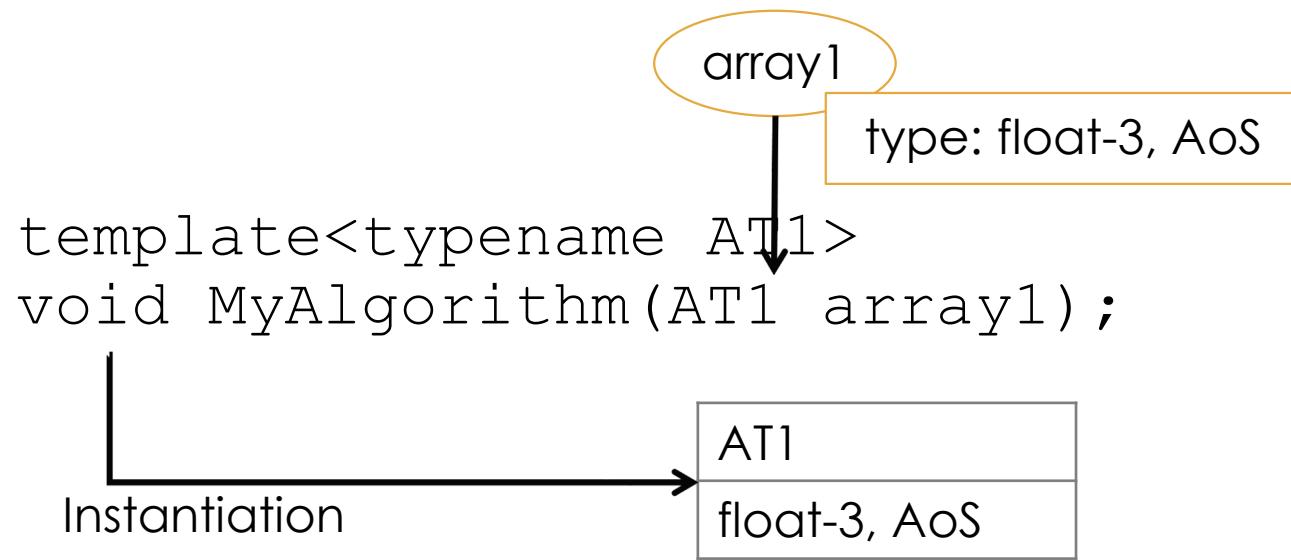
# What Went Awry: The Virtual Methods Saga



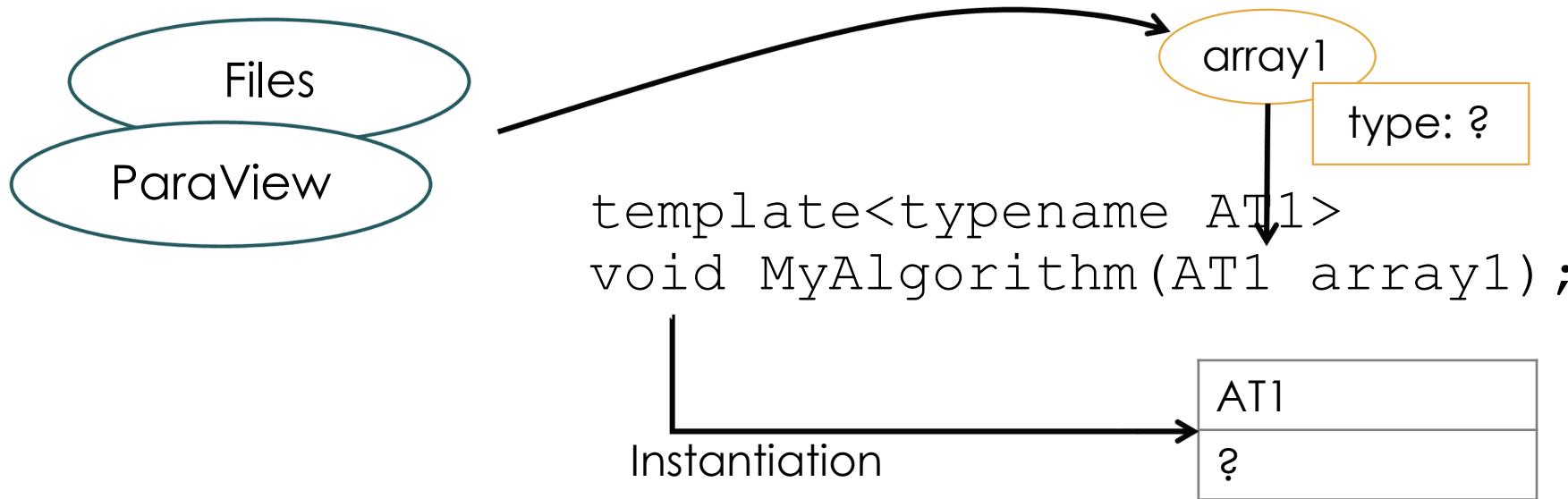
# What Went Awry: The Virtual Methods Saga



# What Went Awry: The Virtual Methods Saga



# What Went Awry: The Virtual Methods Saga



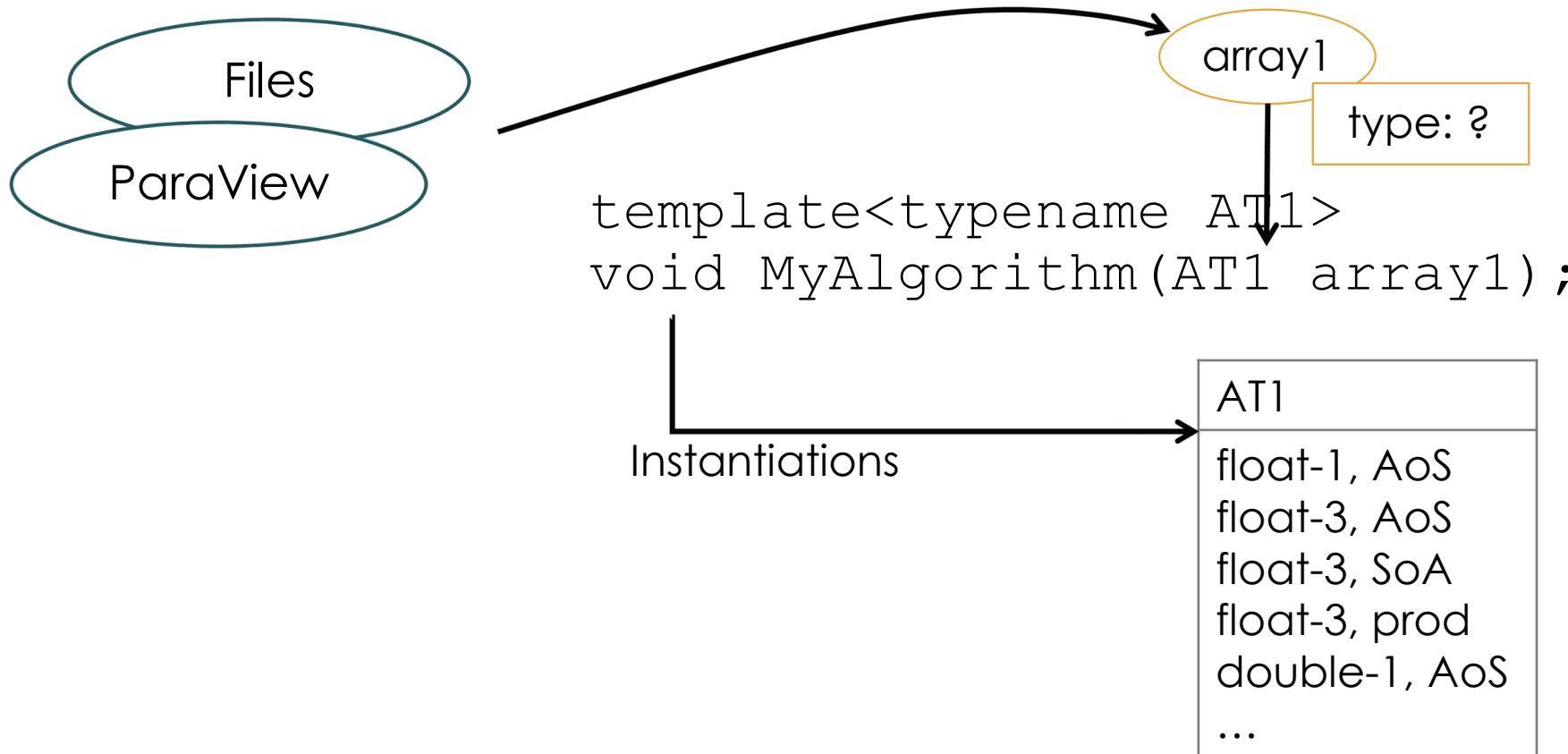
# Potential Solutions

- Use virtual methods on the GPU
  - I recommend from experience not doing this!
  - Virtual methods are not always supported (e.g., SYCL)
  - Even when supported, they have serious limitations that limit their effectiveness

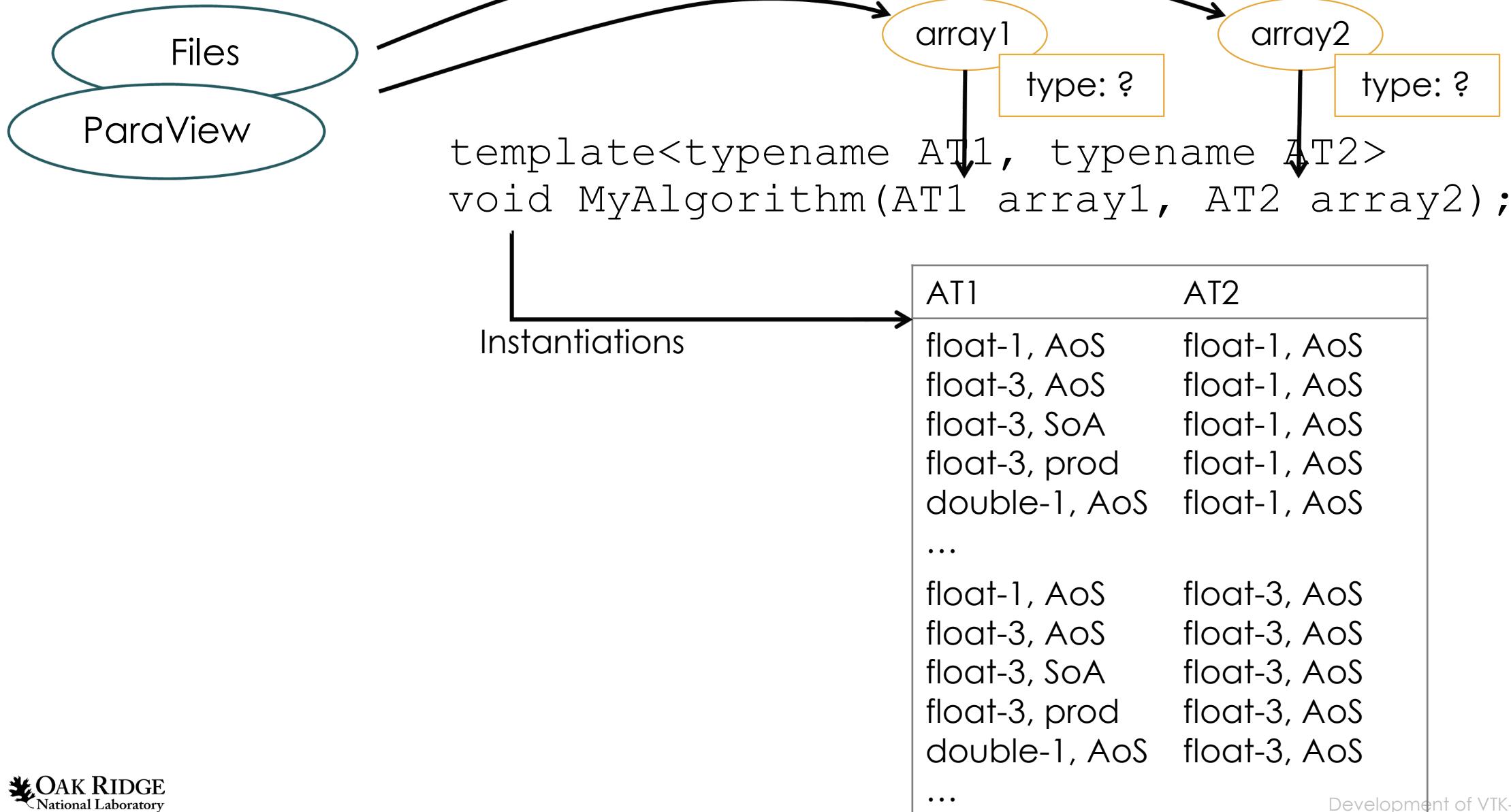
# Potential Solutions

- Use virtual methods on the GPU
  - I recommend from experience not doing this!
  - Virtual methods are not always supported (e.g., SYCL)
  - Even when supported, they have serious limitations that limit their effectiveness
- Make instantiations of every possible template instantiation

# What Went Awry: The Virtual Methods Saga



# What Went Awry: The Virtual Methods Saga



# Potential Solutions

- Use virtual methods on the GPU
  - I recommend from experience not doing this!
  - Virtual methods are not always supported (e.g., SYCL)
  - Even when supported, they have serious limitations that limit their effectiveness
- Make instantiations of every possible template instantiation
  - We had some templates that required *thousands* of instantiations
  - Very slow to compile and very memory intensive, especially with device compilers

# Potential Solutions

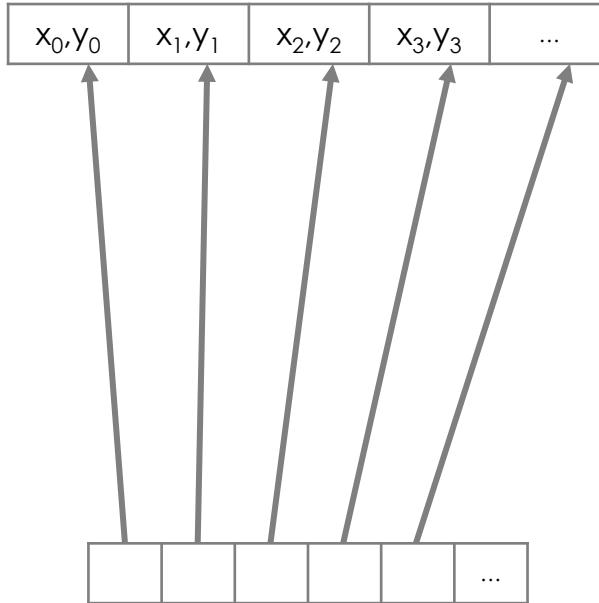
- Use virtual methods on the GPU
  - I recommend from experience not doing this!
  - Virtual methods are not always supported (e.g., SYCL)
  - Even when supported, they have serious limitations that limit their effectiveness
- Make instantiations of every possible template instantiation
  - We had some templates that required *thousands* of instantiations
  - Very slow to compile and very memory intensive, especially with device compilers
- JIT compilation
  - We did not try this

# Potential Solutions

- Use virtual methods on the GPU
  - I recommend from experience not doing this!
  - Virtual methods are not always supported (e.g., SYCL)
  - Even when supported, they have serious limitations that limit their effectiveness
- Make instantiations of every possible template instantiation
  - We had some templates that required *thousands* of instantiations
  - Very slow to compile and very memory intensive, especially with device compilers
- JIT compilation
  - We did not try this
- Assume the type, have a fallback
  - Fallback 1: Copy to a known type
  - Fallback 2: Make a general type (e.g., flexible indexing that does some arithmetic to find memory locations)

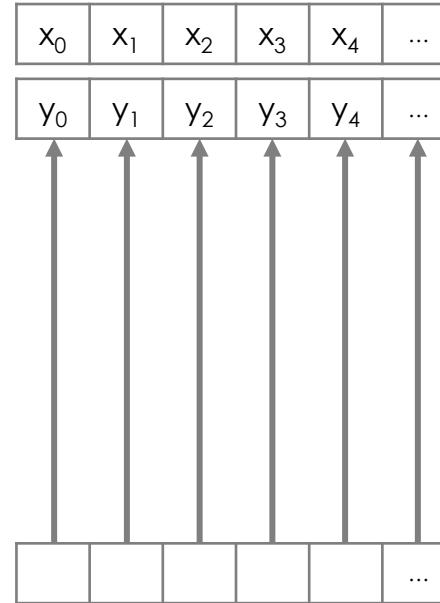
# Universal Array Access with Strides

float-2 AoS



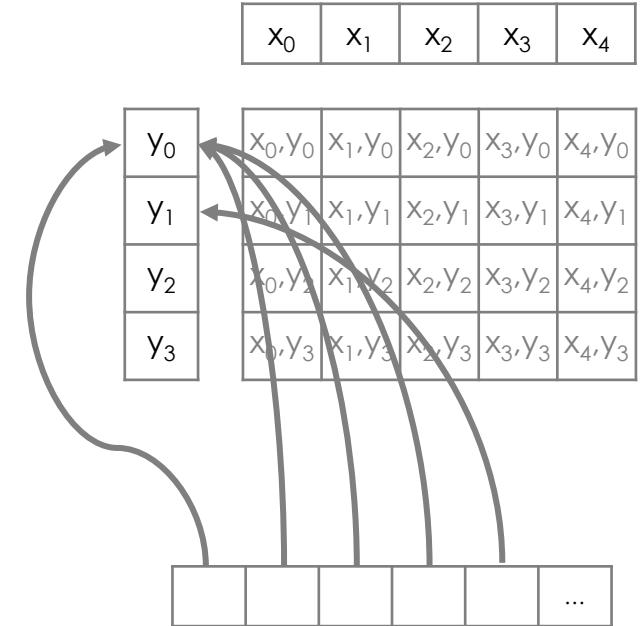
float strided  
Stride: 2, Offset 1,  
Div: 1, Mod: 0

float-2 SoA



float strided  
Stride: 1, Offset 0,  
Div: 1, Mod: 0

float-2 outer product

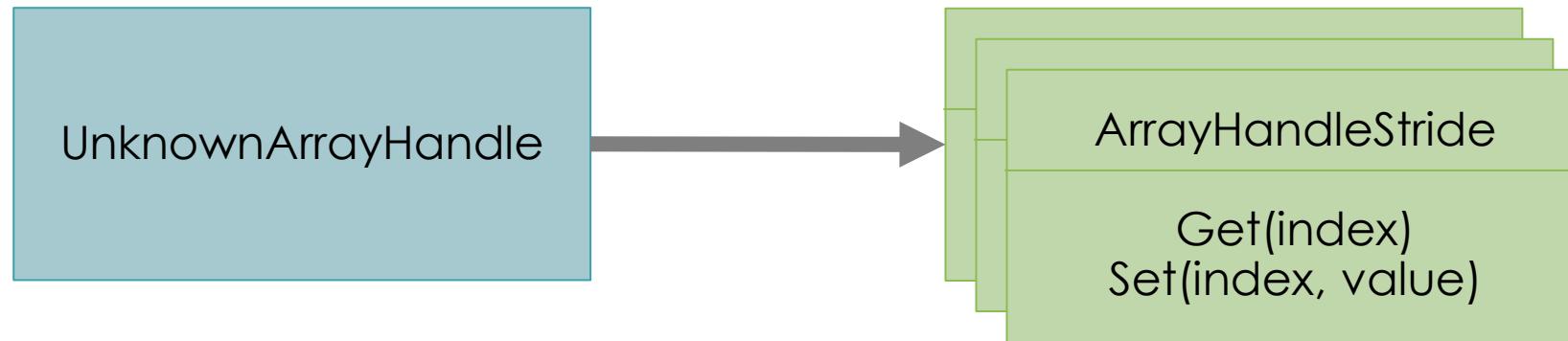


float strided  
Stride: 1, Offset 0,  
Div: 4, Mod: 0

# Getting an Unknown Array from an Abstract Object



# Getting an Unknown Array from an Abstract Object



# Final Remarks



# High Performance Software Foundation (HPFS)

- Formed at the close of ECP to help stewardship of software
- Provides a neutral hub for high-performance software projects
  - Builds, promotes, and advances a portable software stack for HPC
- We joined the HPSF under the rebranding **Viskores**
  - Renaming required to avoid trademark issues
- We expect to transition the software over the next year

# Summary

- Build tests early, test often!
  - The longer a problem goes, the harder it is to fix
  - Tests are worth it even in “research” code
- Use pull/merge requests
- Regular informal meetings go a long way to compensate for lack of formal development methods
  - More generally, pick and adapt agile practices that benefit you
- If you care about compute throughput, use a porting layer
- Beware: some of the most basic design patterns get tricky in a restricted GPU environment

# Acknowledgements

- This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering, and early testbed platforms, in support of the nation's exascale computing imperative.
- This work was supported by the U.S. Department of Energy (DOE) RAPIDS SciDAC project under contract number DE-AC05-00OR22725.
- This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.
- This research used resources of the Argonne Leadership Computing Facility, a U.S. Department of Energy (DOE) Office of Science user facility at Argonne National Laboratory and is based on research supported by the U.S. DOE Office of Science-Advanced Scientific Computing Research Program, under Contract No. DE-AC02-06CH11357.

For More Information Visit <http://m.vtk.org>

