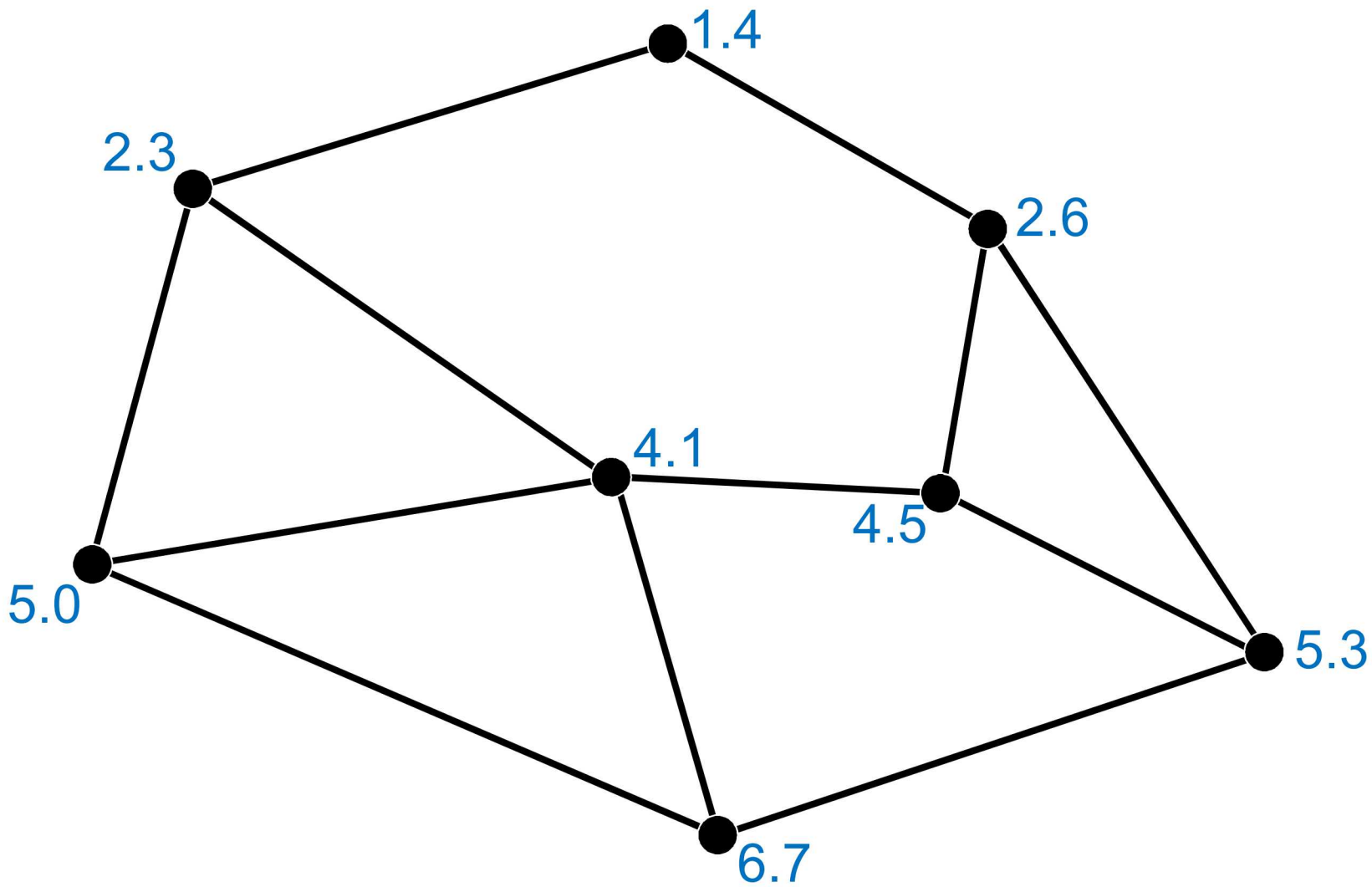
A ToolKit for Scientific Visualization on Many-Core Processors

Hank Childs University of Oregon
Kenneth Moreland Sandia National Laboratories
David Pugmire Oak Ridge National Laboratory
Robert Maynard Kitware, Inc.

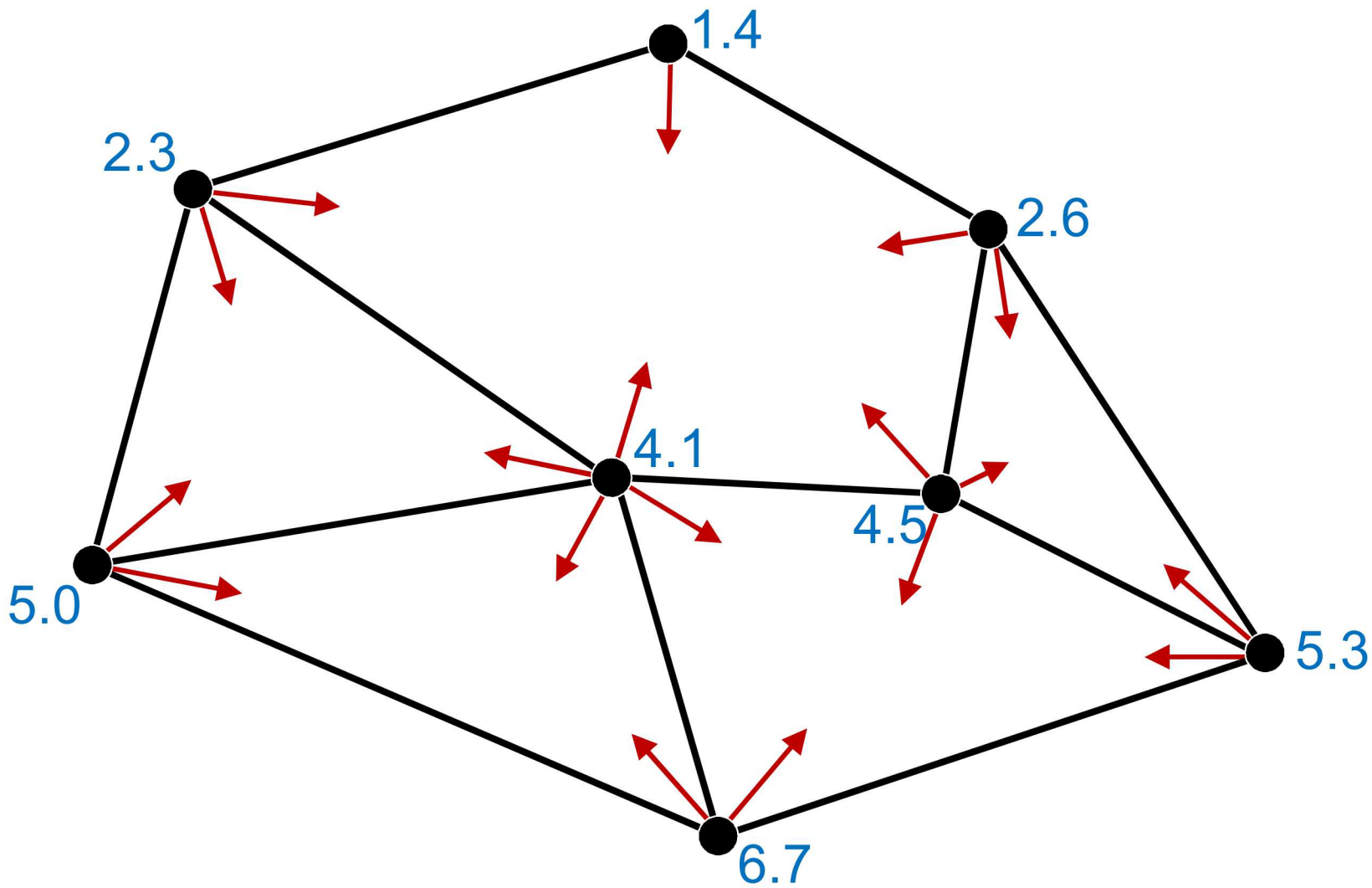
More Advanced Algorithm #1

SAND2019-12456PE

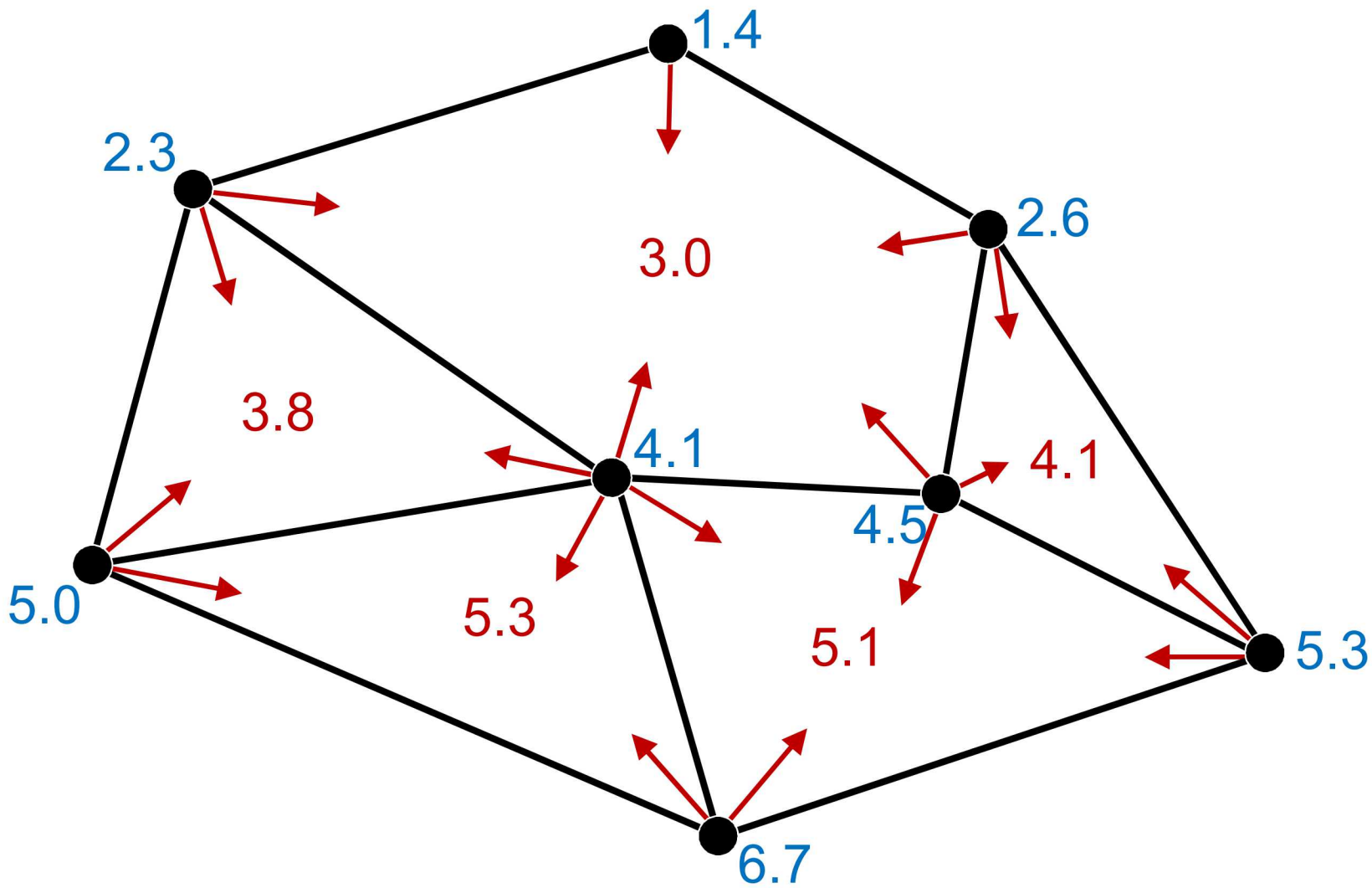
Motivating Example: Average Point Data to Cells



Motivating Example: Average Point Data to Cells



Motivating Example: Average Point Data to Cells



To follow along…

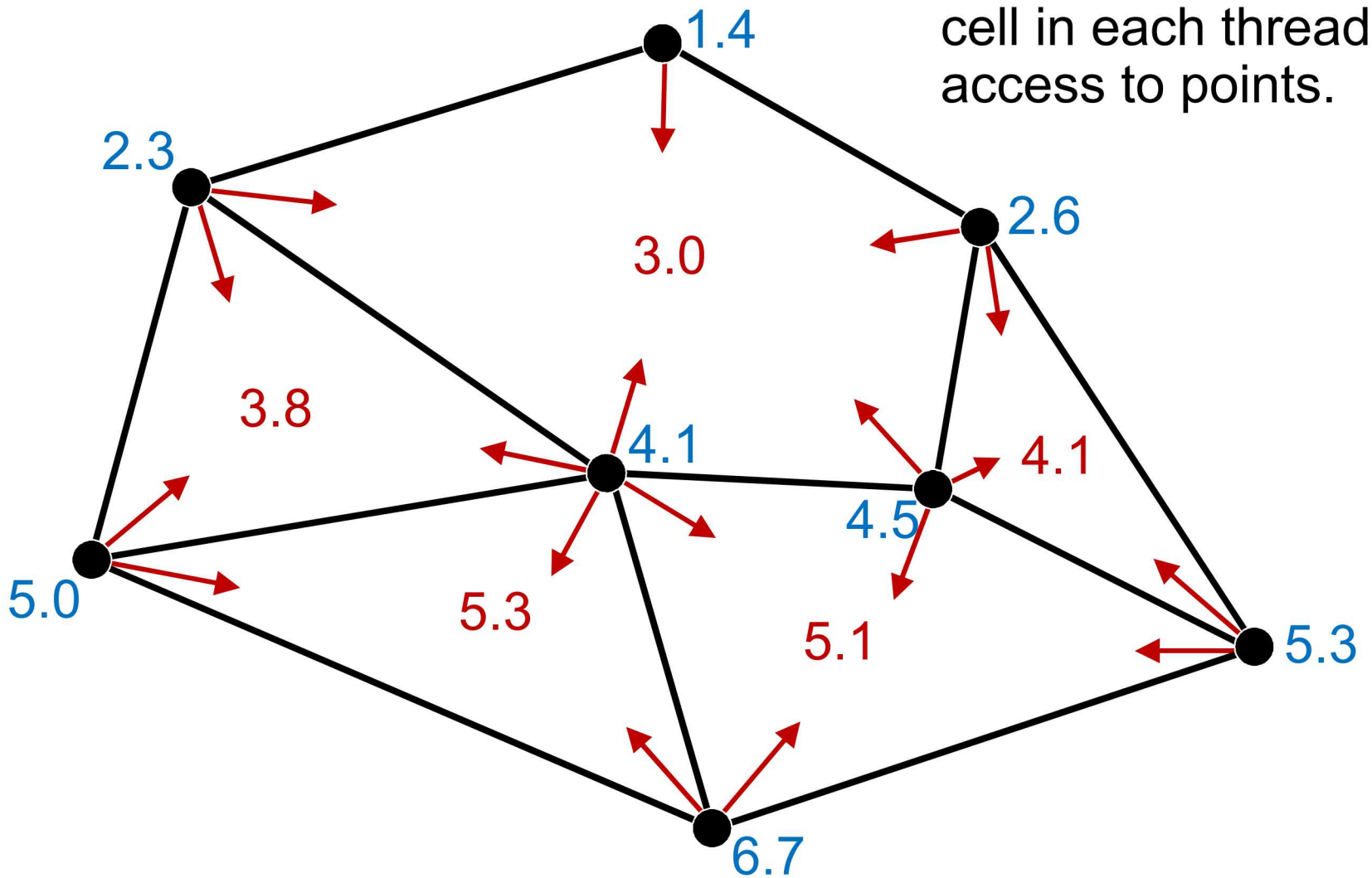
- The source code shown in this session is in the **tut_point_to_cell.cxx** file.

```
struct ConvertPointFieldToCells
{
```

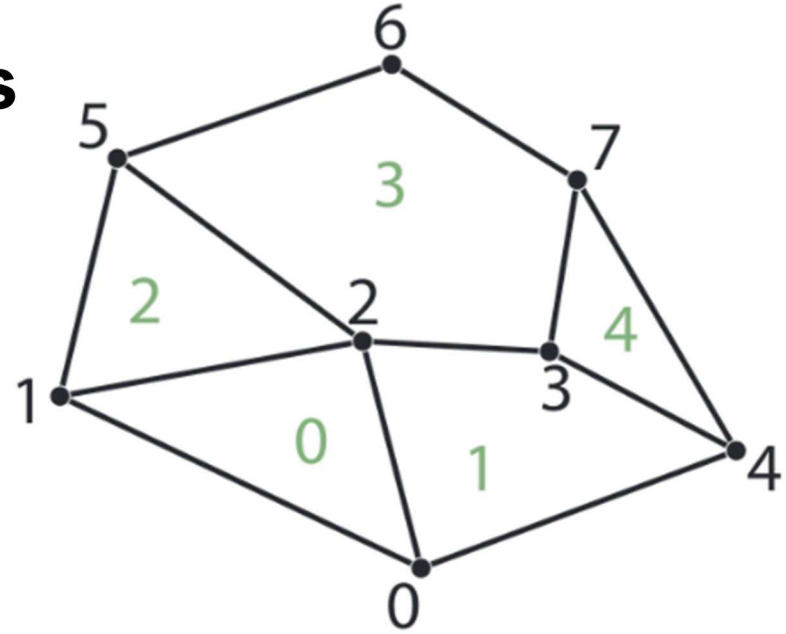
```
struct ConvertPointFieldToCells : vtkm::worklet::WorkletVisitCellsWithPoints
{
```

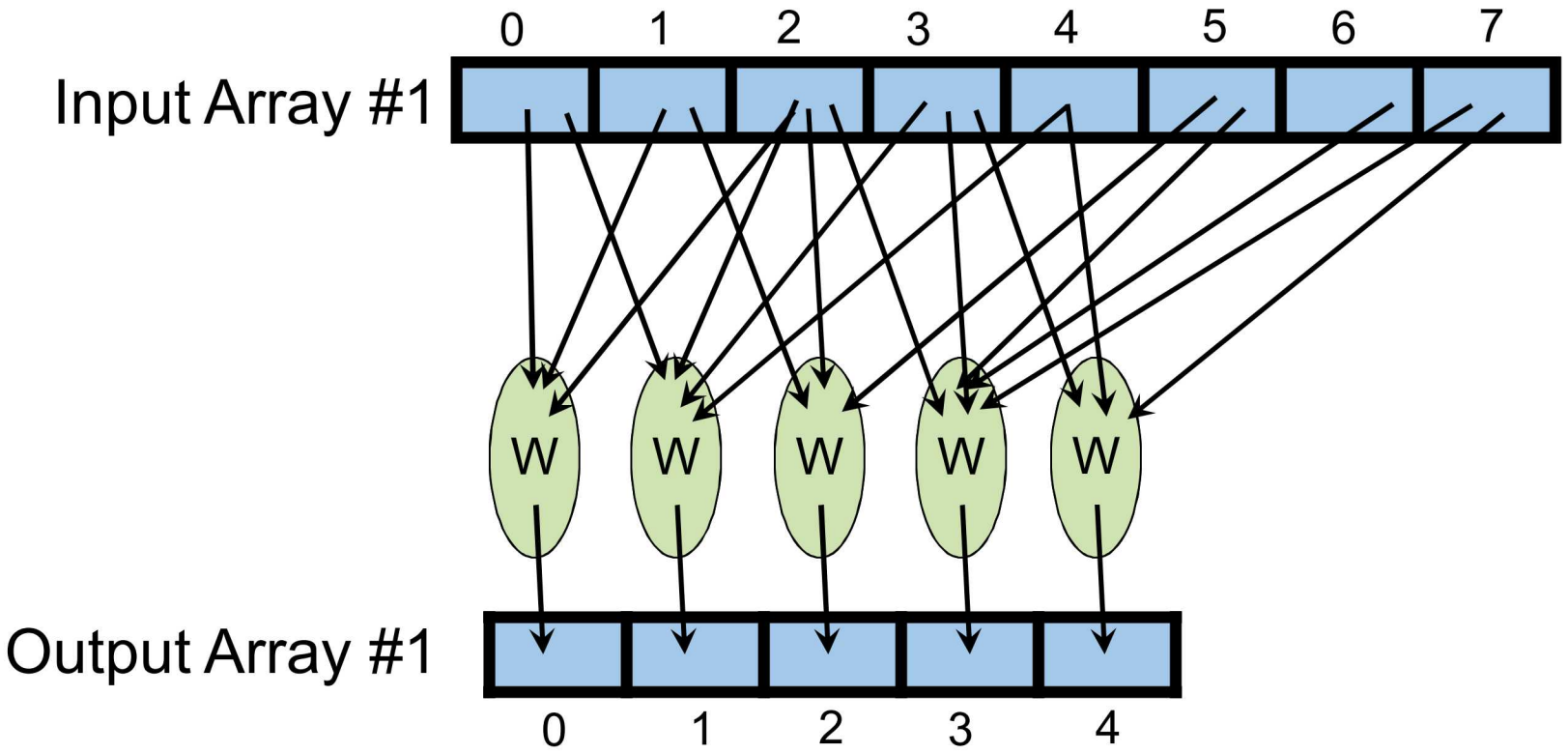
```
struct ConvertPointFieldToCells : vtkm::worklet::WorkletVisitCellsWithPoints
{
```

Worklet semantics visits a cell in each thread with access to points.

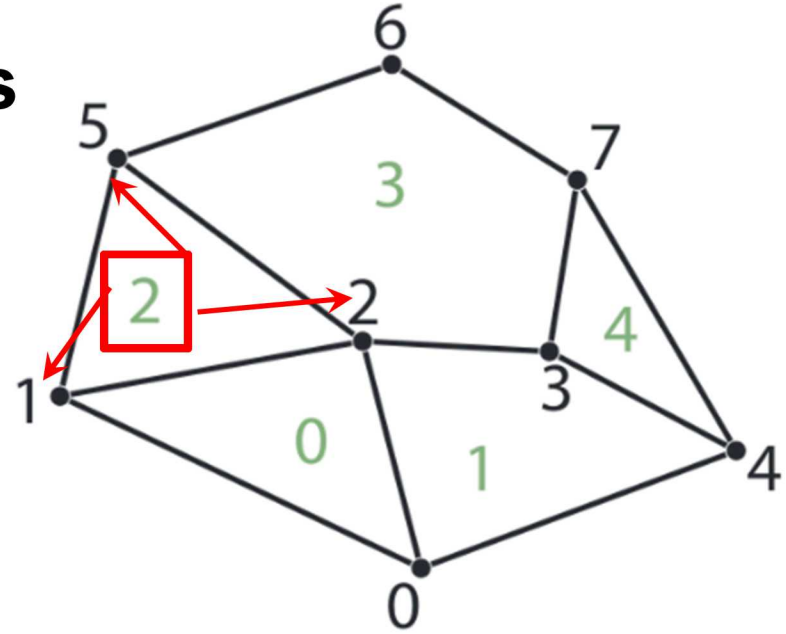


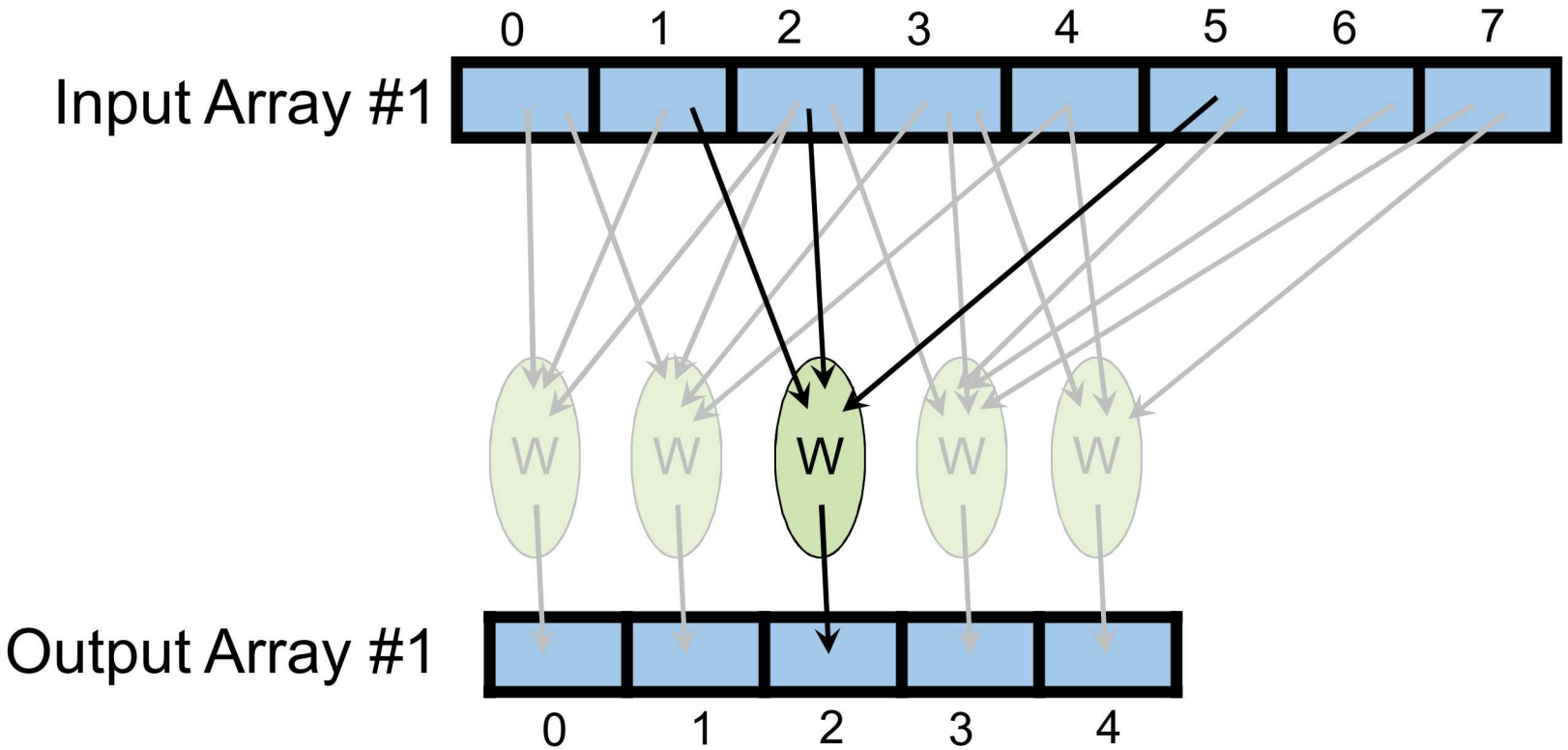
VTK-m Parallel Pattern: Visit Cells with Points





VTK-m Parallel Pattern: Visit Cells with Points





```
struct ConvertPointFieldToCells : vtkm::worklet::WorkletVisitCellsWithPoints
{
```

```
struct ConvertPointFieldToCells : vtkm::worklet::WorkletVisitCellsWithPoints
{
  using ControlSignature = void(CellSetIn topology,
                                FieldInPoint inPointField,
                                FieldOutCell outCellField);
```

```
struct ConvertPointFieldToCells : vtkm::worklet::WorkletVisitCellsWithPoints
{
  using ControlSignature = void(CellSetIn topology,
                                FieldInPoint inPointField,
                                FieldOutCell outCellField);
```

To follow topology connections, VTK-m must be given the cell set.

```
struct ConvertPointFieldToCells : vtkm::worklet::WorkletVisitCellsWithPoints
{
  using ControlSignature = void(CellSetIn topology,
                                FieldInPoint inPointField,
                                FieldOutCell outCellField);
```

To follow topology connections, VTK-m must be given the cell set.

Field in/out now has to specify whether they come from the points or the cells.

```
struct ConvertPointFieldToCells : vtkm::worklet::WorkletVisitCellsWithPoints
{
  using ControlSignature = void(CellSetIn topology,
                                FieldInPoint inPointField,
                                FieldOutCell outCellField);
```

```
struct ConvertPointFieldToCells : vtkm::worklet::WorkletVisitCellsWithPoints
{
  using ControlSignature = void(CellSetIn topology,
                                FieldInPoint inPointField,
                                FieldOutCell outCellField);

  using InputDomain = _1;
```

```
struct ConvertPointFieldToCells : vtkm::worklet::WorkletVisitCellsWithPoints
{
  using ControlSignature = void(CellSetIn topology,
                                FieldInPoint inPointField,
                                FieldOutCell outCellField);

  using InputDomain = _1;
```

Specifies which argument of the ControlSignature is used for scheduling.

```
struct ConvertPointFieldToCells : vtkm::worklet::WorkletVisitCellsWithPoints
{
  using ControlSignature = void(CellSetIn topology,
                                FieldInPoint inPointField,
                                FieldOutCell outCellField);

  using InputDomain = _1;
```

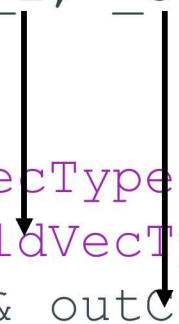
```
struct ConvertPointFieldToCells : vtkm::worklet::WorkletVisitCellsWithPoints
{
  using ControlSignature = void(CellSetIn topology,
                                FieldInPoint inPointField,
                                FieldOutCell outCellField);
  using ExecutionSignature = void(_2, _3);
  using InputDomain = _1;
```

```
struct ConvertPointFieldToCells : vtkm::worklet::WorkletVisitCellsWithPoints
{
  using ControlSignature = void(CellSetIn topology,
                                FieldInPoint inPointField,
                                FieldOutCell outCellField);
  using ExecutionSignature = void(_2, _3);
  using InputDomain = _1;
```

Specifies which arguments of the ControlSignature are passed to the worklet's operator.

```
struct ConvertPointFieldToCells : vtkm::worklet::WorkletVisitCellsWithPoints
{
  using ControlSignature = void(CellSetIn topology,
                                FieldInPoint inPointField,
                                FieldOutCell outCellField);
  using ExecutionSignature = void(_2, _3);
  using InputDomain = _1;

  template <typename InPointFieldVecType, typename OutCellFieldType>
  void operator()(const InPointFieldVecType& inPointFieldVec,
                  OutCellFieldType& outCellField) const
  {
```

```
struct ConvertPointFieldToCells : vtkm::worklet::WorkletVisitCellsWithPoints
{
  using ControlSignature = void(CellSetIn topology,
                                FieldInPoint inPointField,
                                FieldOutCell outCellField);
  using ExecutionSignature = void(_2, _3);
  using InputDomain = _1;

  template <typename InPointFieldVecType, typename OutCellFieldType>
  void operator()(const InPointFieldVecType& inPointFieldVec,
                  OutCellFieldType& outCellField) const
  {
```



```
struct ConvertPointFieldToCells : vtkm::worklet::WorkletVisitCellsWithPoints
{
  using ControlSignature = void(CellSetIn topology,
                                FieldInPoint inPointField,
                                FieldOutCell outCellField);
  using ExecutionSignature = void(_2, _3);
  using InputDomain = _1;

  template <typename InPointFieldVecType, typename OutCellFieldType>
  void operator()(const InPointFieldVecType& inPointFieldVec,
                  OutCellFieldType& outCellField) const
  {
```

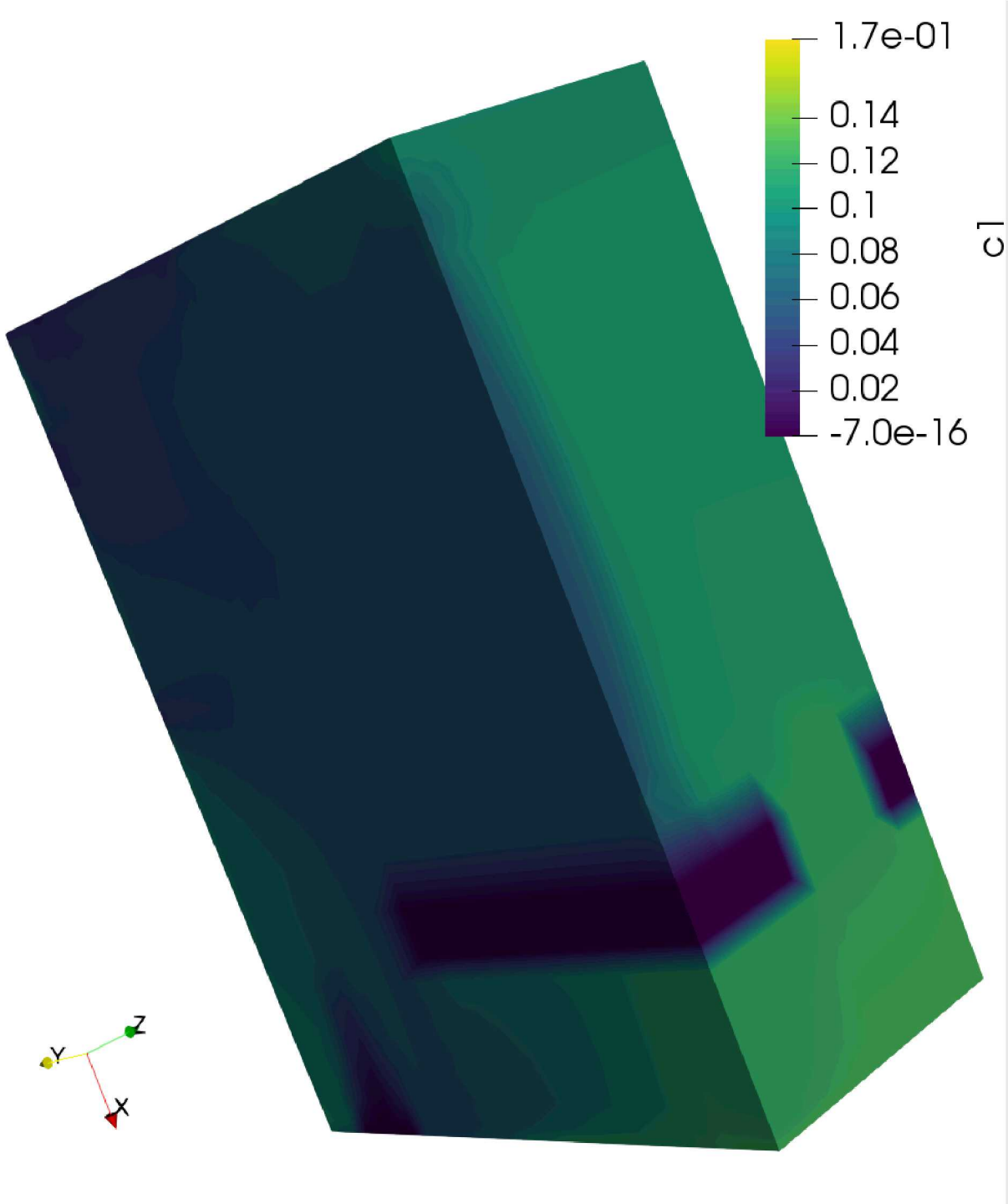
Input point field is passed in a Vec-like object containing the values for all connected points.

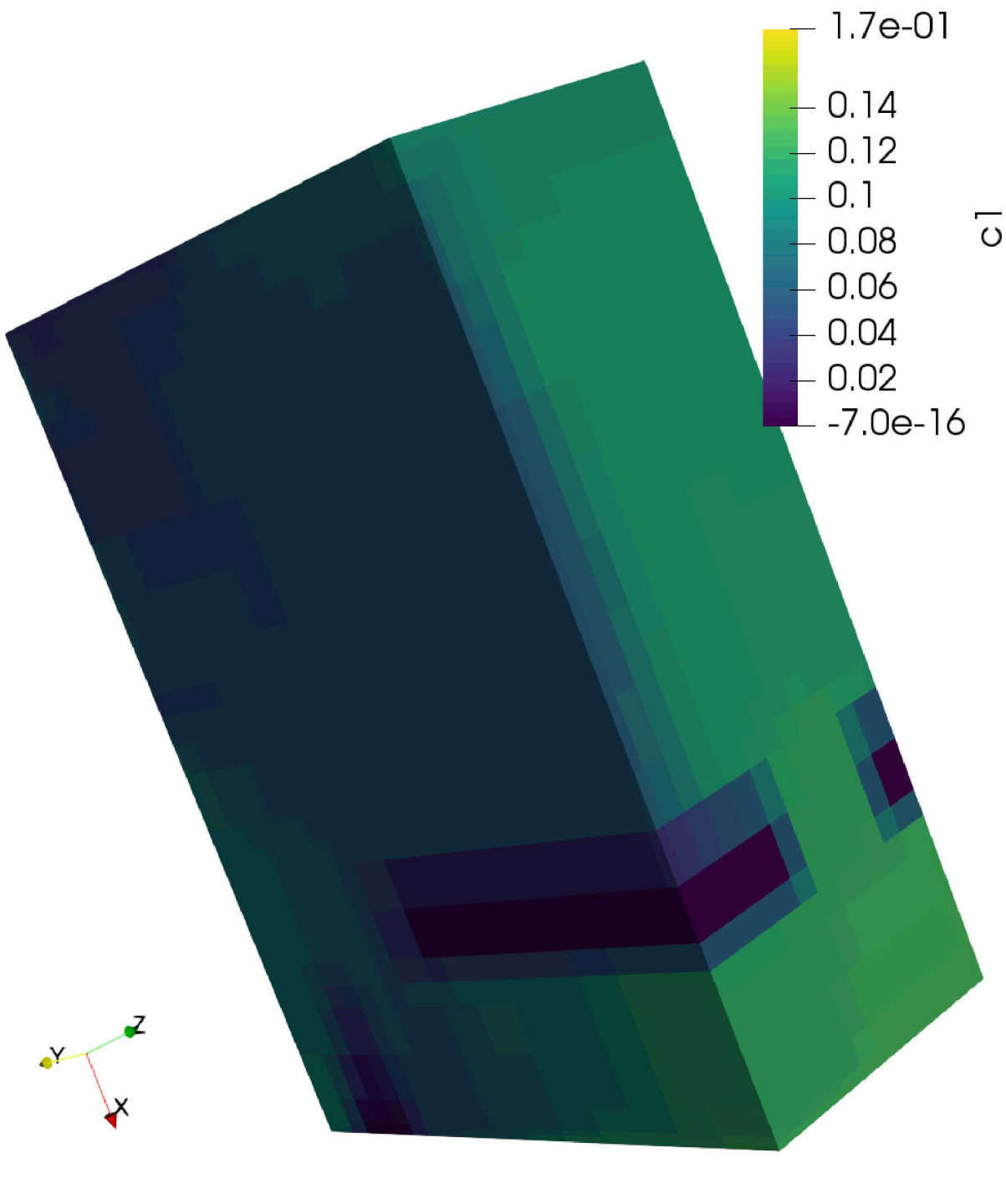
```
struct ConvertPointFieldToCells : vtkm::worklet::WorkletVisitCellsWithPoints
{
  using ControlSignature = void(CellSetIn topology,
                                FieldInPoint inPointField,
                                FieldOutCell outCellField);
  using ExecutionSignature = void(_2, _3);
  using InputDomain = _1;

  template <typename InPointFieldVecType, typename OutCellFieldType>
  void operator()(const InPointFieldVecType& inPointFieldVec,
                  OutCellFieldType& outCellField) const
  {
    vtkm::IdComponent numPoints = inPointFieldVec.GetNumberOfComponents();

    outCellField = OutCellFieldType(0);
    for (vtkm::IdComponent pointIndex = 0; pointIndex < numPoints; ++pointIndex)
    {
      outCellField = outCellField + inPointFieldVec[pointIndex];
    }
    outCellField = outCellField / OutCellFieldType(numPoints);
  }
};
```

```
template<typename ArrayHandleType, typename Policy>
VTKM_CONT cont::DataSet ConvertPointFieldToCells::DoExecute(
  const vtkm::cont::DataSet& inDataSet,
  const ArrayHandleType& inField,
  const vtkm::filter::FieldMetadata& fieldMetadata,
  vtkm::filter::PolicyBase<Policy> policy)
{
  VTKM_IS_ARRAY_HANDLE(ArrayHandleType);

  using ValueType = typename ArrayHandleType::ValueType;

  vtkm::cont::ArrayHandle<ValueType> outField;
  this->Invoke(vtkm::worklet::ConvertPointFieldToCells{},
               vtkm::filter::ApplyPolicyCellSet(inDataSet.GetCellSet(), policy),
               inField,
               outField);
```

```
template<typename ArrayHandleType, typename Policy>
VTKM_CONT cont::DataSet ConvertPointFieldToCells::DoExecute(
  const vtkm::cont::DataSet& inDataSet,
  const ArrayHandleType& inField,
  const vtkm::filter::FieldMetadata& fieldMetadata,
  vtkm::filter::PolicyBase<Policy> policy)
{
  VTKM_IS_ARRAY_HANDLE(ArrayHandleType);

  using ValueType = typename ArrayHandleType::ValueType;

  vtkm::cont::ArrayHandle<ValueType> outField;
  this->Invoke(vtkm::worklet::ConvertPointFieldToCells{},
               vtkm::filter::ApplyPolicyCellSet(inDataSet.GetCellSet(), policy),
               inField,
               outField);
```

You should apply the policy to the cell set. (Specifies which cell set types to build for.)


1.7e-01
0.14
0.12
0.1
0.08
0.06
0.04
0.02
-7.0e-16
c1
Z
Y
X


1.7e-01
0.14
0.12
0.1
0.08
0.06
0.04
0.02
-7.0e-16
c1
Z
Y
X

An assignment for you: convert cell data to point data

- Can use **tut_point_to_cell.cxx** as a starting point.

```
struct ConvertPointFieldToCells : vtkm::worklet::WorkletVisitCellsWithPoints
{
  using ControlSignature = void(CellSetIn topology,
                                FieldInPoint inPointField,
                                FieldOutCell outCellField);
  using ExecutionSignature = void(_2, _3);
  using InputDomain = _1;

  template <typename InPointFieldVecType, typename OutCellFieldType>
  void operator()(const InPointFieldVecType& inPointFieldVec,
                  OutCellFieldType& outCellField) const
  {
    vtkm::IdComponent numPoints = inPointFieldVec.GetNumberOfComponents();

    outCellField = OutCellFieldType(0);
    for (vtkm::IdComponent pointIndex = 0; pointIndex < numPoints; ++pointIndex)
    {
      outCellField = outCellField + inPointFieldVec[pointIndex];
    }
    outCellField = outCellField / OutCellFieldType(numPoints);
  }
};
```

```
struct ConvertPointFieldToCells : vtkm::worklet::WorkletVisitCellsWithPoints
{
  using ControlSignature = void(CellSetIn topology,
                                FieldInPoint inPointField,
                                FieldOutCell outCellField);
  using ExecutionSignature = void(_2, _3);
  using InputDomain = _1;

  template <typename InPointFieldVecType, typename OutCellFieldType>
  void operator()(const InPointFieldVecType& inPointFieldVec,
                  OutCellFieldType& outCellField) const
  {
    vtkm::IdComponent numPoints = inPointFieldVec.GetNumberOfComponents();

    outCellField = OutCellFieldType(0);
    for (vtkm::IdComponent pointIndex = 0; pointIndex < numPoints; ++pointIndex)
    {
      outCellField = outCellField + inPointFieldVec[pointIndex];
    }
    outCellField = outCellField / OutCellFieldType(numPoints);
  }
};
```

Need to visit points with cells

```
struct ConvertPointFieldToCells : vtkm::worklet::WorkletVisitCellsWithPoints
{
  using ControlSignature = void(CellSetIn topology,
                                FieldInPoint inPointField,
                                FieldOutCell outCellField);
  using ExecutionSignature = void(_2, _3);
  using InputDomain = _1;

  template <typename InPointFieldVecType, typename OutCellFieldType>
  void operator()(const InPointFieldVecType& inPointFieldVec,
                  OutCellFieldType& outCellField) const
  {
    vtkm::IdComponent numPoints = inPointFieldVec.GetNumberOfComponents();

    outCellField = OutCellFieldType(0);
    for (vtkm::IdComponent pointIndex = 0; pointIndex < numPoints; ++pointIndex)
    {
      outCellField = outCellField + inPointFieldVec[pointIndex];
    }
    outCellField = outCellField / OutCellFieldType(numPoints);
  }
};
```

Need to visit points with cells

Reverse point/cell semantics