SAND2015-9294PE

Exceptional service in the national interest

Sandia National Laboratories

Intrepid2

Integrating Kokkos with Intrepid

U.S. DEPARTMENT OF ENERGY
NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

Intrepid

Sandia National Laboratories

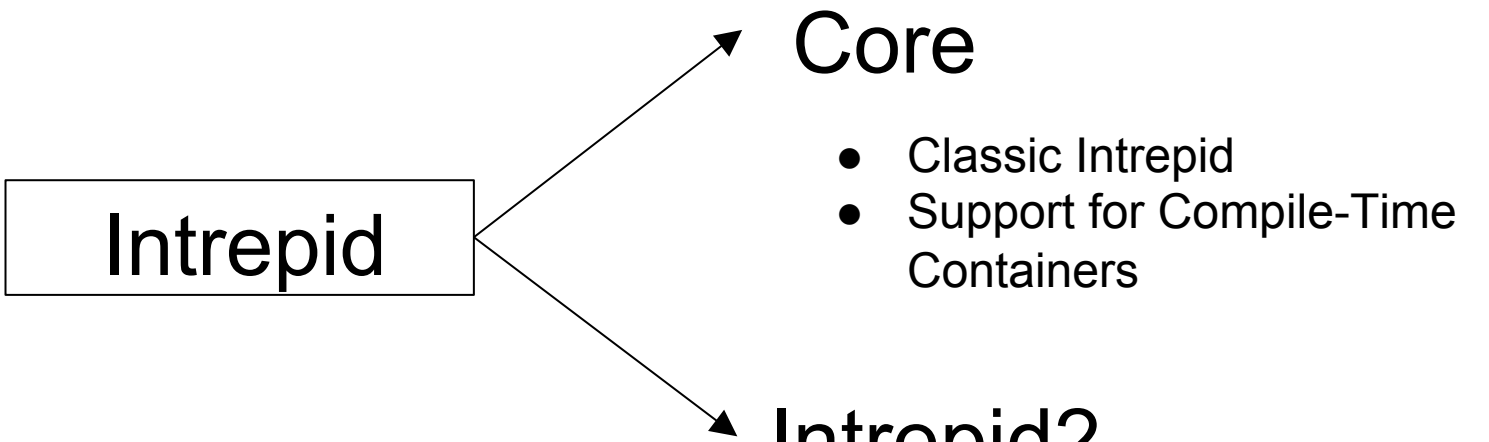
Intrepid package is a library of tools for compatible discretizations of Partial Differential Equations (PDEs)

Used in: Drekar, Albany and other codes

Current Status of Intrepid

Sandia National Laboratories

Intrepid is currently split into two sub-packages but will soon become its own package



Core

- Classic Intrepid
- Support for Compile-Time Containers

Intrepid2

- Classic Intrepid
- Support for Compile-Time Containers
- Kokkos Parallel Functors

Support for Compile-Time Containers

Sandia National Laboratories

The following code causes error with compile time rank containers

```
switch(rank){
  case 2:
     container(1,5)=container(1,5)+10;
  break;

  case 3:
     container(1,5,7)=container(1,5,7)+10;
   break;
}
```

Support for Compile-Time Containers

Sandia National Laboratories

Our solution to this issue is to use wrapper class with partial template specialization on rank

The code becomes

```
ArrayWrapper<Scalar,ArrayContainer, Rank<ArrayContainer >::value,
false>containerWrap(container)

switch(rank){
  case 2:
     containerWrap(1,5)=containerWrap(1,5)+10;
  break;

  case 3:
     containerWrap(1,5,7)=containerWrap(1,5,7)+10;
  break;
}
```

Support for Compile-Time Containers

Sandia National Laboratories

```
template<class Scalar,class ArrayType>
struct ArrayWrapper<Scalar,ArrayType,5,false> {
   ArrayType& view;

  typedef typename Return_Type<ArrayType, Scalar>::return_type rtype;
  ArrayWrapper( ArrayType& view_):view(view_) {};
  index_type dimension(int i)const{
            return view.dimension(i);
  }
  int rank()const{
            return 5;
  }
  rtype operator() (index_type i0, index_type i1=0, index_type i2 = 0,
                    index_type i3 = 0, index_type i4 = 0, index_type i5 = 0,
                    index_type i6 = 0, index_type i7 = 0) const {
            return view(i0,i1,i2,i3,i4);
  }
};
```

Intrepid is a Good Candidate for Parallelization

Sandia National Laboratories

Intrepid has a large number of nested loops without loop carried dependencies

```
for(index_type cell = 0; cell < numCells; cell++) {
        for(index_type point = 0; point < numPoints; point++) {
          for(index_type row = 0; row < matDim; row++) {
            outputDataWrap(cell, point, row) = \
              inputDataLeftWrap(cell, point, row)*inputDataRightWrap(cell, point, row);
          } // Row-loop
        } // P-loop
      }// C-loop
```

Parallel Functors Intrepid2

Sandia National Laboratories

```
template <class Scalar,class ArrayOutFieldsWrap,class ArrayInDataWrap,class
ArrayInFieldsWrap,class ArrayInFields>
struct matvecProductDataField_3_3 {
  ArrayOutFieldsWrap outputFields;
  ArrayInDataWrap inputData;
  ArrayInFieldsWrap inputFields;
typedef typename conditional_eSpace<ArrayInFields>::execution_space execution_space;
  matvecProductDataField_3_3 (ArrayOutFieldsWrap outputFields_, ArrayInDataWrap
inputData_,ArrayInFieldsWrap inputFields_) :
              outputFields (outputFields_),inputData (inputData_),inputFields(inputFields_)
  {}
  KOKKOS_INLINE_FUNCTION
  void operator () (const index_type cell) const {
              for(index_type field = 0; field < outputFields.dimension(1); field++) {
              for(index_type point = 0; point < outputFields.dimension(2); point++) {
              for(index_type row = 0; row < outputFields.dimension(3); row++) {
              outputFields(cell, field, point, row) = \
              inputData(cell, point, row)*inputFields(field, point, row);
              } // Row-loop
              } // P-loop
              } // F-loop
   }
};
```

Parallel Functors Intrepid2

Sandia National Laboratories

- We check to see if the container being passed to the kokkos kernel contains a member execution_space
- If the container does not have execution_space it is assumed that the code should be run on the host in serial
- Containers that should work include Kokkos::View, MDField, FieldContainer_Kokkos
- To do this we use

```
typedef typename conditional_eSpace<ArrayInFields>::execution_space execution_space;
```