Trilinos Users Group Meeting

October 30, 2012

Bill Spotz, 1442

Teuchos::MDArrays

SAND 2012-9340C

Sandia National Laboratories

Motivation

- **Finite difference applications**
  - Other structured data applications
- **Data storage for proposed Tpetra::MDVector**
  - Leverage “strided MPI”
- **Python interface**
  - NumPy ndarrays
  - ODIN (Enthought)
- **Higher level interface than Kokkos::MDArray**
  - Interoperability…
- **Leverage Teuchos::Array capabilities/interface**
- **Simple extraction of sub-arrays**
- **Efficient (i,j,k,…) indexing**

Quick Overview of Teuchos::Array types

- **Teuchos::Array<T>**
  - Wrapper around std::vector<T>
  - Adds some memory management
  - Adds debugging tools (e.g. -D Teuchos_ENABLE_ABC:BOOL=ON)
- **Teuchos::ArrayView<T>**
  - Array capabilities for pre-existing buffers of data
  - Memory management under the covers
- **Teuchos::ArrayRCP<T>**
  - Reference-counted arrays
  - T* … does it point to a scalar T or an array of T?

- **Teuchos::MDArray<T>:** stores a Teuchos::Array<T>
- **Teuchos::MDArrayView<T>:** stores a Teuchos::ArrayView<T>
- **Teuchos::MDArrayRCP<T>:** stores a Teuchos::ArrayRCP<T>

Constructing MDArray & MDArrayView

- **MDArray<T>(const ArrayView<size_type> & dims)**
  - Example:

```
typedef Teuchos::MDArray<double>::size_type ord;
// Equivalently: typedef Teuchos::Ordinal ord;
using Teuchos::tuple

Teuchos::MDArray<double> a(tuple<ord>(5,6,7));
```

- **Other optional constructor arguments:**
  - const EStorageOrder storageOrder;
    - C_ORDER, FORTRAN_ORDER, ROW_MAJOR, COLUMN MAJOR, LAST_INDEX_FASTEST, FIRST_INDEX_FASTEST, DEFAULT_ORDER
  - const T & value;

- **MDArrayView<T>(const ArrayView< T > & array,**
  const ArrayView< size_type > & dims,
  const EStorageOrder storageOrder=DEFAULT_ORDER)

MDArrayRCP Constructors

- MDArrayRCP<T>(const ArrayView< T > & array,
  const ArrayView< size_type > & dims,
  EStorageOrder storageOrder=DEFAULT_ORDER)

- MDArrayRCP<T>(const ArrayView< size_type > & dims,
  const T & val=T(),
  EStorageOrder storageOrder=DEFAULT_ORDER)

- MDArrayRCP<T>(const ArrayView< size_type > & dims,
  EStorageOrder storageOrder)

Teuchos::MDArray methods

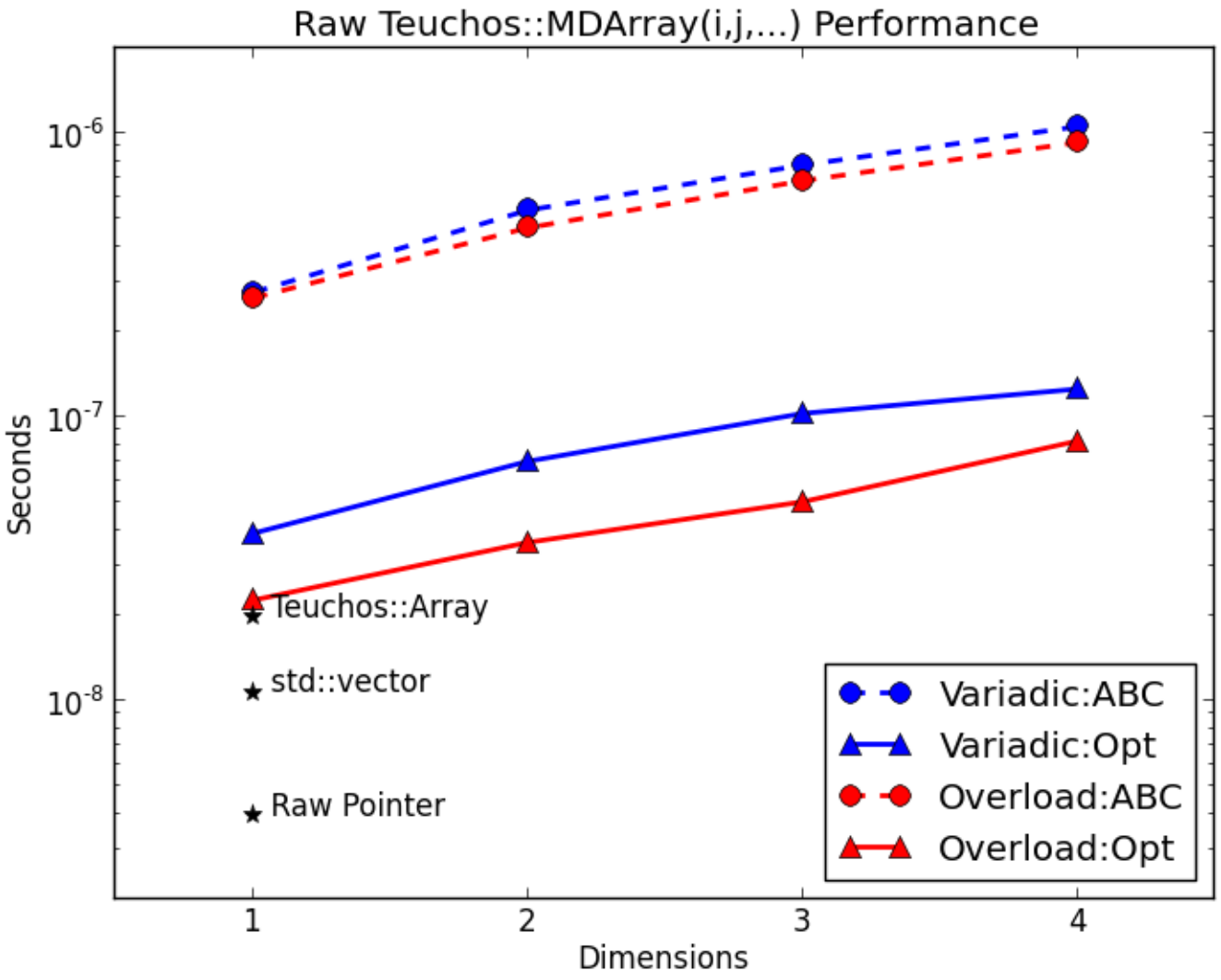
- **Attribute accessor methods:**
  - int num_dims() const;
  - const Array< size_type > & dimensions() const;
  - size_type dimension(int axis) const;
  - size_type size() const;
  - const Array< size_type > & strides() const;
  - const Array< T > & array() const;
  - const EStorageOrder storage_order() const;

- **std::vector-like methods:**
  - void assign(const T & value);
  - T & at(size_type i, ...);
  - const T & at(size_type i, ...) const;
  - size_type capacity() const;
  - void clear();
  - bool empty() const;
  - size_type max_size() const;
  - void resize(const ArrayView< size_type > & dims);
  - void swap(MDArray<T> & a);

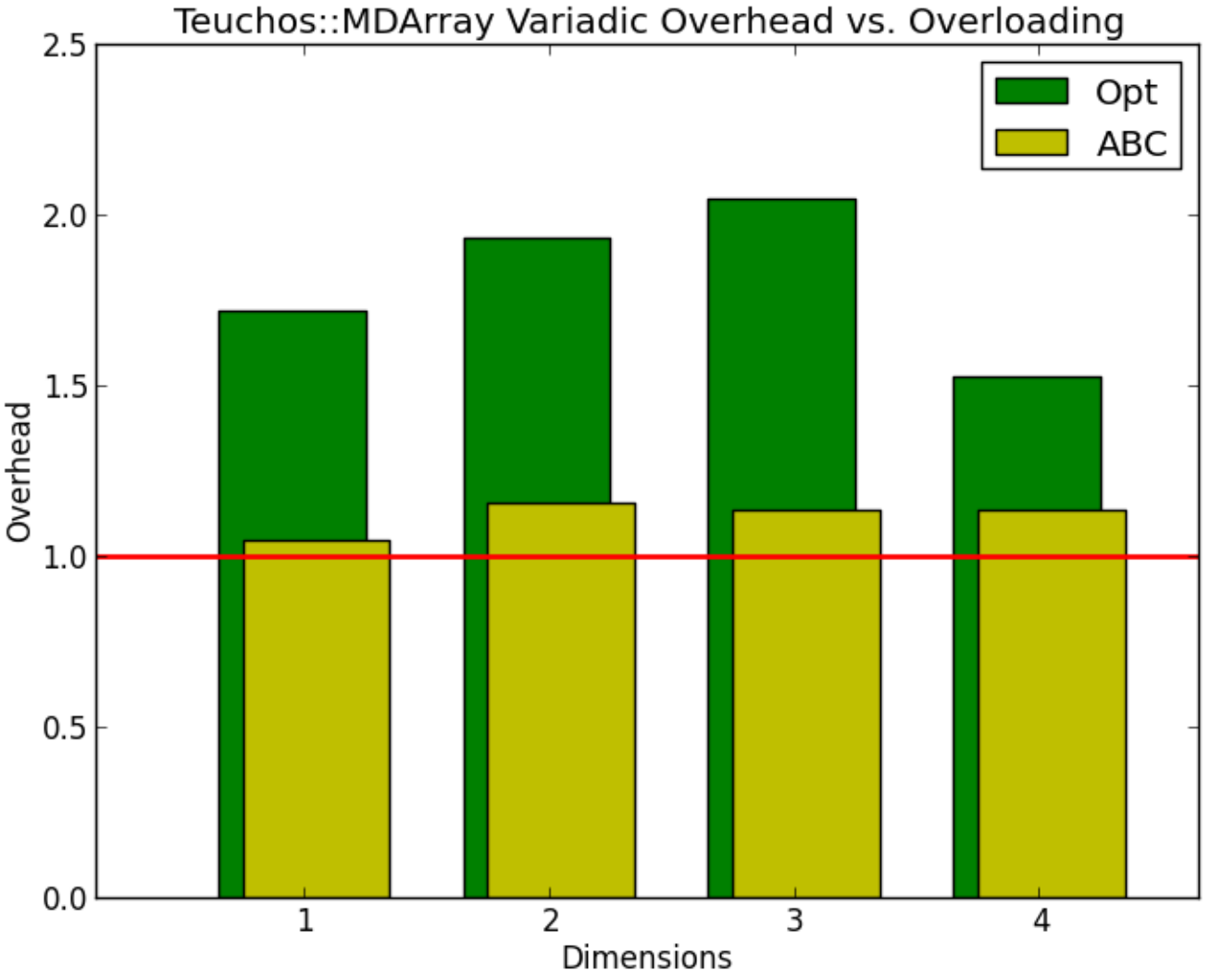
Indexing

- **C++ operator[] requires exactly one argument**
  - Could use single ArrayView argument, but that is clunky
- **To support i,j,k,... indexing, we will implement operator()**
- **Number of dimensions is dynamic … compiler is ignorant**
  - Enter the variadic argument
  - Classic use case: void fprint(char* fmt, ...)
    - Requires at least one argument before "..." typically to define remaining arguments
  - Our use case: T& operator()(size_type i, ...)
- **Downsides:**
  - Efficiency
  - Error checking
- **The alternative is to use overloading:**
  - T& operator()(size_type i);
  - T& operator()(size_type i, size_type j);
  - T& operator()(size_type i, size_type j, size_type k);
  - ...

Indexing Raw Performance



Variadic Performance Relative to Overloading



Obtaining sub-arrays: The Slice struct

- **Teuchos::Slice stores start and stop indexes, and step interval**
  - Inspired by Python slice
- **Differences from Teuchos::Range1D:**
  - Struct with public data members
  - Immutable
  - Upper bound is non-inclusive
  - Step interval
  - Negative values translate to indexing from upper bound
  - Default values
  - Slice bounds(Ordinal len) const; method
- **Example:**

```
// s is an “abstract” Slice, a is some container
Slice bounds = s.bounds(a.size());
for (Ordinal i=bounds.start; i != bounds.stop; i += bounds.step) {
    ...
}
```

Square bracket indexing

- **MDArrayView<T> operator[](size_type i);**
  - Returns MDArrayView with one fewer dimensions
  - Full chain returns MDArrayView of one dimension of length one … you probably want operator().

- **MDArrayView<T> operator[](Slice s);**
  - Returns MDArrayView with same number of dimensions
  - Chaining together requires an internal “next axis” data member
  - Mixing with ordinal version requires that operator to use “next axis” data member
  - **Official recommendation**: always chain together N square brackets for an N-dimensional array (you can use [Slice()])

```
// mda is a 2D MDArray
MDArrayView<double> view1 = mda[Slice(1,-1)];
MDArrayView<double> view2 = view1[0];  // Probably not what you expect
```

Concluding Remarks

- **High-level, multi-dimensional arrays in Teuchos**
  - Unit tests
  - Performance tests
- **Indexing**
  - Highly efficient operator() indexing
  - Powerful operator[] indexing
  - Utilizes Teuchos array bounds checking
- **Iterators**
  - Not implemented yet
  - Simple implementation can provide access w/o regard to dims
  - More sophisticated implementation could provide performance boosts like cache blocking
  - Most efficient multi-dimensional array systems include code generation techniques
- **Thanks to Ross Bartlett for review**