

# CCA Collective Component Specification

Jeeembo Kohl

Oak Ridge National Laboratory

December 2, 1999

Research supported by the Mathematics, Information and Computational Sciences Office, Office of Advanced Scientific Computing Research, U.S. Department of Energy, under contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.

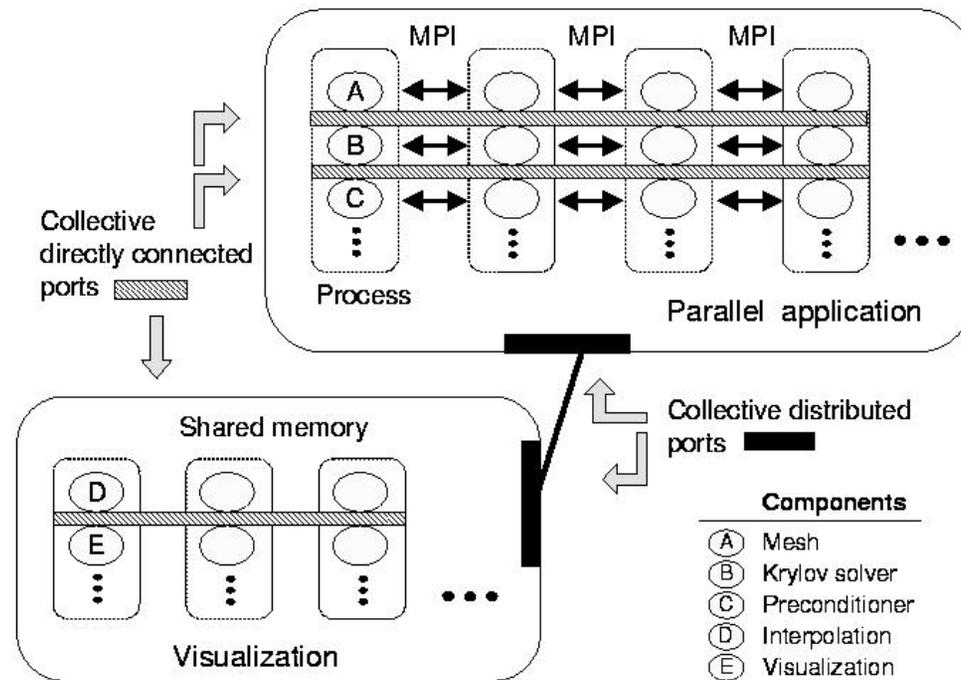
Kohl/1999-1

# Collective Components

- For Parallel Components, Distributed Data
- Extract & Assemble Subsets of Data Fields
  - ⇒ Online Visualization, Model Coupling
- Need Map of Data Layout / Decomposition
  - ⇒ Automate Data Collection (Hide Parallel Details)
  - ⇒ Support Interpolation or Re-Distribution of Data

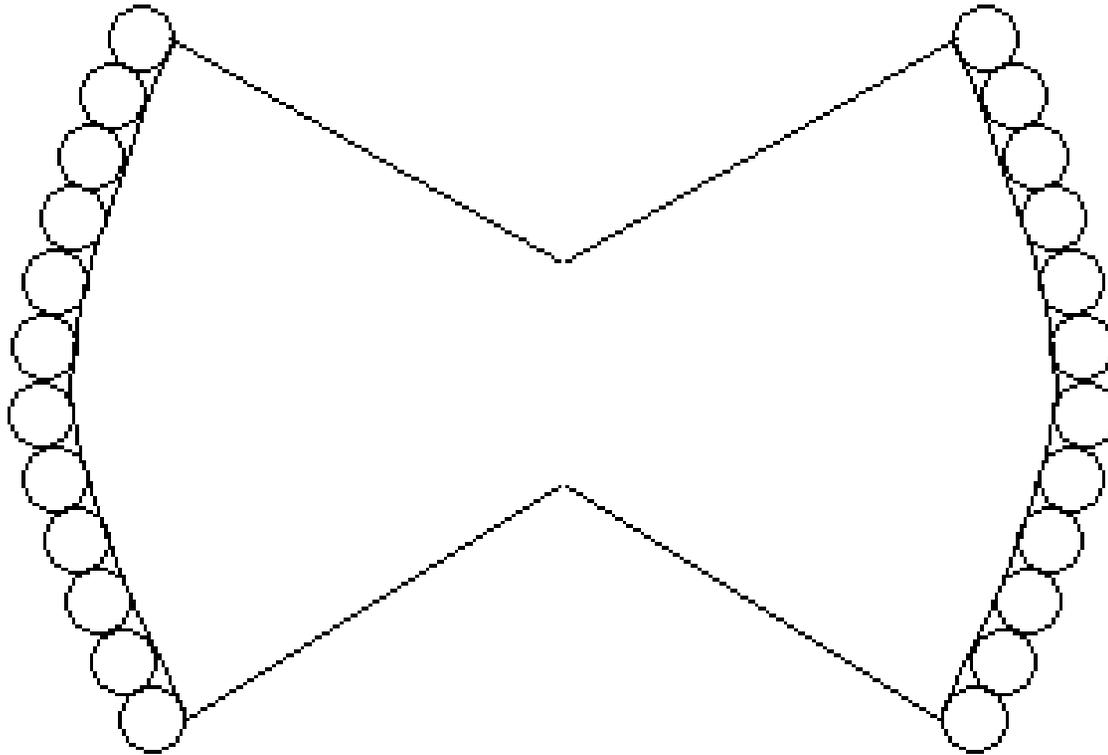
# Arbitrary Data Distributions

- Mixtures:  $N \times N$ ,  $1 \times N$ ,  $N \times M$ , Dynamic  
⇒ Varying Models: Distributed / Shared Memory



# Two Views of a Collective Component

- Front-End ~ Serial Data Via a Single Port
- Back-End ~ Set of Collective (Provides) Ports



# Who Invokes Who?

(or who put the bomb in the bomb-ba-bomb-ba-ba...?)

- Parallel Components “Attach” to CC:
  - ⇒ Each Task ~ Own “Collective (Provides) Port” on CC
  - ⇒ Task Invokes CP Methods to Define Its Data Fields
- Collective Component Uses Task Info
  - ⇒ Set of Collective Ports “Covers” Parallel Component
  - ⇒ Collect Local Data when Task Says “O.K.”
  - ⇒ Translate Data  $1 \times N$ ,  $N \times N$ ,  $M \times N$ ...

# Collective Port Interface

## Spatial & Temporal Mappings

- ***Decomposition*** ~ Context in Global Array
- ***ProcArray*** ~ Shape and Organization
- ***Time Map*** ~ Simulated Time of Data
- ***DataField*** ~ Base Data Field Info
- ***Local*** ~ Allocation & Layout of Local Data
- ***CollectivePort*** ~ Where to Stick the Data...
- ***Interpolation*** ~ Translating Spatial / Temporal

# Decomposition Specification

⌘ Global Context of Local Data Elements ⌘

- Decomposition Types for Each Axis
  - ⇒ With Detail Information Per Axis
- Global Array Size, Shape and Coordinates

# DecompInfo

```
package gov.CCA.Collective {  
    // Information for the Decomposition of a Single Axis  
    interface DecompInfo {  
        void setProperty( in string name, in string value );  
        // e.g. “Block Size” / “10”  
        void addBounds( in integer numBounds, // # of explicit region bounds  
            in array<integer,numBounds> lowerBounds,  
            in array<integer,numBounds> upperBounds );  
    }  
    ...  
}
```

# Decomp

```
package gov.CCA.Collective {  
    ...  
    // Data Decomposition Template for Data Fields  
    interface Decomp {  
        void setDecompBounds(  
            in integer dataDimGlobal, // dim of global array  
            in array<integer,dataDimGlobal> globalLowerBounds,  
            in array<integer,dataDimGlobal> globalUpperBounds );  
        void setDecompAxis(  
            in integer axisIndex, // index of axis x=0, y=1, z=2, etc...  
            in string decompType, // e.g. "Block", "Cyclic"...  
            in DecompInfo info ); // info for decompType...  
    }  
    ...  
}
```

# ProcArray

```
package gov.CCA.Collective {  
    ...  
    // Processor/Process Topology/Organization  
    interface ProcArray {  
        void setProcShape( in integer procDim, // dim of processor array  
                           in array<integer,procDim> procShape ); // size of each axis  
        void setProcAddress( in integer procDim,  
                             in array<integer,procDim> procAddress );  
                                // address of instance in array  
    }  
    ...  
}
```

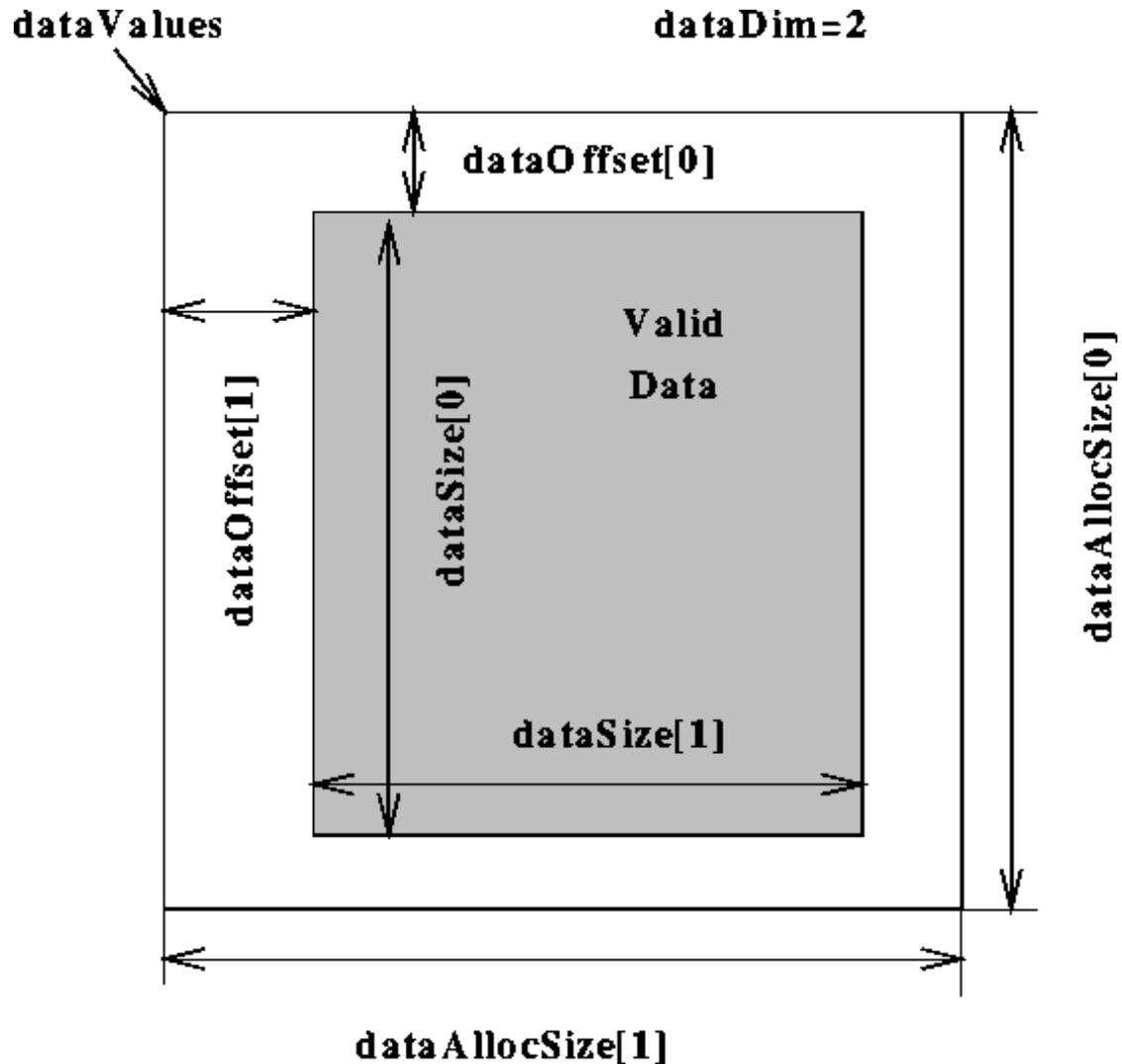
# Time Map

```
package gov.CCA.Collective {  
    ...  
    // Time Values for Time-Based Simulations  
    interface Time {  
        void setTime( in float timeValue, // set time value  
                     in string timeUnitScale, // e.g. “seconds”, “minutes”...  
                     in string timeUnitPrefix ); // e.g. “pico”, “giga”...  
        void incrTime( in float timeValue, // increment time value  
                     in string timeUnitScale,  
                     in string timeUnitPrefix );  
    }  
    ...  
}
```

# DataField

```
package gov.CCA.Collective {  
    ...  
    // Data Field Stuff – Tie It All Together!  
    interface DataField {  
        void setField( in string name, // logical name of array  
                       in Data data ); // reference to actual data...  
        void setSize( in integer dataDim, // dimension of data array  
                     in array<integer,dataDim> dataSize ); // cardinality of each axis  
        void setDecomp( Decomp decomp ); // data distribution for data field  
        void setProcArray( ProcArray parray ); // resources holding data field  
        ...  
    }  
    ...  
}
```

# Local Allocation



# Local Specification

```
package gov.CCA.Collective {  
    ...  
    // Data Field Stuff – Tie It All Together!  
    interface DataField {  
        ...  
        void setOffset( in integer dataDim,  
                        in array<integer,dataDim> dataOffset ); // valid data each axis  
        void setValidData( in integer dataDim,  
                            in array<integer,dataDim> dataValidSize ); // valid size each axis  
        ...  
    }  
    ...  
}
```

# More DataField Cruft...

```
package gov.CCA.Collective {  
    ...  
    // Data Field Stuff – Tie It All Together!  
    interface DataField {  
        ...  
        void setUnits( in string unitScale, // scale of units, e.g. “meters”, “kg”...  
                       in string unitPrefix ); // unit prefix, e.g. “pico”, “deca”, “tera”...  
        void setGrid( in string gridType ); // e.g. “rectangular”, “spectral”...  
        void setTime( in Time time ); // set current simulation time for data field  
        void setPeriod( in Time period ); // simulated time per iteration  
        void setReady( // data field is ready for collection  
                       in boolean incrTime ); // increment field time (by period) or not?  
    }  
    ...  
}
```

# And Now... The Collective Port!

```
package gov.CCA.Collective {  
    ...  
    interface CollectivePort extends gov.CCA.Port {  
        void addDataField( in DataField field ); // hook up this data field  
    }  
}
```

# Interpolation Hooks

```
package gov.CCA.Collective {  
    ...  
    interface Interpolate {  
        void setInterpTemporal( in function spankTime() );  
        void setInterpSpatial( in function spankSpace() );  
    }  
}
```

**Or Specify as Interpolation Components Instead?**

# Conclusion

- Collective Component Interface
  - ⇒ Parallel Components “Attach”
    - Each Task Has Its Own Collective Port
    - Task Invokes C.P. Methods to Define Data Fields
  - ⇒ Collective Component Handles Data Munging
- Interpolation Component Next?