MUX and MxN: A Revelation…

(a.k.a. “Collective” is Dead? :-o)

James Arthur Kohl

David E. Bernholdt

Oak Ridge National Laboratory

Thursday, December 7, 2000

(a date which will live in infamy…?)

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.

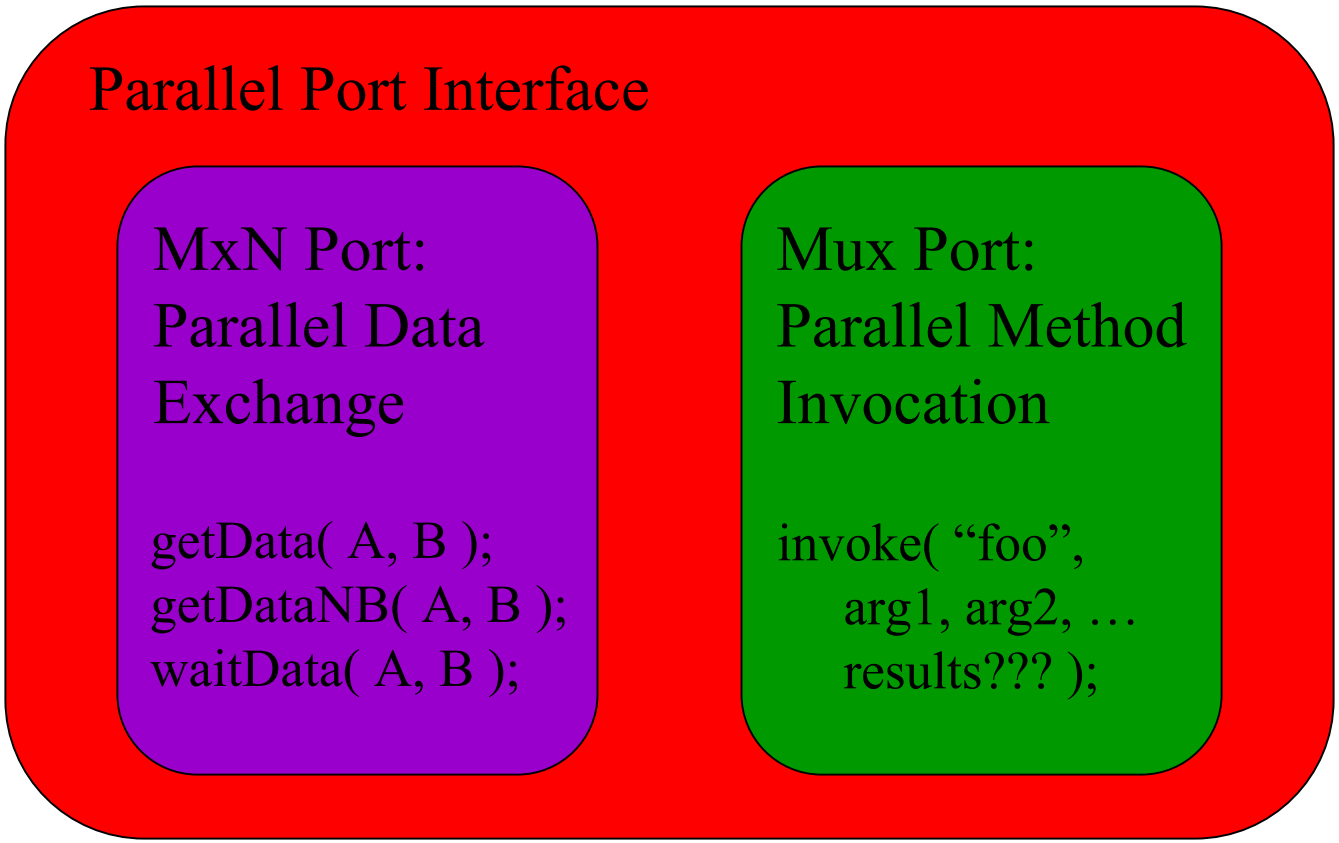
The Often Overlooked Duality…

- What is "Collective" Really?
  - ⇒ Dealing with Parallel Components
  - ⇒ Coupling Disparate Parallel Models
  - ⇒ Interfacing to Serial Components
- Two Sides:
  - ⇒ Parallel Data Exchange
  - ⇒ Parallel Method Invocation
- These are ***Distinct*** Issues
  - ⇒ Need to be Handled Separately

The Big Picture

- Interfacing Parallel Components:
  - ⇒ Exchanging and Extracting Data
    - → MxN Data Translation
    - → Other Convolutions: Grid, Temporal, Units…
      - * Outside of Parallel Paradigm, But Required & Related…
  - ⇒ Invoking Methods on Parallel Components
    - → Selective / Multiple Invocations
    - → Passing Arguments / Returning Value(s)…
- Common Issues
  - ⇒ Synchronization & Scheduling

Parallel Port Interface



Integrated MxN Port Interface

(No More coupleDataFields( )…)

```
class MxN : Port {
    virtual void registerData( Data A );
    virtual void unregisterData( Data A );
    // (Data Properties: synch)
    virtual int getData( Data A, Data B );
    virtual int getDataNonBlocking( A, B );
    virtual void waitData( A, B );
    virtual void releaseData( A, B );
    // Port Properties: freq, init_synch, run_synch…
}
```

MxN Port Properties

- Freq ~ Data Exchange Frequency
  - ⇒ 0: one-shot, send/receive model (PAWS)
  - ⇒ > 0: persistent conn, loose synch (CUMULVS)
- Init_Synch ~ Override Initial Default Synch
  - ⇒ Parallel Data Object Sets "synch" Required
- Run_Synch ~ Optimize Run-Time Synch
  - ⇒ Maintain Synch with All Parallel Tasks, versus
  - ⇒ Just Maintain Synch with Relevant Tasks
    - → Trade Off (Re)Attachment vs. Persistent Overhead…

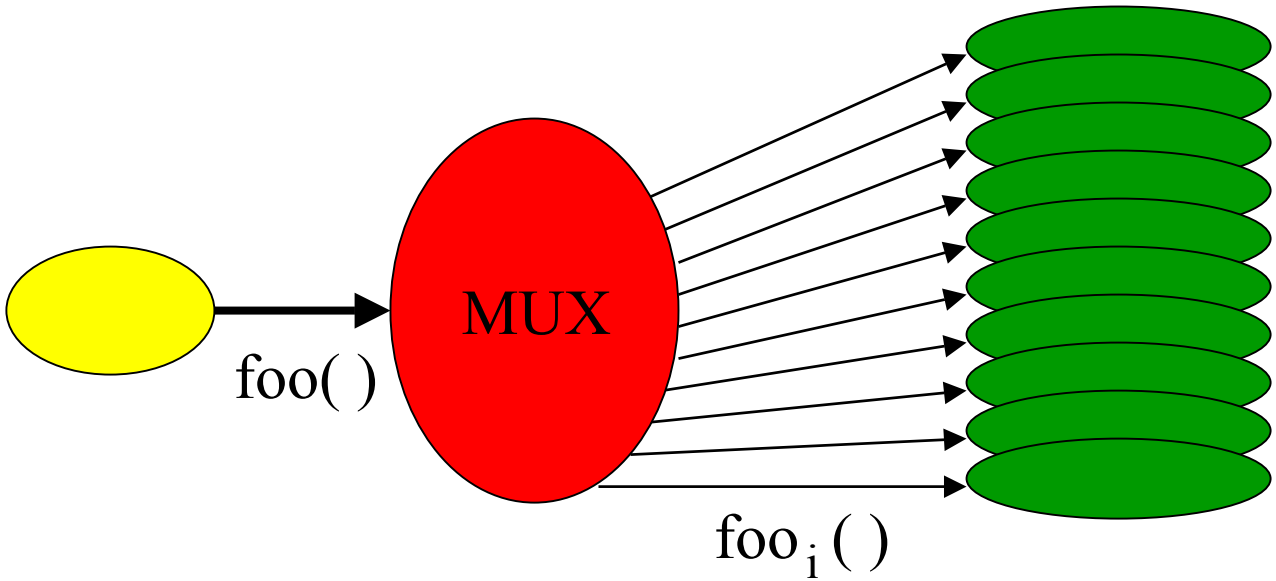
What About “DataFields”?

- Leave it to the Data Object Interfaces…
- MxN can use generic info from:
  - ⇒ Distributed Data Decompositions
  - ⇒ Processor Topology & Allocation
  - ⇒ Mesh / Grid Type
  - ⇒ Data Field Times & Periods
  - ⇒ setReady( )

Parallel Invocation Multiplexer (MUX)

- Coordinates Invocations of Methods on Parallel Components

  $\Rightarrow$ Intermediary Component Schedules Calls



Possible MUX Interfaces…

- invokeAll(“foo”, args, …, results???);
  - ⇒ Call Method on Each Parallel Instance
  - ⇒ Pass Scalar or Parallel (Indexed) Arguments?
  - ⇒ Vector of Results Collected & Returned?
- invokeOne(“foo”, …);
  - ⇒ Select One Instance to Execute Method
  - ⇒ Use Port Properties for Selection Policy?

MUX Synchronization
(invokeAll( ) Only)

- Does the Parallel Invocation Cooperate?
  - ⇒ Require Full / Loose Synchronization Among Parallel Instances?
  - ⇒ Communication Among Instances in “foo( )” ?
  - ⇒ Data Consistency Across Invocations?
- Invocation Order / Servicing Multiple Requests
  - ⇒ Interleave Different Invocations? Policy…
- Results Reductions
  - ⇒ Produce Single Result from Many ~ Voting?

One More Can of Worms…!

- Explicit versus Implicit Parallel Port Handling
  - ⇒ Both Approaches Desirable
  - ⇒ Both Interfaces Utilize the Same Underlying Implementation
- Several Approaches:
  - ⇒ SIDL / Babel (next up – Gary ☺)
  - ⇒ Pluggable Framework Services…

Pluggable Framework Services

- Instantiate Alternate “Handlers” at Run Time
- Implicit MxN / MUX:
  - ⇒ Need “Method Invocation Handler” Interface
  - ⇒ Intercept Invocations on Ports
    - → Insert Automatic MxN Data Extraction for Arguments
  - ⇒ Event-Based or Not?
  - ⇒ Services→addHandler(“Invocation”, MxN→handle( ));
- Likely Other Uses / Pluggable Services
  - ⇒ Improves Portability / Eases Framework Dvlpmt

Summary

- Separate Concerns:
  - $\Rightarrow$ MxN Interface ~ Parallel Data Exchange
  - $\Rightarrow$ MUX Interface ~ Parallel Method Invocation
- Simplify Interfaces → Rely on Data Objects
- Pluggable Framework Services?