Software Tools for Parallel Coupled Simulations

Alan Sussman

UNIVERSITY OF MARYLAND

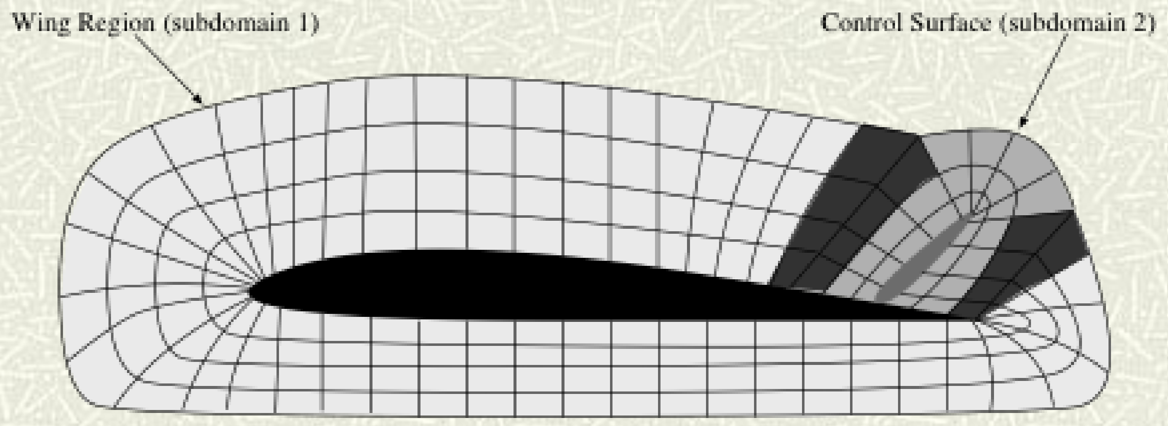
Department of Computer Science &
Institute for Advanced Computer Studies

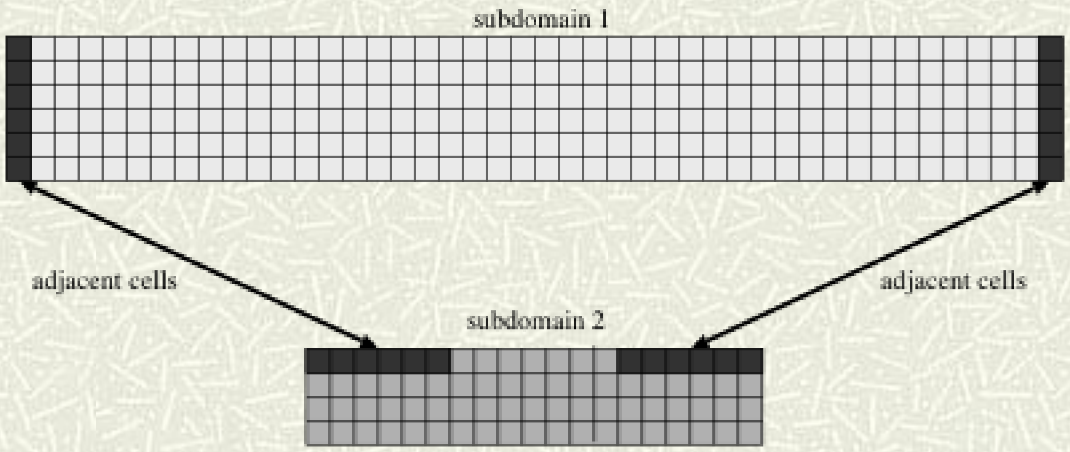
http://www.cs.umd.edu/projects/hpsl/chaos/ResearchAreas/ic/

Ancient History

- Block structured CFD applications
  - Multi-block (Irregularly Coupled Regular Meshes)
  - Multigrid
- TLNS3D CFD application
  - Vatsa et. al at NASA Langley
- How to parallelize effectively, on distributed memory parallel machine?

Multiblock Grid





Solution: Multiblock Parti

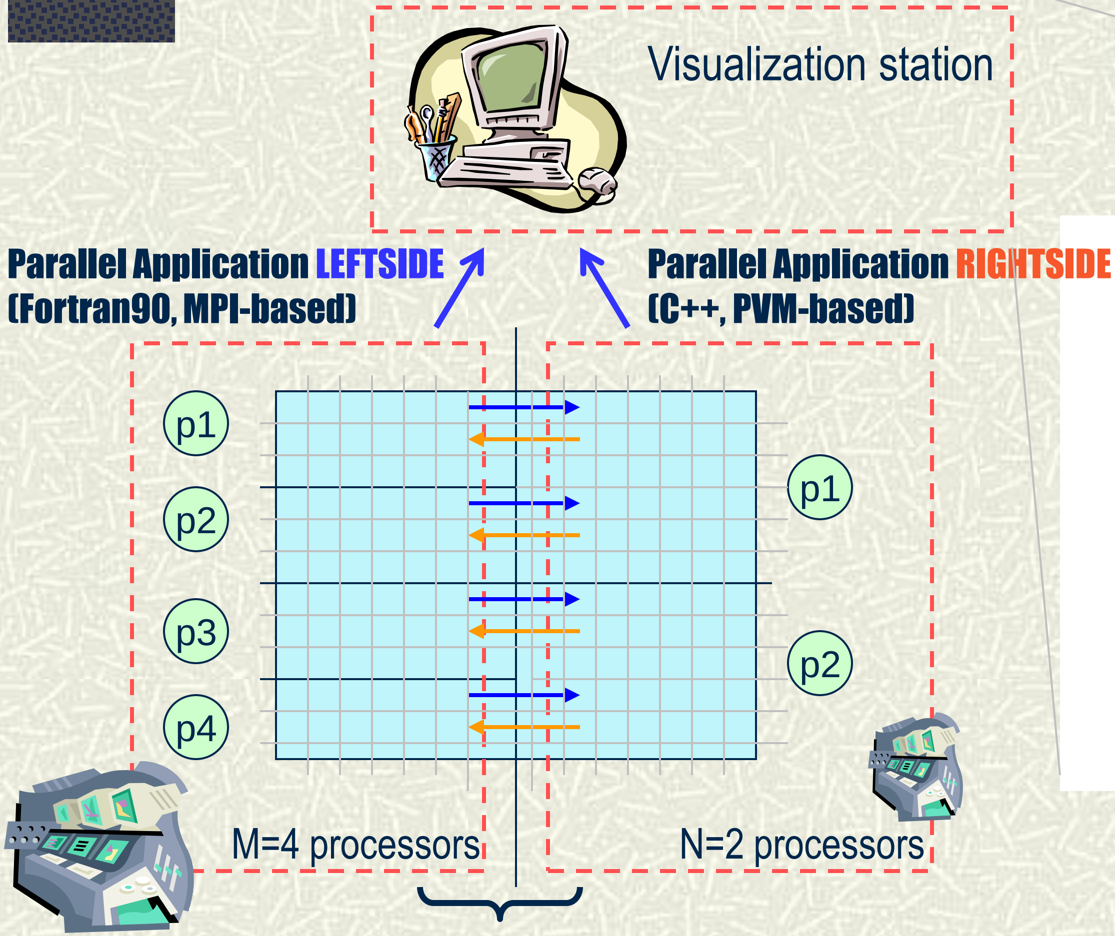
- Capabilities:
  - Runtime data distributions
  - Distribute individual block over parts of processor space
  - Fill in overlap/ghost cells, for partitioned blocks
  - Regular section moves for communication across blocks
  - Enables reuse of *communication schedules*

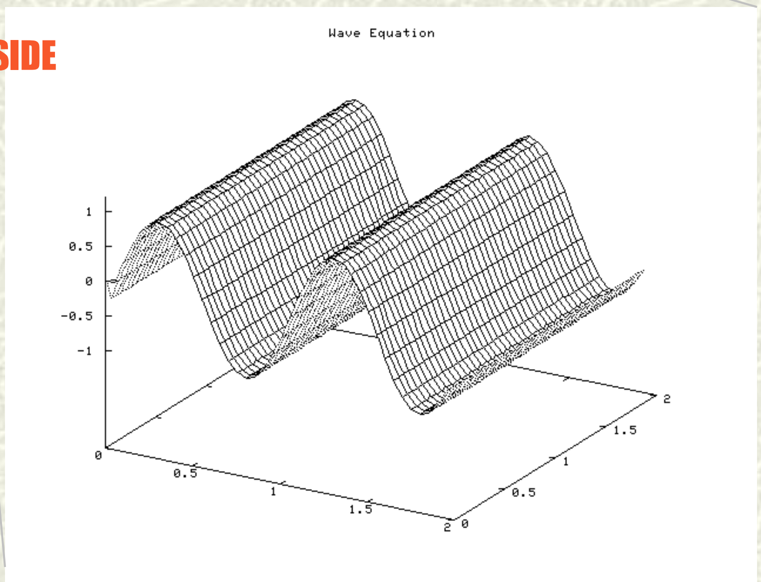
Multiblock Parti

- Shown to provide excellent performance, and scaled to large machine configurations (at the time)
- Other libraries with similar functionality:
  - KeLP (UCSD, Baden)
  - Global Arrays (DOE PNNL)
    - still supported and widely used
- Multiblock Parti used in LLNL P++ array class library
  - for AMR and other distributed array codes

InterComm

A Simple Example (MxN coupling)





2-D Wave Equation

InterComm: Data exchange at the borders (transfer and control)

Coupling Parallel Programs via InterComm

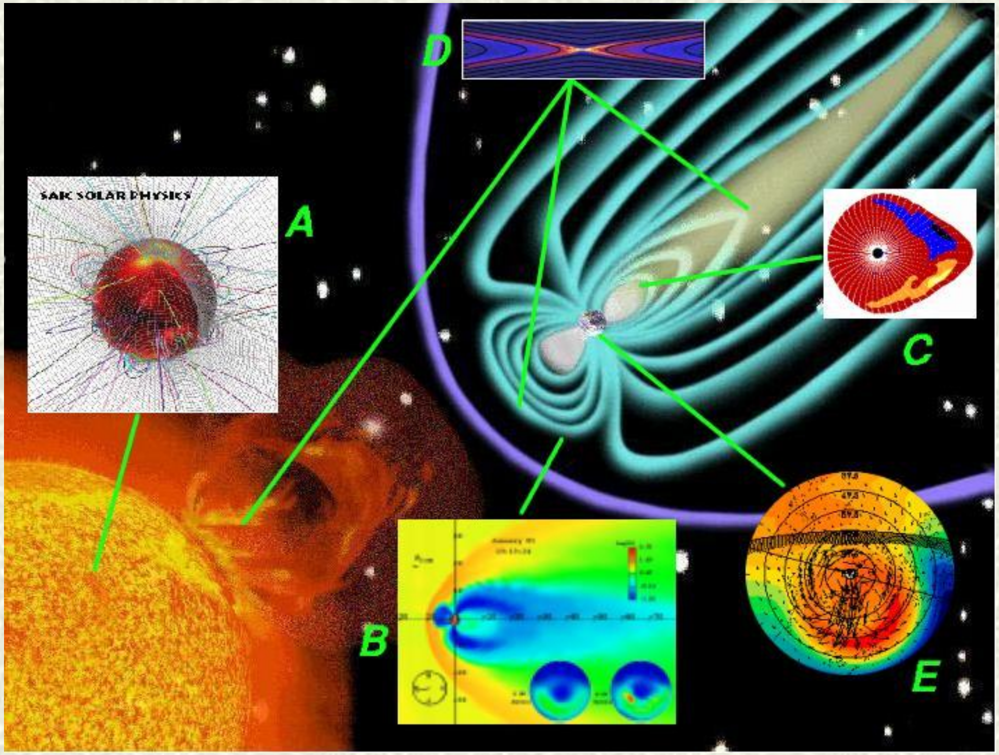
- Introduction
  - Problem Definition (the MxN problem)
  - InterComm in a nutshell
- Design Goals
  - Data Transfer Infrastructure
  - Control Infrastructure
  - Deploying on available computational resources
- Current Status

The Problem

- Coupling *codes*, not models
- Codes written in different languages
  - Fortran (77, 95), C, C++/P++, …
- Both parallel (shared or distributed memory) and sequential
- Codes may be run on same, or different resources
  - One or more parallel machines or clusters (the **Grid**)

Space Weather Prediction

- Major driving application:
  - Production of an ever-improving series of comprehensive scientific models of the Solar Terrestrial environment
  - Codes model both large scale and microscale structures and dynamics of the Sun-Earth system



What is InterComm?

- A programming environment and runtime library
  - For performing efficient, direct data transfers between data structures (*multi-dimensional arrays*) in different programs/components
  - For controlling *when* data transfers occur
  - For deploying multiple coupled programs in a Grid environment – won't talk about this

Data Transfers in InterComm

- Interact with data parallel (SPMD) code used in separate programs (including MPI)
- Exchange data between separate (sequential or parallel) programs, running on different resources (parallel machines or clusters)
- Some people refer to this as the **MxN** problem

InterComm Goals

- One main goal is minimal modification to existing programs
  - In scientific computing: plenty of legacy code
  - Computational scientists want to solve *their* problem, not worry about plumbing
- Other main goal is low overhead and efficient data transfers
  - Low overhead in *planning* the data transfers
  - Efficient data transfers via customized all-to-all message passing between source and destination processes

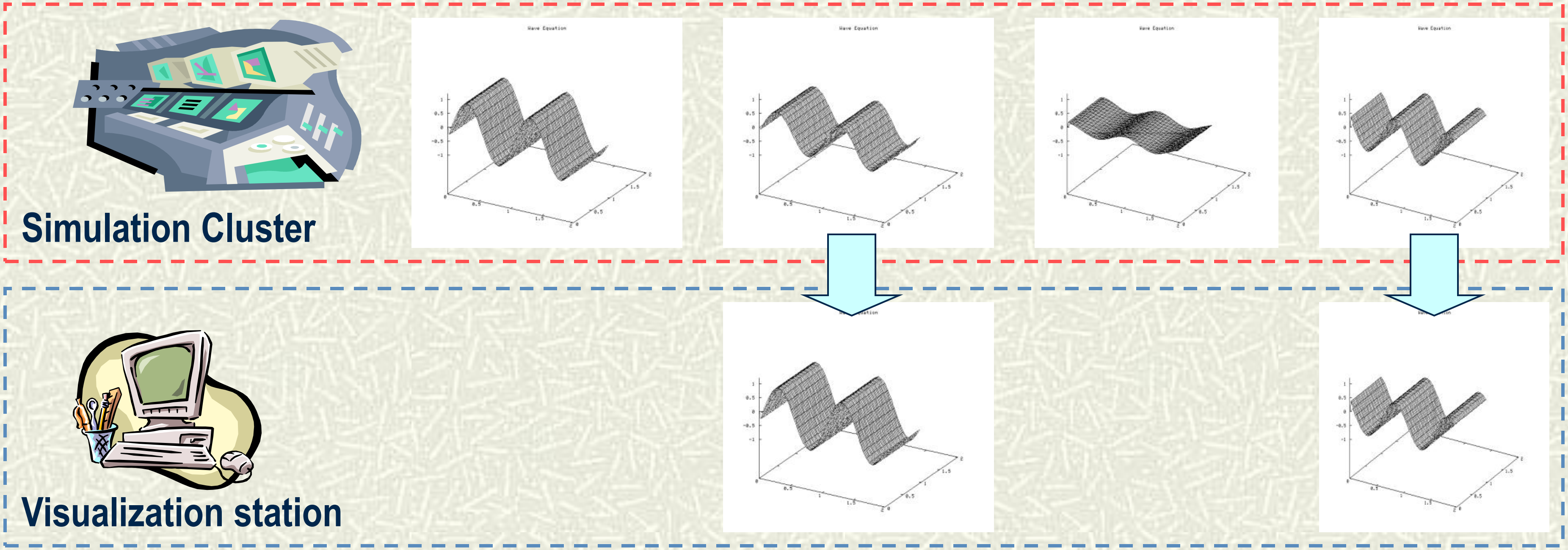
Coupling OUTSIDE components

- Separate coupling information from the participating components
  - Maintainability – Components can be developed/upgraded individually
  - Flexibility – Change participants/components easily
  - Functionality – Support variable-sized time interval numerical algorithms or visualizations
- Matching information is specified separately by application integrator
- Runtime match via simulation time stamps

Controlling Data Transfers

- A flexible method for specifying *when* data should be moved
  - Based on matching export and import calls in different programs via timestamps
  - Transfer decisions take place based on a separate coordination specification
    - Coordination specification can also be used to deploy model codes and grid/mesh translation/interpolation routines
    - specify what codes to run and where to run them)
    - called an XML job description (XJD) file

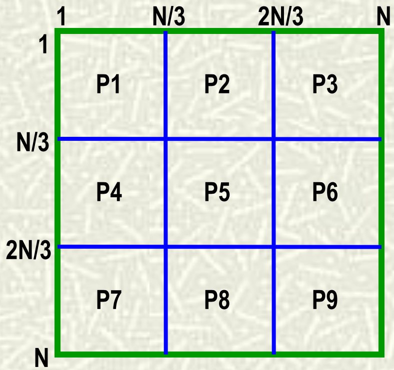
Example



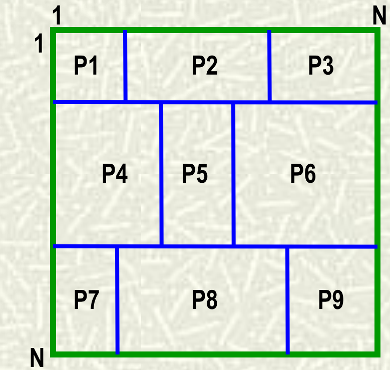
- Simulation exports every time step, visualization imports every 2nd time step

Issues in Coupling Codes

- To enable a program to be coupled to others, we need to:
  - Describe data distribution across processes in each parallel program
    - Build a *data descriptor*
  - Describe data to be moved (imported or exported)
    - Build set of *regions*
  - Build a communication schedule
    - What data needs to go where
  - Move the data
    - Transmit the data to proper locations



Regular Block



Generalized Block

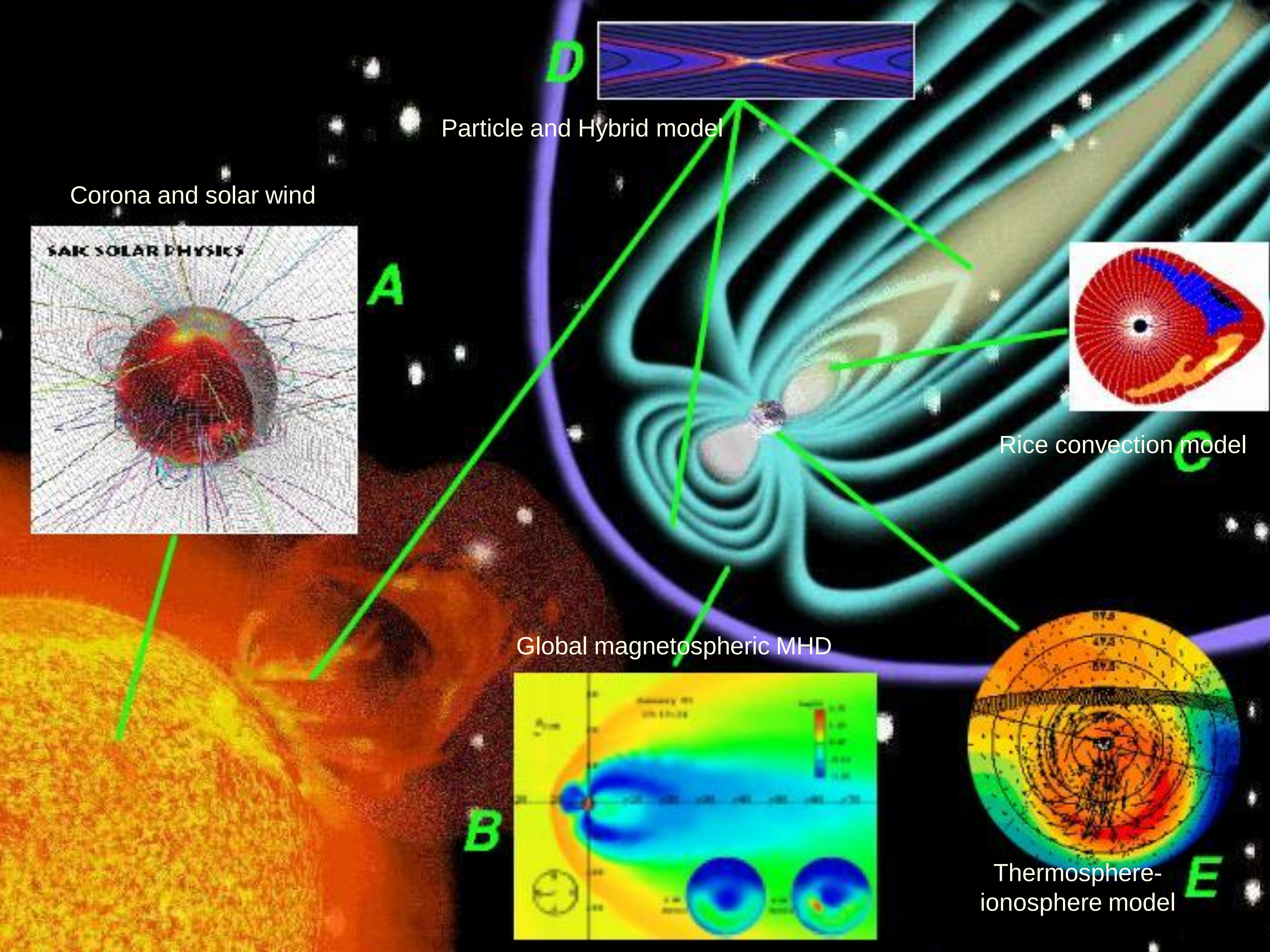
Plumbing

- Bindings for C, C++/P++, Fortran77, Fortran95
- *External* message passing and program interconnection via MPI or PVM
- Each model/program can do whatever it wants internally (MPI, OpenMP, pthreads, sockets, …) – and start up by whatever mechanism it wants (in XJD file)

Current status

- http://www.cs.umd.edu/projects/hpsl/chaos/ResearchAreas/ic/
- First InterComm 2.0 release in 2009
  - Dynamic timestamp matching supported
  - requires pthreads support from OS
  - Supported on Linux clusters, NCAR bluefire (IBM Power7, with LSF scheduler), Cray XT, other high-end machines
- Integrated with ESMF (Earth System Modeling Framework)
  - wrap ESMF objects for communication via InterComm
  - Part of ESMF code contributed code base

END OF TALK



Particle and Hybrid model
D
Corona and solar wind
SAIC SOLAR PHYSICS
A
Rice convection model
C
Global magnetospheric MHD
B
Thermosphere-
ionosphere model
E

Data Transfer

- It all starts with the Data Descriptor
  - Information about how the data in each program is distributed across the processes
  - Usually supplied by the program developer
- Compact or Non-Compact descriptors
  - Regular Blocks: collection of offsets and sizes (one per block)
  - Irregular Distributions: enumeration of elements (one per element)
- Performance issue is that different algorithms perform best for different combinations of source/destination descriptors and local vs. wide area network connections

Separate codes from matching

Exporter Ap0

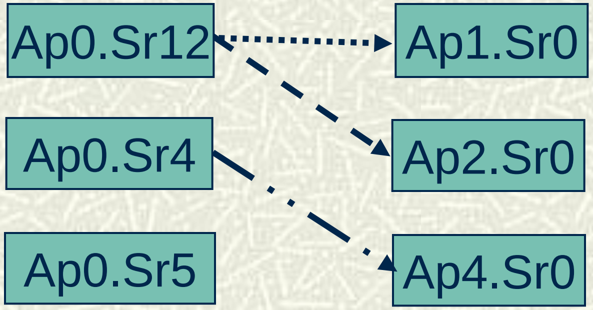
```
define region Sr12
define region Sr4
define region Sr5
...
Do t = 1, N, Step0
    ... // computation
    export(Sr12,t)
    export(Sr4,t)
    export(Sr5,t)
EndDo
```

Importer Ap1

```
define region Sr0
...
Do t = 1, M, Step1
    import(Sr0,t)
    ... // computation
EndDo
```

Configuration file

```
#
Ap0 cluster0 /bin/Ap0 2 ...
Ap1 cluster1 /bin/Ap1 4 ...
Ap2 cluster2 /bin/Ap2 16 ...
Ap4 cluster4 /bin/Ap4 4
#
Ap0.Sr12  Ap1.Sr0  REGL  0.05
Ap0.Sr12  Ap2.Sr0  REGU  0.1
Ap0.Sr4   Ap4.Sr0  REG   1.0
#
```

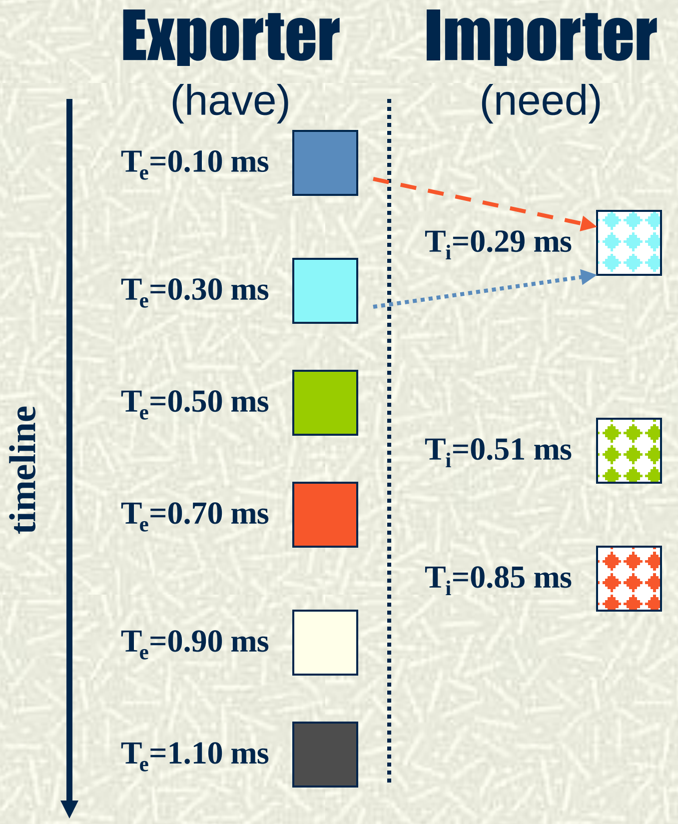


Approximate Matching

- Exporter Ap0 produces a sequence of data object ***A*** at simulation times 1.1, 1.2, 1.5, and 1.9
  - ***A***@1.1, ***A***@1.2, ***A***@1.5, ***A***@1.9
- Importer Ap1 requests the same data object ***A*** at time 1.3
  - ***A***@1.3
- Is there a match for ***A***@1.3? If **Yes**, which one and why?

Controlling Data Transfers

- *Import* and *Export* operations are time-stamped (T_i and T_e)
- Issues in designing *Decision Functions*
  - Matching Policy
    - Does the import timestamp *match* any of the exported timestamps, subject to a particular policy?
  - Precision
    - Which of the exported data most closely matches what is requested to be imported?
- Decision functions directly affect InterComm buffering decisions



Deploying Components

- Infrastructure for deploying programs and managing interactions between them
  - Starting each of the models on the desired Grid resources
  - Connecting the models together via the InterComm framework
  - Models communicate via the import and export calls

Motivation

- Developer has to deal with …
  - Multiple logons
  - Manual resource discovery and allocation
  - Application run-time requirements
- Process for launching complex applications with multiple components is
  - Repetitive
  - Time-consuming
  - Error-prone

Deploying Components

- A single environment for running coupled applications in the high performance, distributed, heterogeneous Grid environment
- We must provide:
  - Resource discovery: Find resources that can run the job, and automate how model code finds the other model codes that it should be coupled to
  - Resource Allocation: Schedule the jobs to run on the resources – without you dealing with each one directly
  - Application Execution: start every component appropriately and monitor their execution
- Built on top of basic Web and Grid services (XML, SOAP, Globus, PBS, Loadleveler, LSF, etc.)

What else is out there?

- CCA MxN Working Group
- Parallel Application Work Space (PAWS) [Beckman et al., 1998]
- Collaborative User Migration, User Library for Visualization and Steering (CUMULVS) [Geist et al., 1997]
- Model Coupling Toolkit (MCT) [Larson et al., 2001]
- Earth System Modeling Framework (ESMF)
- Space Weather Modeling Framework (SWMF)
- Roccom [Jiao et al., 2003]
- Overture [Brown et al., 1997]
- Cactus [Allen et al., 1999]

Summary and Ongoing Work

- InterComm: a comprehensive high-performance framework for coupling parallel scientific codes
- Plumbing for high performance data transfers is fully functional and released, deployment services released, control functions released
- Continuing to working with our customer base to modify their codes and couple their *models*