Journal of Atmospheric and Solar-Terrestrial Physics 66 (2004) 1469–1479

Journal of ATMOSPHERIC AND SOLAR-TERRESTRIAL PHYSICS

www.elsevier.com/locate/jastp

The CISM code coupling strategy

C.C. Goodrich[a,*], A.L. Sussman[b], J.G. Lyon[a,c], M.A. Shay[d], P.A. Cassak[d]

[a]Center for Integrated Space Weather Modeling, Boston University, 725 Commonwealth Ave, Boston, MA 02215, USA
[b]Department of Computer Science, University of Maryland, College Park, MD 20742, USA
[c]Department of Physics, Dartmouth College, Hanover, NH 03755, USA
[d]Institute for Research in Electronics and Applied Physics, University of Maryland, College Park, MD 20742, USA

Received 13 February 2004; received in revised form 7 April 2004; accepted 8 April 2004

Abstract

The success of the Center for Integrated Space Weather Modeling (CISM) depends on the production of an ever-improving series of comprehensive scientific models describing the Solar Terrestrial environment from the solar surface to the upper atmosphere of earth. We describe here our strategy for coupling the codes we have selected as the basis for these models, which include core global codes which address the corona, heliosphere, the earth's magnetosphere, and ionosphere, and codes which model important local processes such as magnetic reconnection. Coupling these codes requires four separate functions: efficient transmission of information among codes, interpolation of grid quantities, translation of physical variables between codes with differing physical models, and control mechanisms to synchronize the interaction of codes. The characteristics of these codes dictate an approach involving loosely coupled groups of independently running programs. We have selected two existing software packages, InterComm and Overture, to provide the basis of our coupling framework. By combining the strengths of these packages, we obtain the benefits of simplified coding of translation routines and inter-grid communication between distinct codes with minimal code modification. The NASA Living With a Star program shares both the scientific goals and code coupling challenges of CISM, and is equally involved in the coupling strategy and development we present.
© 2004 Elsevier Ltd. All rights reserved.

Keywords: Code coupling; Solar-terrestrial physics

1. Introduction

The central focus of the Center for Integrated Space Weather Modeling (CISM) is the production of an ever-improving series of comprehensive scientific models of the solar terrestrial environment. We plan to produce new model versions every 2 years, using increasingly sophisticated coupling technology and incorporating a greater number of component models, thereby improving and expanding the physics of the space environment the model contains. Each new version will be evaluated independently within CISM for a period of at least a year before being released to the scientific and operational communities and the public through the efforts of the CISM Knowledge Transfer and Education teams.

In order to achieve this goal, we need to develop software through which the codes can be coupled together efficiently and with a maximum amount of flexibility for adding new physics and new simulation models. As described previously in Luhmann et al. (2004), CISM has chosen to use existing codes that model accurately either individual regions of the

*Corresponding author. Tel.: +1-617-358-3244; fax: +1-617-358-3242.
E-mail address: ccg@bu.edu (C.C. Goodrich).

solar-terrestrial environment, such as the corona or magnetosphere, or particular physical processes, such as magnetic reconnection or ion/electron acceleration. Our reliance on existing codes necessitates that the software we use require only minimal code modification. At the same time, the software must support extensive numeric (grid) interpolation both at boundaries, (e.g., the coronal and heliospheric codes), or in overlapping volumes (e.g., the magnetospheric and the radiation belt codes), as well as the translation of physical quantities as necessary among the codes. These constraints require that our coupling technology have four separable functions; (1) efficient transmission of information among codes, (2) interpolation of grid quantities, (3) translation of physical variables between codes with differing physical models, and (4) control mechanisms to synchronize the execution and interaction of the codes.

The NASA Living With a Star (LWS) program shares with CISM the scientific goal of understanding the interaction of the Sun with the Earth and the near Earth space environment. While a primary focus of LWS is the deployment of a fleet of spacecraft to probe the solar, near earth (radiation belts), and ionosphere conditions, the LWS science architecture team (http://lws.gsfc.nasa.gov/lws_news.htm) has pointed out that modeling efforts are crucial to the success of the LWS mission. Numerical and theoretical modeling, using the same or equivalent codes as CISM, will be critical to provide a context and framework for understanding the observations, just as the observations will be critical for improving the models and codes. Thus the LWS program faces the same challenges in code coupling as CISM.

We describe here the coupling strategy we are developing for CISM and LWS. Our intent is to use the results of the current work in code coupling technology in computer and computational science as much as possible, which we summarize briefly in the next section. We describe the technical challenges we face in coupling the major codes in solar terrestrial physics. Finally we describe the specific framework model we are developing, and present the preliminary results we have obtained with it.

2. Code coupling tools

The problem of coupling of independently developed programs has been a focus of computer and computational science research in recent years, with particular attention given to the management of parallel data structures inherent in parallel applications. We have surveyed this work in developing our coupling approach. While some provide similar methods to distribute parallel data structures and to represent the data distribution, the tools employ various strategies to transfer such distributed data between application components. Of the systems summarized below, InterComm, PAWS, CUMULVS, and MCT are targeted at building applications composed of independently developed codes. Roccom, Overture and Cactus, on the other hand, are problem solving environments targeted at building and running complex scientific applications in parallel, which include methods for communicating between processes within one program.

There is one community wide effort, sponsored by DOE, worth mentioning before summarizing particular development projects. The Common Component Architecture (CCA) forum (Allan et al., 2002; Armstrong et al., 1999; http://www.cca-forum.org/.) has been developing a set of common interfaces to provide interoperability among high performance application components. The CCA MxN working group (http://www.csm.ornl.gov/cca/mxn/) has designed interfaces to transfer data elements between parallel components running with different numbers of processes in each parallel component (hence MxN). The developers of several of the projects described here are contributors to the CCA.

InterComm (Lee and Sussman, 2004) is a runtime library that achieves direct data transfers between distributed data structures, in particular multidimensional arrays, among different parallel programs (application components). Programs do not need to know in advance any information about others with which they will communicate, since all the information required for data transfers is computed by InterComm at runtime. InterComm generates all the information required to execute direct data transfers between programs, building a customized all-to-all communication pattern (Snir et al., 1998), and storing the information in a communication schedule (Edjlali et al., 1997). After analyzing the data structures in the communicating programs, InterComm uses efficient algorithms to generate communication schedules which enable data to be sent directly from the processors on which it resides to the processors in the receiving program that are the destinations of the data, i.e., a complete MxN transfer. InterComm is a direct descendent of MetaChaos (Edjlali et al., 1997; Saltz et al., 1997, 1998), and can be used with any parallel program that can describe its data distribution, including explicit message passing programs using, for example, MPI (Snir et al., 1998).

Parallel Application Workspace (PAWS) (Beckman et al., 1998; Keahey et al., 2001) provides the ability to share data structures between parallel applications. PAWS supports scalar values and parallel multi-dimensional array data structures. An application defines a global domain to provide each process with a global view of a data structure across all processes. In PAWS, this global domain is divided into subdomains, with each subdomain assigned to one process, representing a local view of a part of the data structure. Since the shape of a PAWS sub-domain can be arbitrarily defined by an application, multi-dimensional arrays in PAWS can be partitioned and distributed completely generally. A PAWS controller is a process that links applications and parallel data structures in the applications. A PAWS application registers itself as an active application with the PAWS controller when it starts execution. The application also registers the data structures that it will share with other applications. To transfer data elements between data structures of two applications, the PAWS controller establishes a connection between those data structures using information in its registry, and uses the parallel layout of both data structures to compute communication schedules for the data transfer.

Collaborative User Migration, User Library for Visualization and Steering (*CUMULVS*) (Geist et al., 1997; Papadopoulos et al., 1998) is a middleware library that facilitates the remote visualization and steering of parallel applications, and supports sharing parallel data structures between programs. Although it supports multi-dimensional arrays like PAWS, arrays cannot be distributed in a fully general way. A multi-dimensional array is partitioned into chunks in each dimension (as in high performance FORTRAN (HPF) (Koelbel et al., 1994)), or each data element is explicitly distributed (assigned to a process) by the application. In addition, the application programmer must export a topology for processes that represent the ownership of each data chunk. A receiver program (a visualizer) in CUMULVS is not a parallel program. It specifies the data it requires in a request that is sent to the parallel sender program. After receiving the request, the sender program generates a sequence of connection calls to transfer the data.

Model Coupling Toolkit (*MCT*) (Larson et al., 2001) is a system that has been developed for the Earth System Modeling Framework (ESMF) (http://www.esmf.ucar.edu/). ESMF has developed various earth system simulation components and a flux coupler component. The flux coupler serves to transfer data between the physics simulations component using the MCT functionality. In MCT, a globalSegmentMap is defined to describe the distribution of a data structure across processes. The globalSegmentMap describes each continuous chunk of memory for the data structure in each process. Using globalSegmentMaps, MCT can generate a router—a communication scheduler that tells processes how to transfer data elements between a simulation component and the flux coupler. Therefore, all data transfers between two physics components are executed through the flux coupler.

Roccom (Jiao et al., 2003) is an object-oriented software framework for high performance parallel rocket simulation. Multiple physics modules have been developed to model various parts of the overall problem to build a comprehensive simulation system. A physics module builds distributed objects (data and functions) called windows and registers them in Roccom so that other modules can share them with the permission of the owner module. A window may be partitioned into multiple panes for parallelism, and each process in a module may have multiple panes. For example, if a window has a multi-dimensional array as its data attribute, the array can be partitioned into subarrays, each of which can be a pane.

Overture (Brown et al., 1997, 1999a, b) is a C++ framework designed for the solution of differential equations on overset, mapped grids. A mapped grid is a logically rectangular mesh that has a mapping function, either analytic or numerical, to a spatial grid. Overset grids are a hierarchical collection of grids. They are hierarchical in the sense that there is a definite priority associated with the grids; for example, where the highest priority grid exists, it takes precedence over all other grids and the calculation is done on that grid only. Overture has facilities for grid generation from analytic functions or numerical data sets describing some fraction of the grid. The set of nested grids and their interaction form a composite grid. Composite grids may be operated on in Overture in the same way as are individual grids. Physical variables can be assigned to the composite grids in Overture. Interpolations on overlap regions between grids are handled by Overture functions. In addition, Overture is built on top of the C++ parallel array class library P++, so that arithmetic operations with grid variables can use the simple syntax of P++. Grids that move relative to one another are also handled by Overture functions. Another package built on P++, called AMR++, provides extensions for Berger-type (Berger and Colella, 1989) adaptive mesh refinement.

CACTUS (Allen et al., 1999, 2001) is a scientific problem solving environment targeted at easing code development on various parallel platforms. It has integrated facilities for parallel I/O, PDE solvers, web interfaces, and visualization tools. The main interesting feature of Cactus is the ability to incorporate application specific code as thorns. A thorn connects to Cactus core services through the Cactus API. Cactus is essentially a component framework, like CCA, with many predefined components useful in various classes of scientific applications. Cactus also has facilities for executing components/thorns in parallel. Cactus employs the services of MPICH-G2 (Karonis et al., 2003) to execute application components in parallel, and in distributed computing environments.

3. Code coupling requirements

We have determined our coupling strategy through the evaluation of these packages with regard to the characteristics of the CISM codes, which establish the requirements for code coupling. As indicated in Luhmann et al. (2004), we are using a set of simulation codes that model both the large scale and microscale structures and dynamics of the Sun–Earth system. The fluid codes that together provide the global description of the solar terrestrial environment are the SAIC corona code (Linker et al., 1999; Mikic et al., 1999), the University of Colorado solar wind code (Odstrcil et al., 1996; Odstrcil and Pizzo, 1999a, b), the LFM global MHD magnetospheric code (Fedder et al., 1995a, b), and the TING (Wang et al., 1999) ionosphere–thermosphere code. These codes are supplemented and enhanced through interaction to codes that model specific regions or local physical processes with global impact. The internal ring particle dynamics is evaluated by following the particle drifts with the Rice Convection Model (Wolf, 1970). Energy release and particle acceleration are modeled using local MHD and kinetic codes including microscale Hall MHD (Shay et al., 2001), hybrid (Shay et al., 1998; Krauss-Varban, 1994), and particle codes (Shay and Drake, 1998).

Together these codes address the range of physical regimes needed for a comprehensive model. They present as well the coupling issues that we need to address within CISM (Table 1). Through a brief discussion of the physical domains that the codes cover and the sorts of linkage issues that arise, we highlight the requirements our framework must satisfy.

The first two codes are the causal chain from the Sun to upstream of the magnetosphere. The outward boundary of the coronal code is the inner boundary of the solar wind code. The boundary is particularly simple because the solar wind flow is taken to be supersonic and, thus, is all one way from corona to solar wind. The solar wind driving at 1 AU of the magnetosphere–ionosphere–thermosphere system is similarly a one way interaction. However, within geospace, the couplings of the codes are more complex, involving generally bi-directional interaction at both boundaries and within overlapping volumes.

The *SAIC corona code* (*MAS*) (Mikic et al., 1999) uses semi-implicit MHD to calculate the structure of the corona and the production of the solar wind. The code uses the magnetic field deduced from observations of the solar surface to provide the lower boundary for the code. It has been compared quite successfully against coronagraph observations and coronal whole patterns.

The *University of Colorado solar wind code* (ENLIL) (Odstrcil et al., 1996; Odstrcil and Pizzo, 1999a, b) has been used to model the solar wind from the inner heliosphere to the interaction with the interstellar medium. The inner boundary to this code allows for supersonic inflow of the solar wind plasma with spatially and temporally varying plasma and fields. The magnetosphere is actually embedded in the solar wind, but since the flow is supersonic, only solar wind information from the upwind side of the magnetosphere needs to be used for coupling.

The *Lyon–Fedder–Mobarry* (*LFM*) global MHD code (Fedder et al., 1995b) is in some sense the "spine" upon which the other Geospace models are attached. It solves the ideal MHD equations using solar wind conditions as input. An integral self-consistent, but extremely simplified, ionosphere model provides the inner boundary for the freestanding global code. The LFM provides input to all the other Geospace models (and receives necessary data back from them) in the ways indicated in Table 1. The other magnetospheric codes have been chosen to present the major sorts of linkages that will be generally needed:

The *microscale codes* will be embedded in the MHD code. The MHD codes (MAS, ENLIL, and LFM) supply the full suite of MHD variables on the boundaries of these codes. In turn, these codes supply the same variables (which can be extracted from the individual code variables) for the internal boundaries of the MHD domains as well as characteristics of energetic particles. The set of microscale codes include a two-fluid (Hall MHD with electron inertia) model (Shay et al., 2001), the hybrid models (particle-in-cell ions with fluid,

Table 1
Simulation codes included in this proposal showing the codes they link to and the type of linkage involved

Code	Institution	Links to	Linkage type	Coupling
SAIC Corona	SAIC	Solar Wind	Boundary to Boundary	$\Rightarrow$
Solar Wind	Colorado	Corona	Boundary to Boundary	$\Leftarrow$
		LFM	Internal Boundary to Boundary	$\Rightarrow$
LFM	Dartmouth	All but Corona	Spine	$\Leftrightarrow$
RCM	Rice	LFM	Volume to Volume	$\Leftrightarrow$
		TING (potentially)	Boundary to Boundary	$\Leftrightarrow$
Reconnection	Maryland	LFM	Boundary to Internal Boundary	$\Leftrightarrow$
TING	NCAR	LFM	Boundary to Boundary	$\Leftrightarrow$

The linkage is A (in column 1) to B (in column 3). The "Coupling" column indicates the flow of the coupling; $\Rightarrow$ implies info from col 1 to col 3, $\Leftarrow$ from col 3 to col 1, $\Leftrightarrow$ two-way coupling.

finite-mass electrons) (Shay et al., 1998; Krauss-Varban, 1994), and a full particle model (particle-in-cell electrons and ions) (Shay and Drake, 1998). These codes have been used to demonstrate the difference between classic resistive MHD reconnection and reconnection that includes the physics of electron whistlers, and have shown that the non-MHD physics plays a critical role in producing reconnection rates consistent with observed energy release times. In addition, the kinetic effects present in the hybrid and full particle codes lead to significant heating and particle acceleration that cannot be modeled with fluid codes.

The *Rice Convection Model* (*RCM*) follows the drift physics of thermal particles in the inner magnetosphere (Wolf, 1970). The different particle populations of the inner magnetosphere cannot be treated as a single fluid because they all move differently. By bounce averaging the motions of these particles, the calculation can be reduced to a two-dimensional problem (e.g., in the equatorial plane). However, the code needs the magnetic flux tube volume threading the reference plane as well as the electric field and plasma distribution on the outer boundary. These needs require both volume and boundary interaction with the LFM.

The *Thermosphere–Ionosphere Nested Grid* (*TING*) model is a three-dimensional, serial code designed to simulate the thermosphere/ionosphere system (Wang et al., 1999). It is an adaptation of the National Center for Atmospheric Research thermosphere/ionosphere general circulation model. The model itself is comprised of a global coarse grid and one (or more) nested grids inside the coarse grid. It interacts with the MHD code only on external boundaries for both codes. For TING, the MHD provides a precipitating electron flux and average energy as well as an electric potential for the ionosphere. TING provides in return the conductances needed for the solution of the electric potential equation by the MHD code. The variables passed must be mapped along field lines from the lower boundary of the MHD code to the ionosphere and interpolated to consistent grid positions.

We note three main technical issues regarding the coupling of these codes:

1. There is no simple mechanism by which information from one code can be shared with another code. Different codes generally use very different spatial layouts, different physical variables, and employ different temporal and spatial scales.
2. For codes that have been developed over an extended period of time, which is the rule in space science, changes in code structure to permit interoperation with other codes usually requires a major restructuring, perhaps including a change in programming language. It is unrealistic to expect such major code changes for the purpose of code coupling.
3. As seen in the preceding section, most modern framework software is based on object-oriented programming (OOP) technology; to be used effectively, existing codes must be completely rewritten. It also can be difficult for many scientists to use and often does not produce efficient code. The advantage of OOP, with features such as inheritance, data hiding, and function name overloading is that it makes the production of modular software systems, such as a comprehensive magnetosphere-ionosphere model, relatively easy to write and maintain. The drawback is that this flexibility makes it much more difficult for current compilers to produce highly optimized code.

Despite the coupling challenges they present, our codes share one important characteristic. All the codes use structured spatial grids to organize the calculations, as do most all space physics codes. Even rectangular block adaptive mesh refinement (AMR) codes can be viewed as having a set of nested grids.

4. Our coupling framework strategy

Given these issues and the nature of our codes, we have concluded that our framework approach should be a loosely coupled one, which coordinates the asynchronous operation of the individual codes. In this approach, it is appropriate to divide the problem into two parts: (1) the communication of data between codes and (2) the translation, interpolation, and filtering of data required to meaningfully transfer it between codes. After surveying currently available software packages, we have decided that the combination of the InterComm and Overture packages is the most appropriate basis for our framework.

Overture was developed first at Los Alamos and is now supported at Livermore, from which it is freely available. We have developed a good working knowledge of Overture through the revision of one of our codes, the LFM MHD code. As described earlier, Overture is an object-oriented framework written in C++ for computation on overlapping (overset) grids. It automatically handles the definition of overlapped regions and the interpolation between data on differing grids, which need not be regular and can be defined numerically. Overture is built upon the P++ class library, which allows Fortran90-like syntax through a C++ class library, and provides both general arithmetic operations and automatic domain-decomposition parallelization using the Multiblock PARTI software package (Agrawal et al., 1995). These features enable relatively straightforward development of parallel versions of C++ model codes. For example, the latest version of the LFM code executive is written in P++ to use these features to manage the data preparation and exchange and parallel execution of computational tasks in FORTRAN adopted from the serial version of LFM.

From the perspectives of both CISM and LWS, the major limitation of Overture is that a significant revision of existing codes generally is required to use its full capability. We address this problem by combining the capabilities of the InterComm communication library with Overture. As described earlier, InterComm supports communication between different parallel components either within a single program or running as separate programs, perhaps running at different sites. InterComm also allows parallelized programs to efficiently obtain results from parallelized sensor or scientific databases (e.g. databases accessed using the Maryland active data repository (Kurc et al., 1999; Chang et al., 1998, 1999, 2000) or DataCutter (Beynon et al., 2001) software). Like Overture, InterComm interoperates with the Multiblock PARTI package, which were both developed by the same group at Maryland. Thus integration of InterComm and Overture has been reasonably straightforward.

In this approach, all our codes will be linked with quite minor modification with the InterComm library enabling them to intercommunicate, and run as independent codes, assembled by a CISM or independent space scientist desiring to run a coupled simulation. The work needed to transform data from one code to another will be isolated in Overture code, running either as a separate program or through library calls within one or both programs involved in the data exchange. The Overture code can provide various services, including grid interpolation and physical data translation. This will minimize changes to our codes and shift all new functions into Overture, whose features will simplify this task. The result will be a collection of core and translation binary programs communicating in a loosely coordinated confederation.

By combining the strengths of the Overture and InterComm packages, we obtain the benefits of simplified coding of translation routines and inter-grid communication between distinct codes with minimal code modification. Let us emphasize this point; with these tools existing codes can be linked in most cases by modifying no more than a few lines of code. There may, of course, be a substantial amount of coding to translate from one grid to another and from one physical model to another. However, that work can all be done externally to the existing codes and only a few communication calls need to be inserted in the original programs. Fig. 1 shows a conceptual picture of how two codes, A and B, can be linked. The three program modules indicated in the figure could be peers running as separate programs (either sequential or parallel), or parts of a single program, or a combination of both. Our plan allows for all three possibilities, with the specific linkages differing as required. As an example, coupling of two legacy FORTRAN codes is probably most easily accomplished as three peer programs.

In our example, B is a calculation embedded in the larger simulation A. There is an overlap region for the two calculations where information needs to be passed back and forth. Array variable X in A provides the necessary information to B in the overlap region, but B requires Y for its calculation. A, in turn, requires an update on X in its overlap region with B. (It might appear that there is a bottleneck in this example, since A sends to the interface code which sends to B and then the data flow reverses. This appears to be a serial loop. However, the usual case is that, for example, the X sent back to A is actually the X values at an advanced time step. While A is waiting for this boundary information to return from B it is calculating the new X at each point in its non-overlapped region.) Beneath the code figures are text blocks showing the code fragments needed to produce the linkage. The arrows linking the blocks indicate the software tools used to do the actual communication. Since most Space Science codes are written in FORTRAN, we have shown the FORTRAN routines needed to perform the inter-program communication. Conceptually, only three calls are required: Export(X), Import(X), and BuildDescriptor(X). The action of the first two is obvious; the third builds the data descriptor required so that InterComm can schedule and perform the needed communications across all processes in each program. The center of the figure shows the inter-grid and data translation program. The code snippet here is in C++ and uses the Overture framework. The basic code quantities are P++ arrays (Quinlan, 2000). As previously noted, P++ arrays contain data descriptors that allow InterComm to access the array data.

Another advantage with this peer process technique of coupling, beyond minimal disturbance of the original codes, is the use of object oriented coding in the translation peer. Various features of OOP, such as function overloading, lend themselves to code reuse in new situations. Thus, once a coupling type is accomplished once, other similar couplings can be accomplished very rapidly. Linking codes will probably never be *plug and play*, but the interoperation strategy we are pursuing leads to as close to a modular system as is possible. Each successful coupling peer provides a template (in both the usual and the computer science meanings) for other linkages.

Our strategy addresses our requirements of efficient transmission of information among codes, and translation of physical data and interpolation of grid quantities between their different physical models, with minimal modification to the physics codes. In this strategy, the main computational cost we incur is the overhead of using InterComm to move data between codes resulting

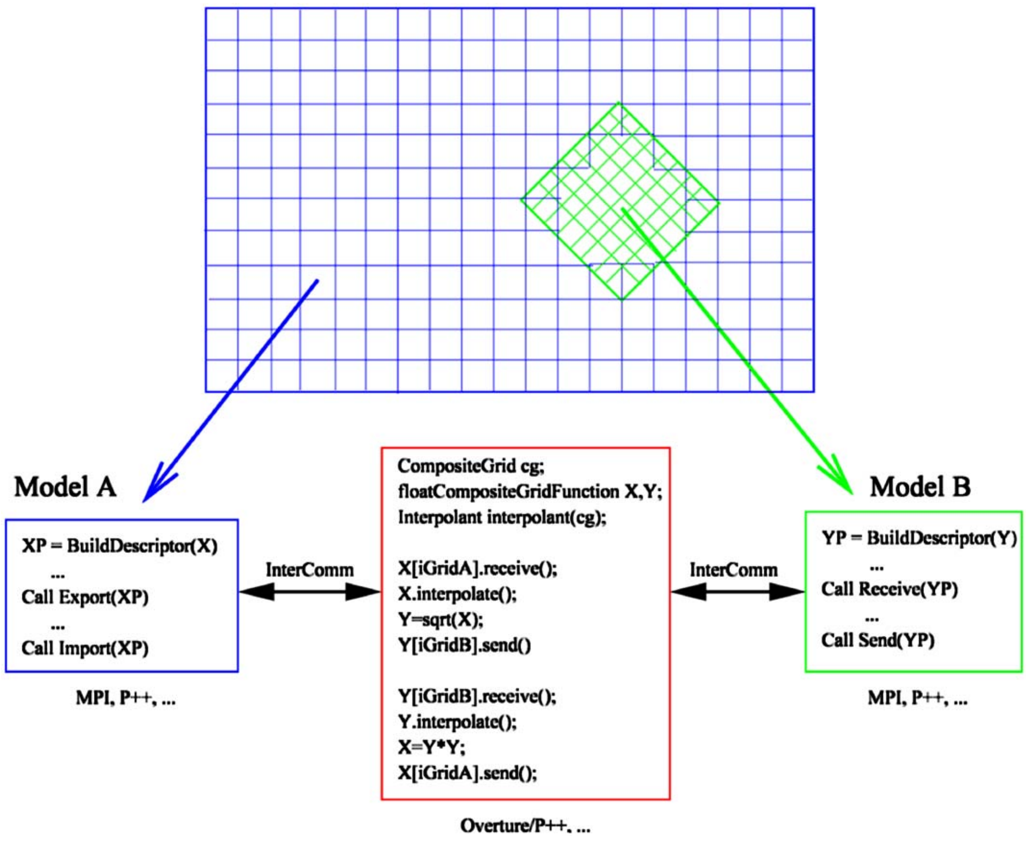

Fig. 1. A simple example illustrating how Overture and InterComm can be used to couple two simulation codes.

from generating the all-to-all communication pattern, which is quantified in multiple scenarios in Lee and Sussman (2004). While the InterComm implementation will continue to be optimized for better performance so that any programs using its facilities will benefit, the main benefit of using the library comes from the application developer not having to implement the coupling in an inflexible, ad hoc way.

Our strategy requires in addition the need to control the execution of our loosely coupled codes and synchronizing the data transfers among them. Using InterComm, an individual model code only specifies what data will be made available for a potential data transfer (either an import or an export), and does not specify when an actual data transfer will take place. Decisions about when data transfers will take place are made through a separate coordination specification, which is provided by the space scientist building the complete coupled simulation. Such a coordination specification can be used both to deploy model codes and grid translation/interpolation routines onto computational resources (i.e. how many and what machines to run each model code on), and to set up and control the interconnections among them at runtime (i.e. determine when data transfers occur). The coordination specification is read by InterComm at initialization, and used to determine at runtime when data transfers should be performed, effectively performing synchronization between pairs of communicating programs (model codes and/or grid manipulation routines). More details about the coordination process will appear in a future paper.

5. Preliminary coupling results

We have tested both InterComm and Overture in simple situations to confirm our use of them as the basis of our coupling strategy.

Using InterComm, we performed a data passage trial using two identical rectangular domains (256×64 cells) adjacent to each other in the x direction. Boundary data was passed at the common inner boundary as well as at the outer boundaries, resulting in a 512×64 periodic domain.

The code used in this coupling study was the massively parallel, three dimensional two-fluid code f3d developed at the University of Maryland (Shay et al., 2001). The code is parallelized using MPI, but this trial was performed on a single processor. The time is stepped forward using the trapezoidal leapfrog method (Zalesak, 1979; Guzdar et al., 1993) and fourth order accurate spatial differencing. Periodic boundary conditions are enforced in the y direction. This trial had no variation in the y direction, however, so it was essentially a one dimensional simulation. The code changes required to use Intercomm for this study totaled about 20 lines, which included library initialization, declaration of the (parts of the) arrays to be exchanged, and the calls to move the data.

In this trial, f3d was set to solve Faraday's Law,

$$\frac{\partial \vec{B}}{\partial t} = -\vec{\nabla} \times \vec{E}$$

with the Hall effect defining

$$\vec{E} = \vec{J} \times \vec{B},$$

where $\vec{B}$ is the magnetic field $\vec{E}$ is the electric field, and

$$\vec{J} = \vec{\nabla} \times \vec{B}$$

is the current density. These equations have been normalized to magnetic fields of strength B_0, arbitrary lengths of L_0, and times to $t_0 = \Omega^{-1} L_0^2/\delta^2$, where $\Omega = eB_0/mc$ is the ion cyclotron frequency, $\delta = c/\omega_p$ is the ion inertial length, $\omega_p = (4\pi n e^2/m)^{1/2}$ is the ion plasma frequency, n is the plasma density (assumed constant), e is the ionic charge, c is the speed of light, and m is the ion mass. The above equations can be combined to give

$$\frac{\partial \vec{B}}{\partial t} = -\vec{\nabla} \times [(\vec{\nabla} \times \vec{B}) \times \vec{B}].$$

A linear analysis, which assumes $\vec{B} = \hat{x} + \vec{B}_1 \exp(i\vec{k} \cdot \vec{x} - i\omega t)$ produces a dispersion relation of

$$\omega = k k_x,$$

where $k = (k_x^2 + k_y^2 + k_z^2)^{1/2}$ and $k_x = \vec{k} \cdot \hat{x}$. This describes a dispersive, circularly polarized wave known as a whistler wave, important in various plasma physics studies, such as collisionless magnetic reconnection. The phase speed of the whistler wave is k_x, which means that short waves travel faster than long ones.

The code was run in two domains, each of length $L_x = 51.2$ and $L_y = 25.6$. The initial conditions consisted of a pure whistler wave of wavelength $\lambda = L_x/10 = 5.12$ multiplied by a Gaussian envelope centered at $x = 0$ and wide enough to contain a few full wavelengths in the resultant wave packet. That is,

$$B_x(x, t = 0) = 1.0,$$
$$B_y(x, t = 0) = 0.1 \cos(2\pi x/\lambda) \exp(-x^2/32),$$
$$B_z(x, t = 0) = -0.1 \sin(2\pi x/\lambda) \exp(-x^2/32).$$

The initial B_y is pictured in the top plot of Fig. 2. The vertical dashed line marks the boundary of the two domains. For this system, $k_x = k$, so that $\omega = k^2$. The time step was 0.1.

The evolution of B_y can be seen in the lower three plots of Fig. 2. The wave approaches and passes through the artificial boundary smoothly, with no spurious reflections, as is evident at $t = 20.0$. Furthermore, the group velocity of the initial whistler wave is expected to be $v_g = d\omega/dk = 2k = 4\pi/\lambda = 2.45$, which is borne out rather well by the simulation, and the passage across the boundary does not disturb the agreement. Short wavelength components of the wave packet are traveling faster than the longer wavelength components, as expected. Finally, the physics code was also run on a single domain of 512×64 (i.e., without InterComm)

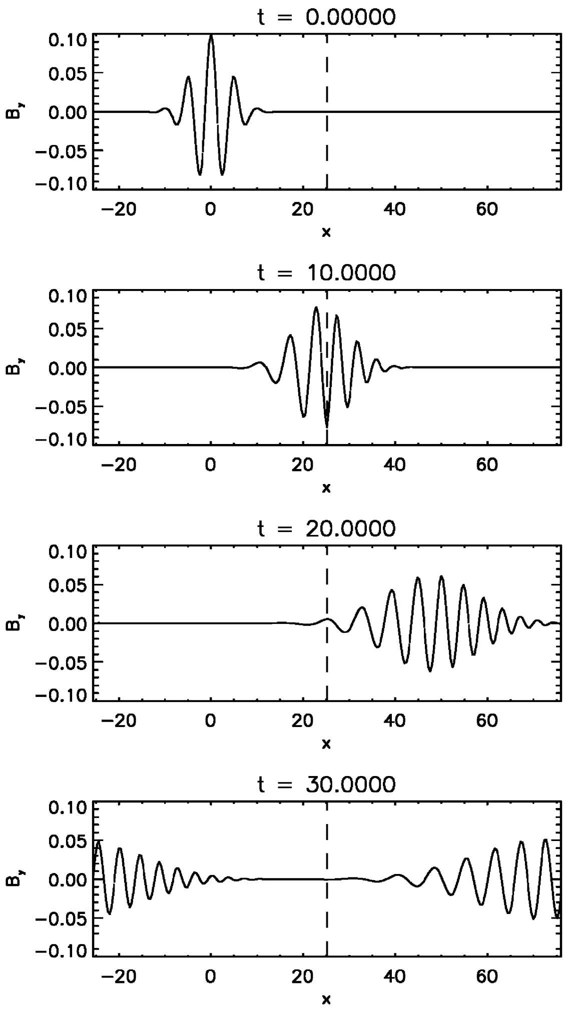

Fig. 2. Evolution of a wave packet across a boundary with data passed between the computational domains using InterComm. The lower plots are progressively later times. The vertical dotted line marks the boundary of the two domains.

and the results were identical. Clearly, the data passage was successful.

This trial was done, as described above, with computational domains with equal cell sizes. We intend to use InterComm to perform a simulation in a region where Hall physics is important embedded in a large region where it is not, such as in the case of magnetic reconnection in the Earth's magnetotail. Thus, further study will include using different cell sizes, different time steps and different physics in the different domains.

In addition, the Overture package has been briefly investigated for its applicability to the data passage problem using a trapezoidal leapfrog time stepping scheme on the convection-diffusion equation,

$$\frac{\partial\vec{v}}{\partial t}+(\vec{v}\cdot\vec{\nabla})\vec{v}=\mu\nabla^2\vec{v}$$

in a two dimensional, circular domain. The domain was covered using a square region for the interior with an annular region for the exterior. It was found that the trapezoidal leapfrog scheme was stable on the overlapping grid system. Also, stability tests of the diffusion equation on the same circular domain were performed, and it was found that the criterion that governs the stability is the one dependent on the cell spacing of the component grids, not the cell spacing of the overlapping grids. As the overlapping grid cells can be quite close, this is an important characteristic. This result supports the stability analysis performed by Duncan (1998). Thus, preliminary investigations of the package were successful. Further investigations are required to determine how Overture can be applied to a problem in which two different (but overlapping) domains have different physical laws pertinent in the two regions.

6. Conclusion

We have outlined here our strategy for coupling codes for CISM and LWS. By combined use of InterComm and Overture we benefit from their strengths and avoid their individual weaknesses; InterComm enables our codes to transfer data among themselves with truly minimal changes. The data translation and grid interpolation are shifted to Overture, in which these tasks are easy to implement. Our initial exploration of the performance of InterComm and Overture confirms the promise of our strategy.

Building on our experience with InterComm and Overture, and in the ad hoc coupling experiments described elsewhere in this volume, we will begin to apply our coupling strategy to the core CISM codes. We will begin with the LFM code, which essentially consists of two coupled codes, the MHD magnetosphere and the ionospheric model. Our first step will be to separate the components of LFM into programs linked through InterComm. In addition to immediately making the parallel LFM run more efficiently on many processors, this version will allow us to experiment with ways to develop code to link the LFM, TING, and the RCM together through their common thread, the ionospheric electrodynamics.

Acknowledgements

This material is based upon work supported by CISM, which is funded by the STC Program of the National Science Foundation under Agreement Number ATM-0120950, and the National Aeronautics and Space Administration (NAG5-12652).

References

Agrawal, G., Sussman, A., Saltz, J., 1995. An integrated runtime and compile-time approach for parallelizing structured and block structured applications. IEEE Transactions on Parallel and Distributed Systems 6 (7), 747–754.

Armstrong, R., Gannon, D., Geist, A., Keahey, K., Kohn, S., McInnes, L., Parker, S., Smolinski, B., 1999. Toward a common component architecture for high performance scientific computing. In: The Eighth IEEE International Symposium on High Performance Distributed Computing (HPDC). IEEE Computer Society Press, Piscataway, NJ, p. 13.

Allan, B., Armstrong, R.C., Wolfe, A.P., Ray, J., Bernholdt, D.E., 2002. The CCA core specification in a distributed memory SPMD framework. Concurrency and Computation: Practice and Experience 5 (15), 323–345.

Allen, G., Goodale, T., Lanfermann, G., Radke, T., Seidel, E., Benger, W., Hege, H.-C., Merzky, A., Massó, J., Shalf, J., 1999. Solving Einstein's Equations on Supercomputers. IEEE Computer 12 (32), 52–59.

Allen, G., Dramlitsch, T., Foster, I., Karonis, N., Ripeanu, M., Seidel, E., Toonen, B., 2001. Supporting efficient execution in heterogeneous distributed computing environments with cactus and globus. Supercomputing Conference 2001. ACM Press, New York.

Beckman, P., Fasel, P., Humphrey, W., Mniszewski, S., 1998. Efficient coupling of parallel applications using PAWS. In: Seventh IEEE International Symposium on High Performance Distributed Computing (HPDC). IEEE, New York, p. 215.

Berger, M.J., Colella, P., 1989. Local adaptive mesh refinement for shock hydrodynamics. Journal of Computational Physics 82, 64.

Beynon, M.D., Kurc, T., Catalyurek, U., Chang, C., Sussman, A., Saltz, J., 2001. Distributed processing of very large datasets with datacutter. Parallel Computing 27 (11), 1457–1478.

Brown, D.L., Chesshire, G.S., Henshaw, W.D., Quinlan, D.J., 1997. Overture: an object oriented software system for solving partial differential equations in serial and parallel environments. In: Eighth SIAM Conference on Parallel

Processing for Scientific Computing. SIAM, Philadelphia, PA UCRL-JC-132017.

Brown, D.L., Henshaw, W.D., Quinlan, D.J., 1999a. Overture: an object-oriented framework for solving partial differential equations on overlapping grids. In: SIAM Conference on Object Oriented Methods for Scientific Computing. SIAM, Philadelphia, PA, pp. 177–184.

Brown, D.L., Henshaw, W.D., Quinlan, D.J., 1999b. Overture: object-oriented tools for overset grid applications. In: AIAA Conference on Applied Aerodynamics. AIAA, Reston, VA UCRL-JC-134018.

Chang, C., Acharya, A., Sussman, A., Saltz, J., 1998. T2: a customizable parallel database for multi-dimensional data. ACM SIGMOD Record 27 (1), 58–66.

Chang, C., Ferreira, R., Sussman, A., Saltz, J., 1999. Infrastructure for building parallel database systems for multi-dimensional data. In: The Second Merged IPPS/SPDP Symposiums. IEEE Computer Society Press, Piscataway, NJ, pp. 582–589.

Chang, C., Kurc, T., Sussman, A., Saltz, J., 2000. Optimizing retrieval and processing of multi-dimensional scientific datasets. In: The Third Merged IPPS/SPDP (14th International Parallel Processing Symposium and 11th Symposium on Parallel and Distributed Processing). IEEE Computer Society Press, Piscataway, NJ, pp. 405–410.

Duncan, D.B., 1998. Difference approximations of acoustic and elastic wave equations. In: Toro, E.F., Clarke, J.F. (Eds.), Numerical Methods for Wave Propagation, Fluid Mechanics and its Applications. Kluwer Academic Publishers, Dordrecht, The Netherlands, pp. 197–210.

Edjlali, G., Sussman, A., Saltz, J., 1997. Interoperability of data parallel runtime libraries. In: The Eleventh International Parallel Processing Symposium. IEEE Computer Society Press, Piscataway, NJ, pp. 451–459.

Fedder, J.A., Lyon, J.G., Slinker, S.P., Mobarry, C.M., 1995a. Topological structure of the magnetotail as function of interplanetary magnetic field and with magnetic shear. Journal of Geophysical Research 100, 3613.

Fedder, J.A., Slinker, S.P., Lyon, J.G., Elphinstone, R.D., 1995b. Global numerical simulation of the growth phase and the expansion onset for substorm observed by Viking. Journal of Geophysical Research 100, 19083.

Geist, G.A., Kohl, J.A., Papadopoulos, P.M., 1997. CUMULVS: providing fault-tolerance, visualization and steering of parallel applications. International Journal of High Performance Computing Applications 3 (11), 224–236.

Guzdar, P.N., Drake, J.F., McCarthy, D., Hassam, A.B., Liu, C.S., 1993. Three-dimensional fluid simulations of the nonlinear drift-resistive ballooning modes in tokamak edge plasmas. Physical Fluids B 5 (10), 3712–3727.

Jiao, X., Campbell, M.T., Heath, M.T., 2003. Roccom: an object-oriented, data centric software integration framework for multiphysics simulations. In: The Annual ACM International Conference on Supercomputing. ACM Press, New York, pp. 358–368.

Karonis, N.T., Toonen, B., Foster, I., 2003. MPICH-G2: a grid-enabled implementation of the message passing interface. Journal of Parallel and Distributed Computing 5 (63), 551–563.

Keahey, K., Fasel, P., Mniszewski, S., 2001. PAWS: collective interactions and data transfers. In: The 10th IEEE International Symposium on High Performance Distributed Computing (HPDC). IEEE Computer Society Press, Piscataway, NJ, pp. 47–54.

Koelbel, C., Loveman, D., Schreiber, R., Steele Jr., G., Zosel, M., 1994. The High Performance Fortran Handbook. MIT Press, Cambridge, MA.

Krauss-Varban, D., 1994. Electron acceleration at nearly perpendicular collisionless shocks 3 Downstream distributions. J. Geophys. Res. 99, 2537–2551.

Kurc, T., Chang, C., Ferreira, R., Sussman, A., Saltz, J., 1999. Querying very large multi-dimensional datasets in ADR. In: The 1999 ACM/IEEE SC99 Conference. ACM Press, New York, p. 12.

Larson, W., Jacob, R., Foster, I., Guo, J., 2001. The model coupling toolkit. In: The International Conference on Computational Science. Springer, New York, pp. 185–194.

Lee, J.Y., Sussman, A., 2004. Efficient communication between parallel programs with InterComm. University of Maryland, Department of Computer Science and UMIACS Technical Report CS-TR-4557 and UMIACS-TR-2004-04.

Linker, J.A., Mikic, Z., Biesecker, D.A., Forsyth, R.J., Gibson, S.E., Lazarus, A.J., Lecinski, A., Riley, P., Szabo, A., Thompson, B.J., 1999. Magnetohydrodynamic modeling of the solar corona during Whole Sun Month. Journal of Geophysical Research 104, 9809–9830.

Luhmann, J.G., Solomon, S.C., Linker, J.A., Lyon, J.G., Mikic, Z., Odstrcil,D., Wang, W., Wiltberger, M., 2004. Coupled model simulation of a Sun-to-earth space weather event. Journal of Atmospheric and Solar-Terrestrial Physics, this issue.

Mikic, Z., Linker, J.A., Schnack, D.D., Lionello, L.R., 1999. Magnetohydrodynamic modeling of the global solar corona. Physics of Plasmas 6, 2217–2224.

Odstrcil, D., Pizzo, V.J., 1999a. Three-dimensional propagation of coronal mass ejections (CMEs) in a structured solar wind flow 1, CME launched within the streamer belt. Journal of Geophysical Research 104, 483–492.

Odstrcil, D., Pizzo, V.J., 1999b. Distortion of interplanetary magnetic field by three-dimensional propagation of CMEs in a structured solar wind. Journal of Geophysical Research 104, 28,225–28,239.

Odstrcil, D., Dryer, M., Smith, Z., 1996. Propagation of an interplanetary shock along the heliospheric plasma sheet. Journal of Geophysical Research 101, 19,973–19,986.

Papadopoulos, P., Kohl, J.A., Semeraro, D., 1998. CUMULVS: extending a generic steering and visualization middleware for application fault-tolerance. In: The 31st Hawaii International Conference on System Sciences (HICSS-31). IEEE Computer Society Press, Piscataway, NJ, pp. 127–136.

Quinlan, D., 2000. A + + /P + + Manual. Lawrence Livermore National Laboratory, Livermore, CA.

Saltz, J., Agrawal, G., Chang, C., Das, R., Edjlali, G., Havlak, P., Hwang, Y.-S., Moon, B., Ponnusamy, R., Sharma, S., Sussman, A., Uysal, M., 1997. Programming irregular applications: runtime support, compilation and tools. In: Zelkowitz, M. (Ed.), Advances in Computers. Academic Press, New York, pp. 105–153.

Saltz, J., Sussman, A., Graham, S., Demmel, J., Baden, S., Dongarra, J., 1998. The high-performance computing continuum: programming tools and environments. Communications of the ACM 41 (11), 64–73.

Shay, M.A., Drake, J.F., 1998. The role of electron dissipation on the rate of collisionless magnetic reconnection. Geophysical Research Letters 25, 3759–3762.

Shay, M.A., Drake, J.F., Denton, R.E., Biskamp, D., 1998. Structure of the dissipation region during collisionless magnetic reconnection. Journal of Geophysical Research 103, 9165–9176.

Shay, M.A., Drake, J.F., Rogers, B.N., Denton, R.E., 2001. Alfvenic collisionless magnetic reconnection and the Hall term. Journal of Geophysical Research 106, 3759–3772.

Snir, M., Otto, S., Huss-Lederman, S., Walker, D., Dongarra, J., 1998. MPI: The Complete Reference, second ed. MIT Press, Cambridge.

Wang, W., Killeen, T.L., Burns, A.G., Roble, R.G., 1999. A high-resolution, three dimensional, time dependent, nested grid model of the coupled thermosphere-ionosphere. Journal of Atmospheric and Solar-Terrestrial Physics 61, 385–397.

Wolf, R.A., 1970. Effects of ionospheric conductivity on convective flow of plasma in the magnetosphere. Journal of Geophysical Research 75, 4677–4698.

Zalesak, S.T., 1979. Fully multidimensional flux-corrected transport algorithms for fluids. Journal of Computational Physics 31, 335–362.