A Real Application of the Model Coupling Toolkit

Everest T. Ong, J. Walter Larson, and Robert L. Jacob

Argonne National Laboratory, Mathematics and Computer Science Division
9700 S. Cass Ave., Argonne, IL 60439, USA,
eong@mcs.anl.gov

Abstract. The high degree of computational complexity of atmosphere and ocean general circulation models, land-surface models, and dynamical sea-ice models makes coupled climate modeling a grand-challenge problem in high-performance computing. On distributed-memory parallel computers, a coupled model comprises multiple message-passing-parallel models, each of which must exchange data among themselves or through a special component called a *coupler*. *The Model Coupling Toolkit* (MCT) is a set of Fortran90 objects that can be used to easily create low bandwidth parallel data exchange algorithms and other functions of a parallel coupler. In this paper we describe the MCT, how it was employed to implement some of the important functions found in the flux coupler for the Parallel Climate Model(PCM), and compare the performance of MCT-based PCM functions with their PCM counterparts.

1 Introduction

The practice of climate modeling has progressed from the application of atmospheric general circulation models (GCMs) with prescribed boundary condition and surface flux forcing to *coupled earth system models*, comprising an atmospheric GCM, an ocean GCM, a dynamic-thermodynamic sea ice model, a land-surface model, a river runoff-routing model, and potentially other component models—such as atmospheric chemistry or biogeochemistry—in mutual interaction. The high degree of computational complexity in many of these component models (particularly the ocean and atmosphere GCMs) gave impetus to parallel implementations, and the trend in computer hardware towards microprocessor-based, distributed-memory platforms has made message-passing parallelism the dominant model architecture paradigm. Thus, coupled earth-system modeling can entail the creation of a *parallel coupled model*, requiring the transfer of large amounts of data between multiple message-passing parallel component models. Coupled models present a considerable increase in terms of computational and software complexity over their atmospheric climate model counterparts.

A number of architectures for parallel coupled models exist. One strategy is the notion of a *flux coupler* that coordinates data transfer between the other component models and governs the overall execution of the coupled model, which can

be found in the National Center for Atmospheric Research (NCAR) Climate System Model (CSM)[4][5][11] and the Parallel Climate Model (PCM)[1][3]. Another solution is the use of coupling libraries such as CERFACS' Ocean Atmosphere Sea Ice Surface[15], the UCLA Distributed Data Broker[8], and the Mesh-based parallel Code Coupling Interface (MpCCI)[13]. A third solution implements the coupler as an interface between the atmosphere and the surface components such as the Fast Ocean Atmosphere Model[9].

The authors have studied these previous strategies (and others), and decided to create a Fortran90 software toolkit to support the core functions common to coupled models: gridded data domain decomposition and parallel data transfer, interpolation between coordinate grids implemented as sparse matrix-vector multiplication, time averaging and accumulation, representation of coordinate grids and support for global integral and spatial averaging, and merging of output data from multiple component models for input to another component model. The result is the Model Coupling Toolkit (MCT)[12]. Fortran90 was chosen because the vast majority of geophysical models (our target application) are written in Fortran or Fortran90, and we wanted to provide those users with tools that exploit the advanced features of Fortran90, and avoid interlanguage operability issues. The decision to build a toolkit supporting the low-level data movement and translation functions found in flux couplers was driven by our desire to provide maximum flexibility to our users, and to support the widest possible range of coupled model applications.

In Section Two, we describe the PCM flux coupler benchmark, and present performance figures from the original benchmark and the results of the authors' optimization work on this code. In Section Three we describe the Model Coupling Toolkit. In Section Four, we explain how the MCT can be employed to create parallel coupled models, and present MCT-based pseudocode for our implementation of the PCM coupler benchmark. In Section Five, we report performance figures for the MCT implementation of the PCM coupler benchmark and compare them with previous PCM timings.

2 The Parallel Climate Model Coupler Benchmark

The Parallel Climate Model (PCM)[1][3] comprises the NCAR atmospheric model CCM3, the Parallel Ocean Program (POP) ocean model, a dynamic - thermodynamic sea ice model, the LSM land-surface model, and a river-runoff routing model, with a distributed flux coupler coordinating the transfer, inter-grid interpolation, and time- averaging and accumulation of interfacial flux and boundary condition data. Each of these component models is message-passing parallel based on the Message-Passing Interface (MPI) standard. The model is a single-load-image application, and its components are run sequentially in an event-loop as shown in Figure 1.

A unit-tester of the PCM coupler exists, and executes all the coupler's interpolation, time averaging and accumulation, and multi-component data-merging functions for a ten-day period using dummy components in place of the at-

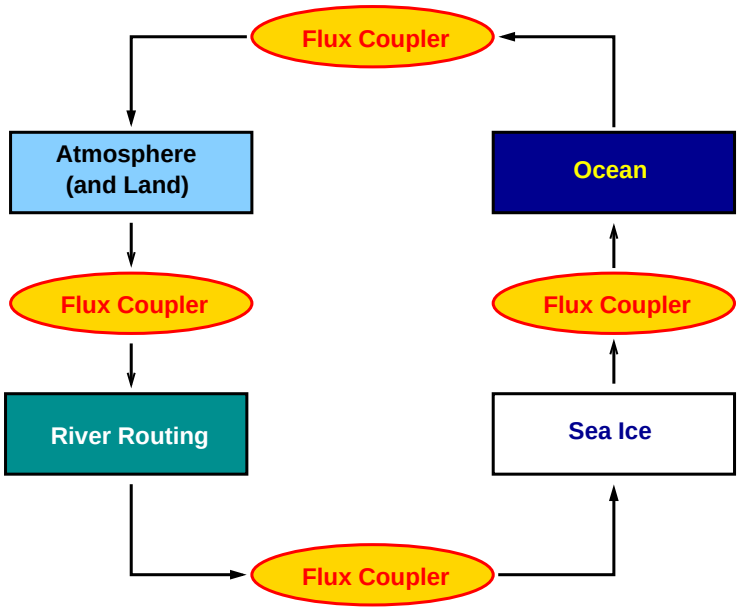

Fig. 1. Schematic of the Parallel Climate Model, showing the component model execution event loop.

mosphere, ocean, *et cetera*. The atmosphere-to-ocean grid operations are: 720 interpolation calls to regrid a bundle of two fields, and two sets of 720 interpolation calls to regrid six fields. The ocean-to-atmosphere interpolation operations are: 240 interpolation calls to interpolate one field (ice fraction), two sets of 240 interpolation calls to regrid six fields of ocean data, and 240 interpolation calls to regrid six fields of sea ice state data (residing on the ocean grid). For our discussion, the PCM coupler benchmark comprises four quantities: (1) total time spent on communications in the atmosphere-to-ocean interpolation, (2) total time spent on floating-point operations in the atmosphere-to-ocean interpolation, (3) total time spent on communications in the ocean-to-atmosphere interpolation, and (4) total time spent on floating point operations in the ocean-to-atmosphere interpolation.

The PCM coupler has been benchmarked for the IBM SP-3[6], assuming an atmosphere with a horizontal grid corresponding to T42 spectral truncation (8192 points) and an ocean with a POPx1 grid (110592 points). This benchmark reflects the coupler design for PCM version one, where the communications employed during the atmosphere-to-ocean and ocean-to-atmosphere interpolation operations sent excess data, and used blocking send and receive operations. Timings on the SP-3 for these operations are presented in Table 1, and are labeled as the "original" values.

The authors optimized the hand-coded PCM communications algorithms by using message-packing to cut bandwidth costs, and employing non-blocking com-

munications[14]. These changes resulted in dramatically reduced timings, as can be seen from the "optimized" values in Table 1.

Table 1. PCM Coupler Benchmark Timings (in seconds)

Number of Processors	Atmosphere-to-Ocean Communications (original)	Atmosphere-to-Ocean Communications (optimized)	Ocean-to-Atmosphere Communications (original)	Ocean-to-Atmosphere Communications (optimized)
16	13.364	2.919	14.542	2.059
32	15.807	2.393	17.138	1.441
64	11.653	2.188	15.090	1.234

3 The Model Coupling Toolkit

The MCT is a set of Fortran90 modules defining derived datatypes and routines that provide core services common to coupled models.

The MCT provides two classes of domain decomposition descriptors: a one-dimensional decomposition (the `GlobalMap`, defined in the module `m_GlobalMap`) and a segmented decomposition for multidimensional and unstructured grids (the `GlobalSegMap`, defined in the module `m_GlobalSegMap`). The MCT provides means for initializing and querying these descriptors, global-to-local indexing services, and the means for exchange of descriptors between component models.

The `MCTWorld` datatype (defined in the module `m_MCTWorld`) provides a registry for component models that describes the processes on which they reside. This allows for automatic translation between local (component model) and global process ID numbers needed for intercomponent model communications. The assignment of communicators and process ID's used by each component model on the communicator `MPI_COMM_WORLD` are determined by invoking the Multi-Process Handshaking (MPH) utility[7].

The `Router` datatype (defined in the module `m_Router`) encapsulates MCT parallel communications scheduling. A `Router` is initialized automatically from a pair of domain decomposition descriptors for the source and destination decompositions. This datatype allows for the automation of the complex parallel intercomponent model data transfers necessary for high performance.

The `AttrVect`, or *attribute vector*, is the MCT's flexible, extensible, and indexible data field storage datatype. This datatype is defined, along with its query and support routines in the MCT module `m_AttrVect`, and has point-to-point and collective communications routines defined by the module `m_AttrVectComms`. This allows the user maximum flexibility in configuring a coupled model. If the component models and coupler are designed properly, the fields passed between component models could be configured at runtime. It is possible to store both

real and integer data in this datatype, which allows a user to carry around associated gridpoint indexing information as a means of monitoring processing and an aid to debugging coupled model applications. The storage order in the derived type is field-major, which is compatible with the notion that most communication and computation in a coupler is processing of multiple fields of *point* data. This allows, where possible, for the field storage datatype to be used directly as a buffer for MPI operations and allows for reuse of matrix elements during the interpolation process, thus boosting performance for this process.

The MCT provides support for flux and state data field interpolation via an efficient sparse matrix-vector multiply that works directly with the MCT's `AttrVect` datatype. The MCT `SparseMatrix` datatype, defined in the module `m_SparseMatrix`, encapsulates sparse matrix element storage, along with space for global and local row-and-column indexing information. This module also provides facilities for sorting of matrix elements, which can yield a more regular memory-access pattern and thus boost single-processor performance during matrix-vector multiplication through better cache usage. Communications support for this datatype are provided in the module `m_SparseMatrixComms`. Schemes for automatically computing domain decompositions based on global row or column index information stored in a `SparseMatrix` are provided in the module `m_SparseMatrixToMaps`, and schemes for computing domain decompositions for the `SparseMatrix` are provided in the module `m_SparseMatrixDecomp`. Various parallel matrix-vector multiplication routines are defined in the module `m_MatAttrVectMul`. The MCT currently has no facilities for computing interpolation matrix elements, and it is assumed the user will compute them off-line using a package such as the Spherical Coordinate Regridding and Interpolation Package (SCRIP)[10].

The `GeneralGrid` datatype (defined in the module `m_GeneralGrid`) is the MCT coordinate grid desccriptor, and can support multidimensional and unstructured grids. This datatype allows the user to store geometric information such as grid-cell cross-sectional area and volume weights, and integer data for any number of grid indexing schemes. Grid geometry stored in a `GeneralGrid` is used by the spatial integration and averaging routines defined in the module `m_GlobalIntegrals`.

The `Accumulator` datatype (defined in the module `m_Accumulator`) provides flexible, extensible, and indexible registers for temporal averaging and accumulation of state and flux data. Communications support for this datatype are defined in the module `m_AccumulatorComms`.

The MCT module `m_Merge` provides routines for merging flux and state data from multiple component models for use by another component model, with fractional area weighting and/or masking encapsulated either in the `GeneralGrid` or `AttrVect` datatypes.

The MCT is built upon the NASA DAO Message Passing Environment Utilities (MPEU), which provide support for basic low-level data types upon which MCT classes are built, Fortran90 module-style access to MPI, and tools for error handling, sorting data, and timing. Both MPEU and MCT have their doc-

umentation built-in, implemented as extractible prologues compatible with the software package ProTeX, which translates the prologues into LaTeX.

The programming interface to the MCT is described fully in the MCT API Definition Document, which is available via the MCT Web site

`http://www.mcs.anl.gov/acpi/mct .`

Currently, the MCT is in beta release. A complete formal release of the MCT will occur in 2002. Work is currently under way to implement the new flux coupler for the Community Climate System Model (CCSM) using MCT[2]

4 Implementation of PCM Flux Coupler Functions using the MCT

Using MCT to couple message-passing parallel models into a coupled parallel model is relatively easy. The user accesses MCT datatypes and modules through Fortran90 use association, creating instantiations of the MCT objects and invoking MCT routines as needed to accomplish the coupling. For the sake of simplicity we consider only the atmosphere-ocean interactions, and present pseudocode where the only coupling activities involved are grid interpolation. In this case, only a subset of the MCT is visible to the user. For example, the `Router` datatype is invoked inside the matrix-vector interpolation routines, and thus is invisible to the user.

The MCT datatypes shared between the ocean, atmosphere, and coupler are defined in the module `m_Couplings`:

```
module m_Couplings
  use m_AttrVect       ! Field Storage Datatype
  use m_GlobalSegMap   ! Domain Decomposition Descriptors
  use m_SparseMatrix   ! Interpolation Matrices
  ! AttrVect for atmosphere data
  type(AttrVect) :: CouplerInputFromAtmos, CouplerOutputToAtmos
  ! AttrVect for ocean data
  type(AttrVect) :: CouplerInputFromOcean, CouplerInputToOcean
  ! Atmosphere and Ocean GlobalSegMap
  type(GlobalSegMap) :: AtmosGlobalSegMap, OceanGlobalSegMap
  ! Atmosphere-to-Ocean and Ocean-to-Atmosphere SparseMatrix
  type(SparseMatrix) :: AtmosToOceanSMat, OceanToAtmosSMat
end module m_Couplings
```

Pseudocode for the atmosphere, showing how MCT datatypes are used, and routines are invoked is contained in the atmosphere subroutine `ccm()` shown below. The pseudocode for the ocean is analgous, and is not shown for the sake of brevity.

```
subroutine ccm(AtmosInputFromCoupler,AtmosOutputToCoupler)
  use m_MCTWorld          ! MCT Component Model Registry
```

```
  use m_AttrVect          ! Field Storage Datatype
! Initialize MCTWorld registry:
  if(initialization) call MCTWorld_Init()
! Initialize Coupler Atmosphere Domain Decomposition:
  if(initialization) call GlobalSegMap_Init(AtmosGlobalSegMap)
  :
! Unpack atmosphere input from AtmosInputFromCoupler AttrVect
  :
! Atmosphere model integration loop!
  do step = 1,nsteps
    :             :
  enddo
:
! Pack atmosphere output into AtmosOutputToCoupler AttrVect
:
  if(shutdown) then
    ! Destroy Coupler Atmosphere Domain Decomposition:
    call GlobalSegMap_Clean(AtmosGlobalSegMap)
    ! Destroy MCTWorld registry
    call MCTWorld_Clean()
  endif
end subroutine ccm
```

Pseudocode for the distributed flux coupler, which executes the ocean and atmosphere components, and interpolates data between the atmosphere and ocean grids is shown below in the program `cpl`.

```
program cpl
  use m_MCTWorld          ! MCT Component Model Registry
  use m_GlobalSegMap      ! Domain Decomposition Descriptors
  use m_AttrVect          ! Field Storage Datatype
  use m_SparseMatrix      ! Interpolation Matrices
  use m_SMatAttrVectMult  ! Parallel Sparse-Matrix-Attribute
                          ! Vector Multiplication Functions
! Initialize MCTWorld registry
  call MCTWorld_Init()
! Initialize input and output Attribute vectors
  call AttrVect_Init(CouplerInputFromAtmos)
  call AttrVect_Init(CouplerOutputToAtmos)
  call AttrVect_Init(CouplerInputFromOcean)
  call AttrVect_Init(CouplerOutputToOcean)
! Initialize Interpolation Matrices:
  call SparseMatrix_Init(AtmosToOceanSMat)
  call SparseMatrix_Init(OceanToAtmosSMat)
  :
! Coupler Time Evolution Loop:
  do icloop=1,ncloops
```

```
      if(mod(icloop,1)==0) then
         ! Run the atmosphere model:
         call ccm(CouplerOutputToAtmos, CouplerInputFromAtmos)
         ! Interpolate this data onto the Ocean Grid.  Here the
         ! matrix elements distributed by row (i.e. based on the
         ! ocean grid vector decomposition).
         call SMatAttrVectMul_xdyl(CouplerInputFromAtmos, &
                                   AtmosGlobalSegMap, &
                                   AtmosToOceanSMat, &
                                   CouplerOutputToOcean )
      endif
      if(mod(icloop,2)==0) then
         ! Run the ocean model:
         call pop(CouplerOutputToOcean, CouplerInputFromOcean)
         ! Interpolate this data onto the Atmosphere Grid.  Here
         ! he matrix elements distributed by column (i.e. based
         ! on the ocean grid vector decomposition).
         call SMatAttrVectMul_xlyd(CouplerInputFromOcean, &
                                   OceanGSMap, &
                                   OceanToAtmosSMat, &
                                   CouplerOutputToAtmos )
      endif
   end do
! Destroy input and output Attribute vectors
   call AttrVect_Clean(CouplerInputFromAtmos)
   call AttrVect_Clean(CouplerOutputToAtmos)
   call AttrVect_Clean(CouplerInputFromOcean)
   call AttrVect_Clean(CouplerOutputToOcean)
! Destroy Interpolation Matrices:
   call SparseMatrix_Clean(AtmosToOceanSMat)
   call SparseMatrix_Clean(OceanToAtmosSMat)
! Destroy MCTWorld registry
   call MCTWorld_Clean()
end program cpl
```

The MCT version of the PCM coupler is very similar to the pseudocode presented above.

5 Performance

Performance results for the MCT implementation of the PCM coupler benchmark are presented in Table 2. The communications routing mechanisms in the MCT are far more flexible than those in the hand-tuned PCM coupler, but comparison of these results with those in Table 1 show the atmosphere-to-ocean and ocean-to-atmosphere communications costs for the MCT implementation are either the same or slightly lower.

Table 2. MCT/PCM Coupler Benchmark Communications Timings (seconds)

Number of Processors	Atmosphere-to-Ocean	Ocean-to-Atmosphere
16	2.909	1.809
32	1.609	1.359
64	1.452	1.156

Floating-point operation timings for the PCM interpolation operations for both the original PCM implementation and the MCT implementation are presented in Table 3. The computation costs are not significantly worse, and usually are significantly better than the original PCM timings. This is very encouraging considering the hand-tuned PCM used f77-style, static arrays, and the MCT implementation employs Fortran90 derived types built on top of allocated arrays. Both the PCM and MCT implementations have been designed so that the loop-order in the interpolation is cache-friendly. The most likely reasons for better computational performance in the MCT implementation are (1) the use of local rather than global arrays for the computations, and (2) the application of sorting of matrix elements in the MCT implementation, both of which improve data-locality and thus improve cache performance.

Table 3. PCM Coupler Benchmark Floating-point Operations Timings (seconds)

Number of Processors	Atmosphere-to-Ocean (PCM)	Atmosphere-to-Ocean (MCT)	Ocean-to-Atmosphere (PCM)	Ocean-to-Atmosphere (MCT)
16	4.636	4.408	2.484	1.616
32	2.470	2.644	1.622	1.180
64	2.188	1.936	1.201	0.984

6 Conclusions

We have described a realistic application of the the Model Coupling Toolkit to implement some of the core functions found in the Parallel Climate Model. The pseudocode presented shows that creating coupled models using the MCT is straightforward. Performance results indicate that the use of advanced features of Fortran90 in this case do not degrade performance with respect to an optimized version of the f77 PCM counterpart. Floating-point performance in the MCT version of the PCM benchmark is slightly better than the original, due to better data locality.

Acknowledgements: We wish to thank many people for the useful discussions and advice regarding this work: Tom Bettge, and Tony Craig of the National

Center for Atmospheric Research; and John Taylor and Ian Foster of the Mathematics and Computer Science Division of Argonne National Laboratory, and Jace Mogill and Celeste Corey of Cray Research. This work is supported by the United States Department of Energy Office of Biological and Environmental Research under Field Work Proposal 66204, KP1201020, and the Scientific Discovery through Advanced Computing (SciDAC) Program, Field Work Proposal 66204.

References

1. Bettge, T. , and Craig, A. (1999). Parallel Climate Model Web Page. `http://www.cgd.ucar.edu/pcm/`.
2. Bettge, T. (2000). Community Climate System Model Next-generation Coupler Web Page. `http://www.cgd.ucar.edu/csm/models/cpl-ng/`.
3. Bettge, T. , Craig, A. , James, R. , Wayland, V. , and Strand, G. (2001). The DOE Parallel Climate Model (PCM): The Computational Highway and Backroads. *Proceedings of the International Conference on Computational Science (ICCS) 2001* , V.N. Alexandrov, J.J. Dongarra, B.A. Juliano, R.S. Renner, and C.J.K. Tan (eds.), Springer-Verlag Lecture Notes in Comupter Science Volume 2073, pp 149-158.
4. Boville, B. A. and Gent, P. R. (1998). The NCAR Climate System Model, Version One. *Journal of Climate*, **11**, 1115-1130.
5. Bryan, F. O. , Kauffman, B. G. , Large, W. G. , and Gent, P. R. (1996) The NCAR CSM Flux Coupler. NCAR Technical Note NCAR/TN-424+STR, National Center for Atmospheric Research, Boulder, Colorado.
6. Craig, A. (2000). Parallel Climate Model IBM SP-3 Benchmarks Web Page. `http://www.cgd.ucar.edu/ccr/bettge/ibmperf/timers.txt`.
7. Ding, C. H. Q. and He, Y. (2001). MPH: a Library for Distributed Multi-Component Environments, Users' Guide. NERSC Documentation, available on-line at `http://www.nersc.gov/research/SCG/acpi/MPH/mph_doc/mph_doc.html`.
8. Drummond, L. A. , Demmel, J. , Mechoso, C. R. , Robinson, H. , Sklower, K. , and Spahr, J. A. (2001). A Data Broker for Distributed Computing Environments. *Proceedings of the International Conference on Computational Science (ICCS) 2001* , V.N. Alexandrov, J.J. Dongarra, B.A. Juliano, R.S. Renner, and C.J.K. Tan (eds.), Springer-Verlag Lecture Notes in Comupter Science Volume 2073, pp 31-40.
9. Jacob, R. , Schafer, C. , Foster, I. , Tobis, M. , and Anderson, J. (2001). Computational Design and Performance of the Fast Ocean Atmosphere Model, Version One. *Proceedings of the International Conference on Computational Science (ICCS) 2001* , V.N. Alexandrov, J.J. Dongarra, B.A. Juliano, R.S. Renner, and C.J.K. Tan (eds.), Springer-Verlag Lecture Notes in Comupter Science Volume 2073, pp 175-184.
10. Jones, P. W. (1999). First- and Second-Order Conservative Remapping Schemes for Grids in Spherical Coordinates. *Monthly Weather Reveiw*, **127**, 2204–2210.
11. Kauffman, B. (1999). NCAR Climate System Model (CSM) Coupler Web Page. `http://www.ccsm.ucar.edu/models/cpl/`.
12. Larson, J. W. , Jacob, R. L. , Foster, I. T. , and Guo, J. (2001). The Model Coupling Toolkit. *Proceedings of the International Conference on Computational Science (ICCS) 2001* , V.N. Alexandrov, J.J. Dongarra, B.A. Juliano, R.S. Renner, and C.J.K. Tan (eds.), Springer-Verlag Lecture Notes in Comupter Science Volume 2073, pp 185-194.

13. Pallas GmbH (2002) MPCCI Web Site. http://www.mpcci.org.
14. Ong, E. T. and Larson, J. .W. (2001). Optimization of the Communications in the PCM Flux Coupler. Technical Memorandum, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, in preparation.
15. Valcke, S. , Terray, L. , Piacentini, A. (2000). OASIS 2.4 Ocean Atmosphere Sea Ice Soil Users' Guide. CNRS/CERFACS Technical Note, available on-line at http://www.cerfacs.fr/globc/.