The Spheroidal Analysis Library and Toolkit: Tools for Climate Model Output Analysis

[1,2,3]J. W. Larson, [2]E. T. Ong, and [2]C. Tokarz

[1]ANU Supercomputer Facility, The Australian National University
Canberra ACT 0200, Australia
E-Mail: Jay.Larson@anu.edu.au
[2]Mathematics and Computer Science Division, Argonne National Laboratory
9700 S. Cass Avenue, Argonne, IL 60439, USA
[3] Computation Institute, University of Chicago
Chicago, IL, USA

Keywords: *Spatial Statistics; Climate Data Analysis; Software Architecture; Parallel Computing*

EXTENDED ABSTRACT

We believe two great opportunities exist in the analysis of climate model data: 1) introduction of parallel computing techniques; and 2) expansion of the types of techniques used to study these data.

Climate modelling has long been recognised as a grand-challenge computational science application. The analysis of climate model output, however, has remained a workstation or personal computer (PC) application using a single-processor programming approach. This situation will not last as global climate models approach mesoscale resolution, and begin to produce petabyte-scale datasets for long integrations.

Climate model and reanalysis data represent large multivariate, logically Cartesian datasets. The ability to analyse and manipulate these data with aplomb is crucial to studies of climate variability, model intercomparison, and model validation. Most studies using these data rely on low-order statistics (i.e. means and variances), forgoing opportunities for deeper and richer analyses—for example, probability density function estimation and high-order statistics from information theory.

A typical analysis application takes input data from files, passes these data in the form of arrays and associated metadata to analysis compute kernels, and after analysis these data will be passed to either visualisation utilities or to file-handling utilities that will create output files (Figure 1). Our work is focused on the center box ("Analysis of Gridded Data").

There is a need for spatially-aware, robust, modular, flexible, and extensible tools that can enable a wide variety of climate data analyses, but are also portable to parallel architectures. Our object is to create a programming model and overall design for analysis kernels that will 1) result in a straightforward programming model that enables rapid development of analysis applications, and 2) will allow the seamless introduction of parallel computing techniques. In this paper we concentrate on the object foundation and will discuss only in brief a parallelisation strategy.

The Spheroidal Data Analysis Library and Toolkit (SpheroiDAL-Tk) is a set of tools for analysing Cartesian gridded data in curvilinear coordinates. SpheroiDAL-Tk offers a data object model for describing and storing gridded field data, and a set of associated tools that manipulate these objects to perform analyses across lower-dimensional subsets (e.g., axes) of the data. We will describe the SpheroiDAL-Tk data model classes and their basic query and manipulation methods, including data transposes, sub-sampling, and integrals. We will illustrate with examples the SpheroiDAL -Tk analysis infrastructure that support spatially aware analysis methods including spatial integrals and eddy statistics, scale separation, probability density estimation, parameter estimation, and information-theoretic metrics. We will describe the SpheroiDAL-Tk Fortran-based programming model, and discuss how it can be extended to other languages. We conclude with a parallel processing roadmap for SpheroiDAL-Tk that encompasses both distributed-memory (i.e., message-passing) and shared-memory (e.g., multicores) parallelism.

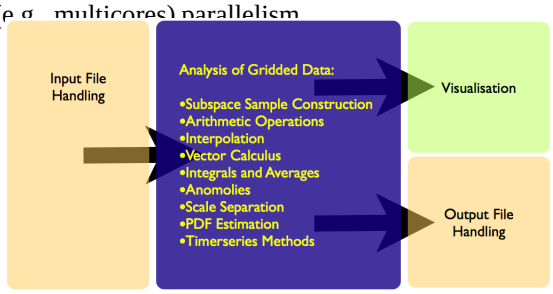

Figure 1. The data analysis and visualisation process.

1 INTRODUCTION

Dramatic performance gains by global earth system models will soon create a crisis situation in the analysis of their output data, and each of the processes in Figure 1 will require parallelism. Soon these models will scale to thousands of processors, will run with high resolution ($\sim$ 10 km) grids, and long integrations will produce petabyte-scale output. Current popular climate data analysis packages remain single-processor applications, and will soon be inadequate to address the large amounts of data they are expected to handle. This problem is twofold: 1) inadequate single address-space memory to handle large data volumes; and 2) insufficient processing power to analyse these data in a timely fashion. The emergence of multicore processors now makes parallel processing relevant to workstations and PCs.

The advent of ensemble climate modelling, and the interest in daily and sub-daily sampled data also creates new challenges. Probability density function (PDF) estimation is increasingly popular, as is the interest in extreme events. PDF estimation requires at a minimum some binning mechanism, but more rigorous approaches entail the use of kernel smoothing techniques. Comparison of two PDFs is an even more difficult task. Extreme events analyses can require sorting and ordering of data. Neither of these functions is offered in popular climate data analysis packages such as Climate Data Analysis Tools (CDAT; http://www-pcmdi.llnl.gov/software-portal/cdat), NCAR Command Language (NCL; http://www.ncl.ucar.edu/), and netCDF Operators (NCO; http://nco.sourceforge.net/).

Both CDAT and NCL offer scientist-friendly programming approaches suitable for single-processor architectures. These packages are not immediately portable to popular parallel programming approaches such as MPI or OpenMP; CDAT's implementation language of Python can be interfaced directly with MPI but not OpenMP, and NCL is not open-source, and thus we are unable to modify it. This situation has motivated our strategy of a lightweight data object model similar to—but more general than—CDAT implemented in Fortran, a language directly compatible with MPI and OpenMP. We call this new package the Spheroidal Data Analysis Library and Toolkit (SpheroiDAL-Tk). This is data model has been pursued to create a relatively straightforward programming model with ease of use similar to CDAT or NCL, and one that ultimately will be more amenable to the inclusion of both shared-memory and distributed-memory parallelism.

In Section 2 we will describe the SpheroiDAL-Tk data model and state the package's functionality. In Section 3 we will describe the SpheroiDAL-Tk programming model and present simple examples. In Section 4 we present a nontrivial example of scale separation using mean polishing, which constitute the first results published for this technique. In Section 5 we present a strategy for parallelising SpheroiDAL-Tk. In Section 6 we summarise and chart our future course.

2 DESIGN AND FEATURES

Here we present the SpheroiDAL-Tk data model, and describe the system's core anlysis functionality.

2.1 DATA MODEL

The object model for SpheroiDAL-Tk's mesh representation is shown in Figure 2. Two classes are of immediate interest to users, while other lower-level "service classes" are hidden from the user. The classes visible to the user are the CartesianMesh and the CartesianField, and the SpheroiDAL-Tk spatial statistics API is organised around these objects. Note that the current design is being prototyped in Fortran95, and fully object-oriented programming is not supported by this language. When we use the terms "class" and "method", we are following the convention set by Decyk et al. [1997], in which a class is a Fortran datatype with well defined interfaces that serve as its methods.

Field data residing on a logically Cartesian mesh is encapsulated by the CartesianField, which contains the field data stored as an appropriately-dimensioned array, and has its spatial mesh described by a CartesianGrid. The CartesianGrid encapsulates the description of any logically Cartesian grid. That is, the mesh points are the Cartesian product of a set of *axes*. The CartesianGrid stores the following attributes for a structured grid: its name; a brief description of the grid; its dimensionality; definition of each axis (using the Axis class); definition of each of the spatial weights (using the Weight class). The Axis class holds the name of the axis, the number of points along the axis, and their corresponding coordinate values. The Weight class encapsulates the data that defines a spatial weight, including its name, a brief description, the dimension(s) upon which it operates, and the spatial dimensions over which it varies. Spatial weights can vary in dimension, depending on what symmetries are present (e.g., azimuthal symmetry in the longitude λ in spherical coordinates). The flexibility of the Weight class allows us to exploit where present underlying symmetries in curvilinear grids to store as compactly as possible their spatial weights. This flexibility is enabled by basing our design of the Weight on the GenericArray class, which implements generic arrays. SpheroiDAL-Tk currently supports real-valued general arrays, and

allows dimensionality ranging from zero (scalar) to four. Hence, the current maximum dimensionality of a SpheroiDAL-Tk grid is four.

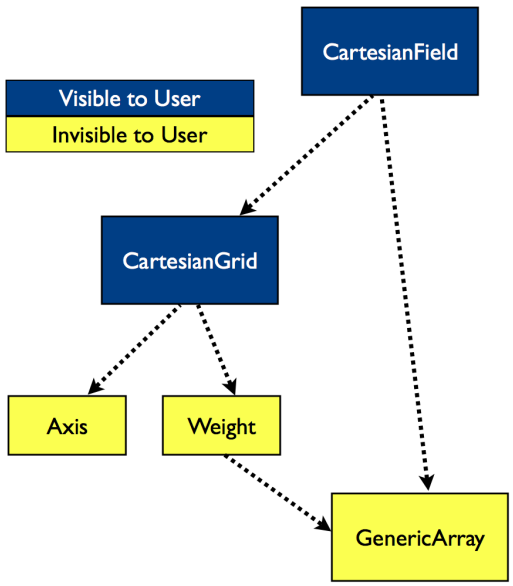

Figure 2. Class dependency diagram for the SpheroiDAL data model.

In addition to the data structures in which class data is held, these objects also have service methods that include creation and destruction, and set and get (query) methods. Higher-level manipulation and analysis functionality is described in the next section.

2.2 FUNCTIONALITY

Ideally, one data toolset would incorporate the types of meteorological and climate analysis capabilities present in the popular tools cited in Section 1. We have long admired the comprehensive toolset offered by the now-defunct CCM Modular Processor package, and would like in the long-term to see a toolset that again matches its functionality. For now, we are concentrating on the generic spatiotemporal statistics kernels required by any good meteorological analysis package, and will add the application-specific funcionality at a later time, implemented on top of the SpheroiDAL-Tk data model and compute kernels. This general approach allows application of the SpheroiDAL-Tk code base to other problems; for example the analysis of gridded data in toroidal coordinates found in modelling of fusion plasmas in tokamaks.

SpheroiDAL-Tk supports a wide variety of basic analyses centered on the CartesianField class. Basic arithmetic binary operations including addition, subtraction, multiplication, and division are supported both as operations involving two CartesianField objects of the same dimensionality (e.g., calculating differences between fields), or of differing dimensionality (e.g., computing an anomaly field). Arithmetic operations involving scalars (e.g., rescaling by a constant value) are also supported. Computation of spatial and time integrals are supported. A module for computation of moments of arbitrary order is also included. This core set of functions allows computation of spatial and temporal means and anomalies as well as moments, correlations, and covariances.

A comprehensive set of multi-key, MergeSort-based sorting tools allows for the determination of medians, quantiles, and rank correlations.

Built on top of these basic averaging routines is a set of scale-separation functions based on the techiniques of mean- and median-polishing Cressie [1993].

A number of tools for distribution function analysis are included. Modules encapsulating the normal and Weibull distributions are included, each offering functions to compute both their respective PDFs and cumulative distribution functions (CDF). An object-based Newton's method solver is included for maximum-likliehood estimation (MLE) of distribution function parameters. PDF estimation through histogram construction is supported by a UnivariateTable class, and associated methods that allow for straight-forward binning of data sampled in a particular direction or subsbace of a CartesianField. One- and two-sample Kolmogorov-Smirnov test functions are also offerd to compare and contrast distribution functions.

Information-theoretic measures are available, notably the Shannon Entropy and block-entropy complexity measures that are used in symbolic dynamics and computational mechanics.

3 PROGRAMMING MODEL

The programming model for SpheroiDAL-Tk follows the Fortran approach of *module use* to gain access to class definitions and explicit interfaces to their methods and other library routines, *declaration* of variables of specific datatypes corresponding to classes in use in a given application, and *invocation* of the necessary class methods or routines to accomplish the desired analysis task. The SpheroiDAL-Tk prototype code is implemented using the Fortran95 standard, and this is currently the only programming language supported. We envison supporting other programming languages such as C++ and Python through use of the Babel language interoperability toolkit, which we have used with success to create a multilingual programming model for the Model Coupling Toolkit (Ong et al. [2007]).

To illustrate this process and the power of SpheroiDAL-Tk, we present two simple, commonly-encounterd examples in the analysis of global climate data: 1) computation of zonal averages $\bar{F}$ for a two-dimensional horizontal field $F(\phi, \lambda)$, and 2) computation of a timeseries of global averages from a two-dimensional plus time dataset $T(\phi, \lambda, t)$.

3.1 ZONAL AVERAGES

Consider a field $F = F(\phi, \lambda)$, which varies over the latitude $\phi \in [-90, 90]$ and longitude $\lambda \in [-180, 180]$—for example, the geographic distribution of the time-averaged sea-level pressure field PSL from a coupled climate model (Figure 3). The zonal average is the spatial average in the longitudinal direction along a given latitude band, that is $\bar{F}(\phi) = \frac{1}{2\pi} \int_{-180}^{180} F(\phi, \lambda) d\lambda$. Below we describe the computation of $\bar{F}(\phi)$ using SpheroiDAL-Tk.

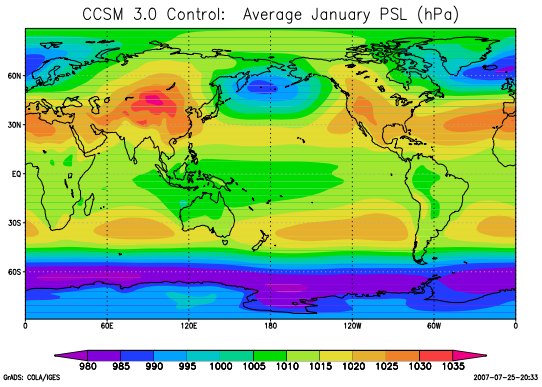

Figure 3. Time-averaged January mean seal-level pressure field from a T85 CCSM 3.0 control simulation.

The first step is gaining access to SpheroiDAL-Tk's class definitions and API:

```
use m_SpheroiDAL
```

Below are declarations of variables instantiating SpheroiDAL-Tk classes that hold the spatial mesh description, and the CartesianField variables for the input PSL values and their zonal averages.

```
type(CartesianGrid) :: LatLonGrid
type(CartesianField) :: psl_LatLon
type(CartesianField) :: psl_ZonAvg
```

The mesh has two coordinates ϕ and λ, discretised on axes stored in one-dimensional arrays `phiVals(1:n_phi)` and `lambdaVals(1:n_lambda)`, respectively. For this particular lat-lon grid, $d\lambda$ is constant for all values of (ϕ, λ) and is thus a scalar.

```
call CartesianGridCreate(LatLonGrid, &
                         'phi:lambda')
call SetAxis(LatLonGrid, 'phi', n_phi, &
             phiVals)
call SetAxis(LatLonGrid, 'lambda', &
             n_lambda, lambdaVals)
call SetWeight(LatLonGrid, 'd_lambda', &
                    'lambda', d_lambda)
```

The CartesianGrid variable `LatLonGrid` is used to create the CartesianField variable `psl_LatLon` that will hold our field data. The field data will be set from a two-dimensional array of values `pslVals` that have been previously initialised.

```
call CartesianFieldCreate(pslField, &
                          LatLonGrid)
call SetFieldData(pslField, pslVals)
```

The zonal average calculation is accomplished by invoking a spatial integration routine, with the result created in the (previously uninitialised) CartesianField output argument `psl_ZonAvg`. The length element $d\lambda$ is extracted from the input CartesianField argument, keyed by the input string token `'d_lambda'`.

```
call SpatialIntegral(psl_LatLon, &
           psl_ZonAvg, 'd_lambda')
TwoPi = 2. * acos(-1.)
call Divide(SurfTempGlobAvg, TwoPi)
```

The resulting field `psl_ZonAvg` is a one-dimensional field containing zonal averages, and its associated CartesianGrid is one-dimensional with one coordinate—the latitude ϕ, and the result is show in Figure 4.

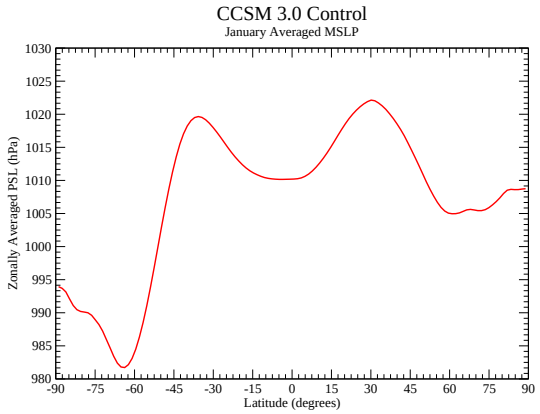

Figure 4. Zonally-averaged mean-sea-level pressure for time-averaged Januaries from a CCSM 3.0 Control simulation.

3.2 GLOBAL AVERAGES

The field $T = T(\phi, \lambda, t)$ varies over the latitude $\phi \in [-90, 90]$, longitude $\lambda \in [-180, 180]$, and $t \in [t_{min}, t_{max}]$—for exmaple, a timeseries of the geographic distribution of the surface temperature field from an atmospheric GCM. The global average is the spatial average in the longitudinal and latitudinal directions, that is $\bar{T}(t) = \frac{1}{4\pi} \int_{-180}^{180} \int_{-90}^{90} T(\phi, \lambda, t) \cos\phi \, d\phi d\lambda$.

The application to compute a timeseries of global averages has the same structure as the zonal average application described above. The differences are in the details of the creation of three-dimensional CartesianGrid and CartesianField objects (as opposed to two-dimensional), and the creation of a cell area Weight object that varies with latitude (as opposed to the scalar $d\lambda$ in the example in the Section 3.1. In addition to the definition of the coordinate axis arrays for (ϕ, λ), the time axis is defined by `tVals(1:n_t)`, and the latitude-dependent cell areas are defined in `d_Area(1:n_phi)`. The salient portions of code are shown below.

```
use m_SpheroiDAL
type(CartesianGrid) :: LatLonTime
type(CartesianField) :: SurfTemp
type(CartesianField) :: SurfTempGlobAvg

call CartesianGridCreate(LatLonGrid, &
              'phi:lambda:t')
call SetAxis(LatLonGrid, 'phi', n_phi, &
             phiVals)
call SetAxis(LatLonGrid, 'lambda', &
             n_lambda, lambdaVals)
call SetAxis(LatLonTime, 't', n_t, tVals)
call SetWeight(LatLonTime, 'd_Area', &
                    'lambda:t', d_Area)
call SpatialIntegral(SurfTemp, &
      SurfTempGlobAvg, &
              'd_Area')
FourPi = 4. * acos(-1.)
call Divide(SurfTempGlobAvg, FourPi)
```

4 CASE STUDY: SCALE SEPARATION

Below we show the results of a case study analysis performed using the SpheroiDAL-Tk programming model. The problem is large-versus-small scale separation in the atmosphere. The traditional approach used by atmospheric dynamicists is *eddy statistics* (Peixoto and Oort [1992]), by which an atmospheric field is expressed in terms of a zonal average and a transient *eddy*, which is the zonal anomaly. We are interested in a multidimensional approach to eddy statistics based on *mean polishing* (Cressie [1993]), through which a multidimensional field is decomposed additively into large-scale effects along each coordinate axis and a local small-scale residual effect.field.

Let $F(\phi, \lambda)$ be a two-dimensional field, where ϕ is the latitude and λ is the longitude. We wish to separate the large and small-scale variation in $F(\phi, \lambda)$. One way to accomplish this is via an additive separation of scales, such as

$$F(\phi, \lambda) = F_{LS}(\phi, \lambda) + F_{SS}(\phi, \lambda), \quad (1)$$

where $F_{LS}(\phi, \lambda)$ and $F_{SS}(\phi, \lambda)$ are the large-scale and small-scale components of F, respectively. A more general and ambitious approach to the problem of scale separation in a two-dimensional field $F(\phi, \lambda)$ is to express F as

$$F(\phi, \lambda) = \mathcal{F}_G + \mathcal{F}_Z(\phi) + \mathcal{F}_L(\lambda) + \mathcal{F}_R(\phi, \lambda), \quad (2)$$

where $\mathcal{F}_G$ is a constant called the *global effect*, $\mathcal{F}_Z(\phi)$ is the *zonal effect*, $\mathcal{F}_L(\lambda)$ is the *longitudinal effect*, and $\mathcal{F}_R(\phi, \lambda)$ is the *residual effect*.

In terms of equation 1, the large-scale field is

$$F_{LS}(\phi, \lambda) = \mathcal{F}_G + \mathcal{F}_Z(\phi) + \mathcal{F}_L(\lambda), \quad (3)$$

and the small-scale field is the residual

$$F_{SS}(\phi, \lambda) = \mathcal{F}_R(\phi, \lambda). \quad (4)$$

Note that the above decomposition is desirable not only because it isolates the small-scale and large-scale effects, but also because the large-scale is also an additive separation of the dependencies on the variables ϕ and λ. Although an additive separation of variables is unusual, it is not unknown. For example, an additive separation of variables is used to solve the Hamilton-Jacobi equation (Goldstein [1980]).

SpheroiDAL-Tk offers a statistical polishing facility capable of performing the aforementioned scale decompositions, and this facility is built on top of compute kernels for integrals over lower-dimensional spaces (in this case over the ϕ- and λ-directions). Polishing occurs through a series of *rounds* comprising computation of a directional average, removal of this average from the two-dimensional field, and accumulation of this quantity in the respective large-scale directional (e.g., $\mathcal{F}_Z(\phi)$ or $\mathcal{F}_L(\lambda)$) or global (i.e.,$\mathcal{F}_G$) effect. This statistical polishing facility is configurable to allow for either mean or median polishing, and can allow the user to choose the ordering of the polishing operations.

For our case study, we examined January monthly-averaged sea-level pressure data from the Community Climate System Model (CCSM3; Collins et al. [2006]) (Figure 3). Mean polishing was performed for multiple rounds on this data, and the results converged rapidly within a few iterations. Resulting in the large-scale field $F_{LS}(\phi, \lambda)$ shown in Figure 5 and the small-scale residual field $F_{SS}(\phi, \lambda)$ shown in Figure 6. The global effect for this analysis was the constant $\mathcal{F}_G = 1007$hPa, and the zonal $\mathcal{F}_Z(\phi)$ and longitudinal $\mathcal{F}_L(\lambda)$ effects are plotted in Figures 7 and 8, respectively. The scale separation is fairly complete, with only weak correlation ($r \approx 0.1$) between $F_{LS}(\phi, \lambda)$ and $F_{SS}(\phi, \lambda)$.

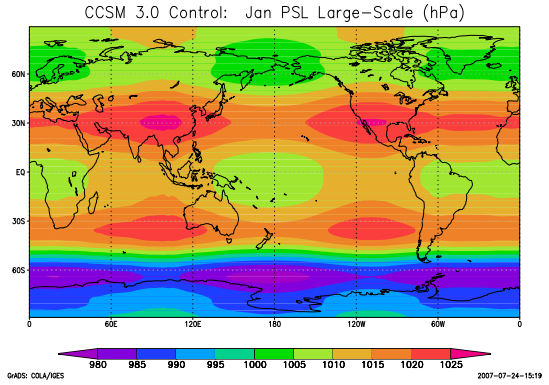

Figure 5. Large-scale PSL field structure computed by SpheroiDAL-Tk statistical polish.

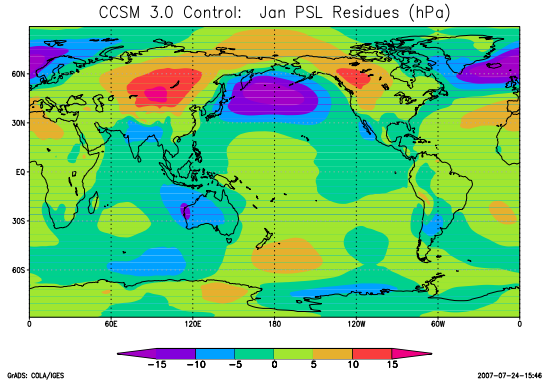

Figure 6. Small-scale "residual" PSL field structure computed by SpheroiDAL-Tk statistical polish.

5 ROADMAP TO PARALLEL PROCESSING

Our chief aim in creating our own object-based data model for SpheroiDAL-Tk was to leave room in the design for implementing parallel processing at a later date. Here we discuss briefly the roadmap to parallel processing for SpheroiDAL-Tk.

Two parallel programming models are currently in

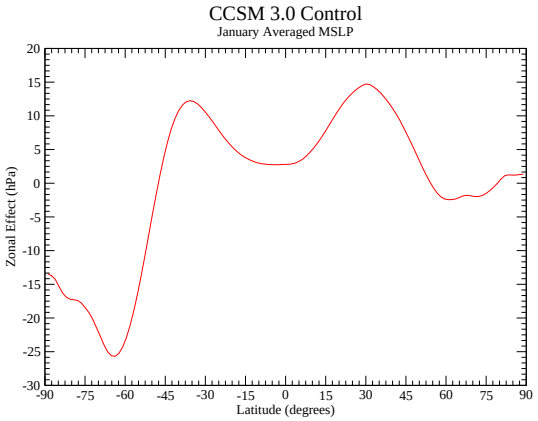

Figure 7. Mean sea-level pressure "zonal effect" computed using SpheroiDAL-Tk mean polish.

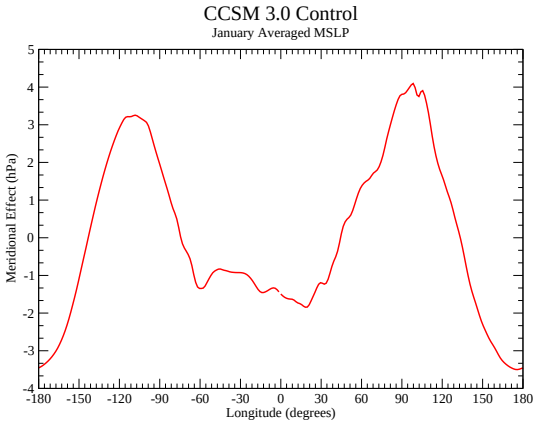

Figure 8. Mean sea-level pressure "meridional effect" computed using SpheroiDAL-Tk mean polish.

wide use in the modelling community: *shared-memory* parallelism and *distributed-memory* parallelism (a.k.a. message-passing). Shared-memory parallelism assumes a single memory address space to which all physical processors have (in theory) equal access (e.g., via a shared memory bus). The dominant portable shared-memory programming model is OpenMP (http://openmp.org), which entails insertion of compiler directives in source code to implement loop-level parallelism. OpenMP has the advantage of simplicity of implementation, but has the weakness of lack of strong control over load balance, and thus typically offers limited scalability. OpenMP has been identified as the emerging programming model for multicore processors. The dominant portable distributed-memory programming model is the Message Passing Interface (MPI; http://www.mcs.anl.gov/mpi/). MPI offers its users fine-grain control over load balance, as well as a number of communication models (i.e., blocking vs. non-blocking) to hide communications costs behind computation. MPI is scalable to large numbers of processors. The chief weakness of MPI is the difficulty in implementing a communications-based

programming model. Because of the complementatry nature of the OpenMP and MPI programming models, another parallel programming model—*hybrid parallelism*—has emerged that incorporates OpenMP within shared-memory nodes and MPI message-passing between physically distinct nodes.

The SpheroiDAL-Tk data model and compute kernels will first be parallelised using OpenMP. This will leave the data model in Figure 2 unaffected, but will require instrumentation of the SpheroiDAL-Tk compute kernel routines with OpenMP directives to achieve do-loop-level parallelism. We are motivated to do this first to provide near-term relief for applications so they can exploit multicore processors, which are becoming the norm for desktop and laptop computers.

Distributed-memory (MPI) parallelism will be necessary for SpheroiDAL-Tk to handle the petabyte-scale data volumes we will soon see coming from long high-resolution climate model integrations. Implementation of MPI parallelism will require extension of the SpheroiDAL-Tk data model shown in Figure 2 to include description of the *domain decomposition* of the CartesianGrid and CartesianField classes. We will almost certainly adopt a purely Cartesian data decomposition (a.k.a. "checkerboard") to achieve this aim, confident that most of the analyses the package performs will have fairly uniform load balance characteristics. The SpheroiDAL-Tk compute kernals will also have to be altered to incorporate message passing across dimensions over which an analysis is being performed.

Notice in all of the above discussion that the parallelism will be implemented *inside* of SpheroiDAL-Tk. That is, the programming examples cited throughout this paper will remain unchanged.

6 CONCLUSIONS AND FUTURE WORK

Climate modelling is entering an exciting era in which the size of model output datasets will explode, reaching and exceeding the petabyte scale. The computational toolset currently available to analyse these data are inadequate to surmount the levels of computational complexity required, and parallel programming will be needed. Meanwhile, the level of sophistication in spatiotemporal data analyses is increasing, which will also require a more general toolset.

We have designed a package called SpheroiDAL-Tk to meet both of these emerging requirements. We have prototyped both a data model and set of compute kernels that are applicable to spatiotemporal data analysis of logically Cartesian data in curvilinear coordiantes. The design results in a scientist-friendly programmming model, which we have demonstrated with both standard examples and a more complex scale separation case study.

Areas for future development work include: creation of an open-source, release-ready version of SpheroiDAL-Tk; creation of multilingual programming interfaces (e.g., Python); development of a shared-memory parallel implementation to exploit multicore processors; and inclusion of message-passing parallelism to support analysis of very large datasets. The long-term result will be a performance-portable package capable of running on workstations, PCs, and cluster-based supercomputers.

ACKNOWLEDGMENTS

This work is primarily supported by the United States Department of Energy's Scientific Discovery through Advanced Computing (SciDAC) program. The ANU Supercomputer Facility is supported in part by the Australian Department of Education, Science, and Training.

REFERENCES

References

Collins, W. D., C. M. Bitz, M. L. Blackmon, G. B. Bonan, C. S. Bretherton, J. A. Carton, P. Chang, S. C. Doney, J. J. Hack, T. B. Henderson, J. T. Kiehl, W. G. Large, D. S. McKenna, B. D. Santer, and R. D. Smith. The Community Climate System Model: CCSM3. *Journal of Climate*, 19(11):2122–2143, 2006.

Cressie, N. A. C. *Statistics for Spatial Data*. Wiley Interscience, New York, Revised edition, 1993.

Decyk, V. K., C. D. Norton, and B. K. Syzmanski. Expressing object-oriented concepts in fortran90. *ACM Fortran Forum*, 16(1):13–18, 1997.

Goldstein, H. *Classical Mechanics*. Addison-Wesley, Reading Mass., second edition, 1980.

Ong, E. T., J. W. Larson, B. Norris, R. L. Jacob, M. Tobis, and M. Steder. Multilingual interfaces for parallel coupling in multiphysics and multiscale systems. In Shi, Y., van Albada, G., Dongarra, J., and Sloot, P., editors, *Proceedings of the Seventh International Conference on Computational Science (ICCS 2007)*, volume 4487 of *Lecture Notes in Computer Science*, pages 931–938, Berlin, 2007. Springer-Verlag.

Peixoto, J. P. and A. H. Oort. *Physics of Climate*. American Institute of Physics, New York, 1992.