(NASA-CR-195971) PARALLELIZATION
AND VISUAL ANALYSIS OF
MULTIDIMENSIONAL FIELDS:
APPLICATION TO OZONE PRODUCTION,
DESTRUCTION, AND TRANSPORT IN THREE
DIMENSIONS Annual Progress Report
(Georgia Inst. of Tech.) 31 p

N95-12798

Unclas

G3/45 0027546

Annual Progress Report:
Parallelization and Visual Analysis of Multidimensional Fields: Application to Ozone Production, Destruction, and Transport in Three Dimensions
Grant No. NAGW-3886

Karsten Schwan

Oct. 1994

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332
e-mail: schwan@cc.gatech.edu
fax: 404-853-9378

/ N7C/////
/N-4S-CR
O C/T.
J 47846
P- 31

1   Introduction

List of investigators:

> Karsten Schwan, College of Computing
> Fred Alyea, Earth and Atmospheric Sciences
> M. William Ribarsky, Information Technology
> Mary Trauner, Information Technology
> Greg Eisenhauer, student, College of Computing
> Yves Jean, student, College of Computing

List of associated investigators (not funded by NASA):

> Thomas Kindler, student, Earth and Atmospheric Sciences
> Dilma Silva, student, College of Computing
> Jeffrey Vetter, student, College of Computing (funded by separate NASA GSRP grant).

This brief report is being submitted for review by Glenn Mucklow, project manager for NASA grant NAGW-3886. The report contains a brief outline of our progress to date, followed by a copy of a draft of a research paper describing some details of our work, about to be submitted for publication. A previous report (submitted July 1994) already included additional research papers produced from related research, but not directly funded by ARPA. Materials sent in this report are also available via the Internet on Mosaic at:

> http://www/gatech/edu/scivis/projects/projects.html

2 Progress to date

This project was initiated approximately 8 months ago. We have spent these months performing the following tasks:

- **Task 1:** Parallelization of Atmospheric Modeling Code:

  - Preparation of standard FORTRAN code version for parallelization; conversion of major code modules from Fortran to C for enhanced portability of code across the SUN Sparc, SGI, and KSR platforms used by our group: in progress, with next steps concerning code parallelization for non-shared memory platforms, including the new IBM SP-2 machine being acquired by Georgia Tech and to be installed Dec. 1994.

  - Parallelization results attained with earlier prototype of modeling code, which has now also been validated: first publication is now ready for submission (draft included with this report); ongoing work concerns additional performance evaluation, porting to non-shared memory architectures, and making this code interactive (see below).

  - Design and implementation are also underway for the on-line monitoring of distributed target machines.

- **Task 2:** On-line program monitoring and steering:

  - An on-line monitoring library has been completed with the CThreads parallel programming library and is now available via Internet FTP; a paper, test programs and CThreads graphical performance views can be viewed interactively via mosaic at "http://www.gatech.edu/ steering/html".

  - design of support for on-line steering has been completed, initial implementation has also been completed; we are now instrumenting the parallel atmospheric modeling code for on-line steering and monitoring; the purpose of such steering will be to permit atmospheric modelers to play 'what if' games with their code (e.g., what if chemical concentrations have certain values? what if the following methods are used for vertical transport?).

  - On-line monitoring and steering concerning the atmospheric modeling code are also addressing performance monitoring and on-line code configuration for performance improvement (e.g., load balancing), which is a significant problem with many large-scale parallel codes.

  - Implementation is underway for construction of a benchmark program which uses synthetic workloads for evaluation of network and I/O loads imposed by remote visualization/monitoring/steering.

  - The atmospheric modeling code is now being made on-line, implying the on-line retrieval of actual satellite observations from both NASA and UARS data sets, the on-line use of those observations in the model, and the subsequent on-line output and visualization of model output, in the future to be done side-by-side with visualizations of observational data.

- **Task 3:** on-line visualization and animation:

- The SGI Explorer environment is being used for visualization of the data sets produced and consumed by atmospheric modeling codes: data has been imported into the SGI environment and data sets output by the atmospheric models are fully documented; additional work is addressing more useful visualizations based on the common graphics libraries used by atmospheric researchers in the SGI environment.

- Software integration: development of a programming library for exchange of complex binary files between different machines and between Fortran and C in order to make future data exchanges across programs and machines easier: the library uses self-describing file formats and has been made available via the Internet to other researchers; as a result, visualization processing can be performed on any machine attached via network to the parallel machine generating output data.

- Development of basic tools part of the Glyphmaker environment, to facilitate future visualizations useful for atmospheric modeling codes; import of atmospheric modeling data from SGI Explorer to the Glyphmaker environment has been completed. In addition, Glyphmaker is being ported out of the Explorer environment for increased speed and interactivity.

- Implementation of Glyphmaker-based visualizations specifically useful for the atmospheric modeling environment.

- Use of Motif-based animations for performance visualization and for program steering of parallel threads programs: initial results can be viewed on mosaic in conjunction with the monitoring work mentioned above.

A Parallel Spectral Model for Atmospheric Transport Processes

Thomas Kindler[1,2], Karsten Schwan[3], Dilma Silva[3], Mary Trauner[3], Fred Alyea[1]

[1]School of Earth and Atmospheric Sciences, Georgia Institute of Technology,
Atlanta, Georgia 30332
[2]Paul Scherrer Institute, CH-5232 Villigen, Switzerland
[3]College of Computing, Georgia Institute of Technology, Atlanta, Georgia 30332

DRAFT

October 10, 1994

1 Introduction

Atmospheric modeling is a grand challenge problem for several reasons, including its inordinate computational requirements and its generation of large amounts of data concurrent with its use of very large data sets derived from measurement instruments like satellites. In addition, atmospheric models are typically run several times, on new data sets or to re-process existing data sets, to investigate or re-investigate specific chemical or physical processes occurring in the earth's atmosphere, to understand model fidelity with respect to observational data, or simply, to experiment with specific model parameters or components.

Our group's contributions to the areas of atmospheric modeling and high performance computing are:

- *Parallel model execution* – we demonstrate the opportunities for parallelism in a global atmospheric modeling code using the spectral solution method, while also evaluating several alternative methods for performing such parallelization.

- *Interactive model execution and steering* – we pursue model parallelization not simply to speed up model execution, but also to explore the use of parallel machines for on-line model execution and for steering model computations so that end users can easily experiment with alternative model parameters, conveniently evaluate and re-evaluate the behavior of specific processes being modeled, and affect or change model execution to improve performance.

Our approach to on-line model interaction and steering differs from what has already been found useful for understanding stored model data, where researchers seek to steer their data visualizations to explore different data domains or even to directly control their virtual reality simulations of such data. We explore programmer interactions with the running models generating such data sets, including giving programmers the ability to direct model execution, to influence the data generated and the computations performed by such models, and to improve model execution performance.

Evidence of the utility of on-line program steering can be found in numerous past publications, many of which are reviewed in [GVS94].

The specific global atmospheric model implemented as part of this research represents atmospheric fields with spherical basis functions, which have been widely used for modeling phenomena like weather prediction, global warming or global change of atmospheric constituents [DS89, HBB+92]. Spectral models have some advantages over the grid-based models commonly parallelized in past and current work[WAB+93, ABB+94, BK90, JH89]. For example, spectral models naturally conserve the area averaged mean square kinetic energy and the mean square vorticity of wind fields, whereas in grid based models these quantities are either not conserved or require additional computation when such conservation is important[WP86]. Despite such advantages, grid based models have found wider acceptance in recent research in part because they are believed (1) to give rise to larger amounts of parallelism than spectral models, and (2) more easily coupled with grid based models simulating local phenomena (e.g., pollution modeling).

This paper demonstrates that a spectral global transport model can be parallelized quite efficiently, and that this parallelization can be performed such that the resulting parallel code will perform well even on large-scale parallel machines (ie., machines with a thousand processors). These levels of parallelism are achieved by using an alternative parallelization to the one used in previous work on parallel spectral transport models[BK90], where parallelism is attained by decomposing atmospheric data along latitudes and/or longitudes. In comparison, our approach also takes advantage of the relative independence of computations at different levels in the earth's atmosphere, resulting in parallelism of up to 40 processors, each independently performing computations for different atmospheric levels and requiring few communications between different levels across model time steps. Next, additional parallelism is attained within each level by taking advantage of the natural parallelism offered by the spectral computations being performed (eg., taking advantage of independently computable terms in equations).

Our parallelization strategy and results also differ from the recent, extensive work on parallel climate models by Foster and Worley[FW94, WF94], most of which was performed concurrently with our research. Specifically, Foster and Worley investigate message passing machines like the Intel Paragon, while our work includes a detailed study of performance overheads arising on shared memory multiprocessors. Parallelism is again attained by use of longitudinal and/or latitudinal domain decomposition, and then further increased by also parallelizing specific model computations within different domains (ie., by parallelizing the FFT computations required within each columnar atmospheric patch). This approach somewhat resembles our own parallelization strategy, in which parallelism attained from level-based data decomposition is increased by concurrent computation of specific calculations within each atmospheric level (e.g., term parallelization).

The specific model parallelized in our research is the transport component of climate models. This enables us to study in detail the performance overheads arising from this component's parallelization. Specifically, our transport model simulates the transport of atmospheric constituents by expanding the used fields in spherical basis functions and then solving the governing differential equation with a spectral approach. As a result, one specific issue addressed by our work is the efficient implementation, representation, and sharing of the global spectral information shared by all of the spectral model's computations, while the model's grid-based data is statically decomposed across different processors' memory units. Such spectral data consists of up to several hundred complex spectral coordinates per level (253 coordinates for 21 wave resolution, which corresponds to a horizontal resolution of 6.4 degrees by 6.4 degrees, and 946 coordinates for 42 wave resolution which corresponds to a vertical resolution of 2.8 degrees by 2.8 degrees). One reason spectral models were believed difficult to parallelize is that such data must be shared across all processors of the machine involved in spectral computations. The experimental results shown below demon-

2

strate that such sharing involves only small communication overheads, especially when compared to sharing any amounts of grid-based data also used within transport and climate models. Such grid data consists of up to several thousand complex numbers per level (1024 numbers for 21 wave resolution and 8192 numbers for 42 wave resolution). Model computations require that grid data is shared among neighboring gridpoints.

An extension of the transport model described in future publications will also address chemical phenomena. This extension is expected to further improve parallel program performance, which prompts us to expect equally good or even improved performance for more complex atmospheric chemistry models based on the spectral transport method investigated in our research.

The target machines used in our research are networked collections of parallel supercomputers and workstations jointly performing model computations, data processing and storage, on-line data visualization, and on-line program monitoring and steering. Our current research is targeting local area networked machines linked to the Internet, with future extensions of this research addressing communication protocol issues in high performance wide area networks. Our current experimental testbed consists of a 64-node KSR-2 shared memory supercomputer coupled with Silicon Graphics visualization engines and IBM RS6000-based workstations used for additional model computations. Socket-based network communications are employed for efficient remote visualization processing, while PVM will be used for the execution of the multi-machine, heterogeneous atmospheric models to be addressed by our future work.

In the remainder of this paper, we first describe the basic functionality of the spectral-based global transport model, followed by a brief validation of its correctness from first principles and in comparison to a Fortran-based version of the model in current use for atmospheric modeling research. In addition, the results of a model simulation af a simple, known species (Carbon-14) is compared to observations. In Section 2.3, we describe model parallelization and its implementation on the KSR-1 and KSR-2 supercomputers. Next, detailed performance measurements demonstrate the opportunities for parallelism at different levels of the model, followed by an evaluation of costs arising from the sharing of spectral data across different processors. Locality of access to data and code and therefore, model performance is shown high even for fairly small models and data sets. Then several on-line strategies for steering model execution coupled with the visualization of model output data are discussed and shown useful to researchers in Earth and Atmospheric Sciences. Conclusions and future research addressing model parallelization, steering, and visualization appear in the last Section.

2 Model Functionality

2.1 The Modeling of Atmospheric Transport Processes

Global transport models are important tools for understanding the distribution of relevant atmospheric parameters like the mixing ratios of chemical species and aerosol particles [RTBW94]. Transport models are often coupled with a variety of chemical reaction mechanisms to describe selected chemical changes of the simulated species during transport. In addition, global transport models can be coupled with local models for a variety of purposes, including the provision of input data to the global model generated by outputs of local air pollution models. The purpose of the transport model developed as part of this research is the investigation of parallelism in transport model execution. In addition, this model is used to answer scientific questions concerning the stratospheric-tropospheric exchange mechanism or the distribution of species like Fluorocarbons (CFC's), Fluorohydrocarbons (CFHC's) or Ozone.

The model's functionality outlined below focuses on the spectral method for modeling atmospheric transport processes. It is elaborated only to the extent necessary for motivating the model's computational needs and data requirements explained further in Section 2.3. Chemical reaction mechanisms extending the transport model will be described in future publications. The mathematical formulation of the model is explained briefly in Appendix A and is described in more detail in several other publications, including [Hau40, Sil54, Pla60, KHYK61, WP86].

The Spectral Approach to Solving Global Transport. Any atmospheric constituent Y with mixing ratio X^1 is subject to a continuity equation involving the constituent. This equation relates the constituents mixing ratio at each point in space with wind velocity and direction by keeping the total mass constant. The continuity equation as well as any expansion to it (e.g. diffusion, chemistry) is usually expanded into spherical coordinates $\lambda = $ longitude, $\mu = sin(\phi)$ ($\phi = $ latitude) and is henceforth called the 'transport equation' in this paper (see equation 7). The spectral solution method for the transport equation takes advantage of the fact that any variable $F(\lambda, \mu, t)$ in a 2 dimensional spherical surface can be approximated by an expansion into a set of orthogonal spherical basis functions, called spherical harmonics. Furthermore, terms like $\frac{\partial F}{\partial \mu}$ and $\frac{\partial F}{\partial \lambda}$ are more easily and often more accurately calculated in the spectral domain, compared to solution methods employing grid-based finite difference schemes. The vertical part of the transport equation on the other hand is commonly calculated by using finite difference methods.

Programming Atmospheric Transport with Spectral Methods. A typical algorithm for solving the transport equation using spectral methods consists of the following steps:

1. Transformation of the initial conditions for species Y into the spectral from the grid domain.

2. Calculation of the derivations $\frac{\partial}{\partial \mu}$ and $\frac{\partial}{\partial \lambda}$ on the right hand side of the Equation 7 (see Appendix A), for all variables.

3. Transformation of the derivatives of all variables into the grid domain.

4. Calculation of the nonlinear products of the derivations in the grid domain.

5. Transformation of all grid results back into spectral domain, and calculation of remaining summations of the different terms.

6. Calculation of the time integration in the spectral domain.

7. Return to step 2 for next time step.

Transport models require that the transport equation for species Y is integrated for each time step for all gridpoints at every level of the atmosphere. Current transport models us up to 42 spectral waves ($lMax = 42$ which corresponds to 946 spectral points, see Appendix A for explanation) per model layer [HBB$^+$92], which results in a resolution of about 128 longitude and 64 latitude gridpoints. The vertical resolutions used in most models assumes between 10 and 40 levels which corresponds to an altitude of about 50 km. Temporal time steps of about 15 minutes are necessary to maintain numerical stability for the model calculations. A general overview of the performance of several different kinds of transport models appears in [Pra92].

---

[1]Concentrations of atmospheric trace gases are usually expressed as a ratio between the number of molecules of species Y and the number of air molecules in a given volume. This is called the mixing ratio of species Y, and is usually given in parts per million (ppm), parts per billion (ppb), or parts per trillion (ppt).
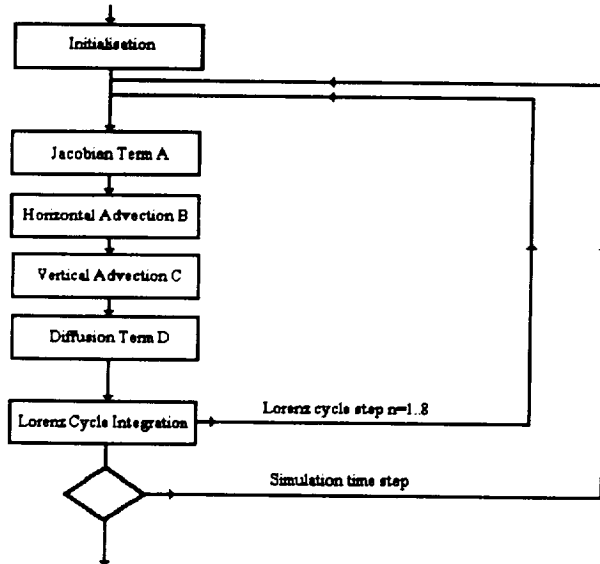
Figure 1: Flow diagram of the sequential versions of STRAT, TRANS21 and TRANS42.

2.2 The Atmospheric Transport Code

The research presented in this paper is based on three codes: 1) STRAT: a sub-program of a global circulation model described in [CAPP75, CAP80]. This code is originally written in Fortran, with a newer version also available in C. STRAT runs on workstations like the IBM RS-6000 machines. 2) TRANS21: a transport code which produces identical results as STRAT but written in C and the Cthreads parallel programming library[SFG$^+$91]. This code is a prototype used for exploration of parallelism in atmospheric modeling and will next be employed for certain scientific investigations, including studies of the global cycle of Nitrous Oxide (N_2O). 3) TRANS42: a version of TRANS21 employing a higher resolution in the horizontal direction. In the remeining part of the paper we will refer to either one of TRANS21 or TRANS42 simply by TRANS.

Instead of computing the windfields inside the model, all codes use observed UKMO windfields [SO93], which are concurrently read from files while the simulation proceeds. The models contain 37 vertical levels starting at the surface (1000mbar) and going up to about 48 km (1mbar). A triangular spectral truncation with $lMax = 21$ is used for TRANS21 (T21, which corresponds to 253 complex spectral data points) to approximate the fields given in the grid domain with 64 longitudinal and 28 latitudinal coordinates. In TRANS42 the truncation is $lMax = 21$ (T42, which corresponds to 946 complex spectral data points) which corresponds to 128 longitudinal and 64 latitudinal coordinates in the grid domain. The stepwise numerical time integration is calculated with an 8-cycle Lorenz-scheme [Lor71] 12 times per day, which leads to an overall time step size of 15 minutes. Therefore, each transformation (the most time consuming part in a spectral model) has to be performed 96 times per day, including the execution of 13 transformations in total in each layer, resulting in the computation of 1248 transformations per day and layer. A flow diagram of the sequential code implementing this model appears in Figure 1.

2.3 Parallelization of the Spectral Transport Code

The TRANS transport codes are parallelized in three ways, which are described and motivated next:

5

Term A   Term A   Term A

Term B   Term B   Term B

Term C   Term C   Term C

Term D   Term D   Term D

Lorenz Integration   Lorenz Integration   Lorenz Integration

Term A   Term A   Term A

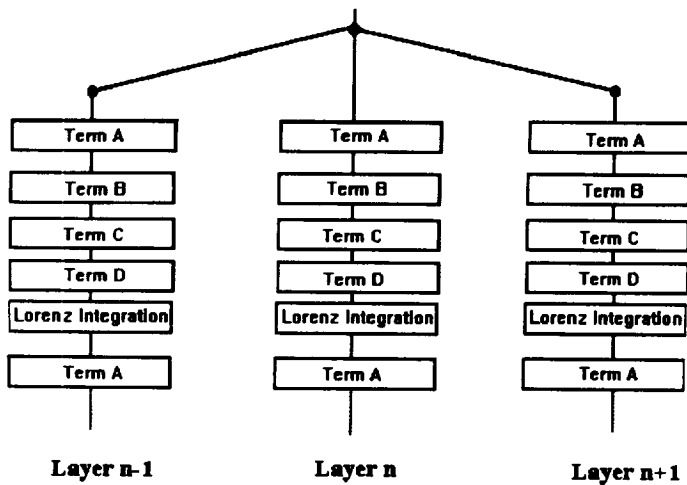**Layer n-1**   **Layer n**   **Layer n+1**

Figure 2: Schematic representation of layer parallelization. The dots represent processors and the lines represent spectral data exchange.

1. *layer parallelization*: each atmospheric layer is computed separately,

2. *term parallelization*: terms A, B, C, and in Equation 7 are calculated independently, and

3. *μ parallelization*: μ loops (e.g., inside the transformations from grid to spectral and spectral to grid) are distributed across multiple processors.

Given these parallelization strategies, the major data structures within the TRANS code to be decomposed across the parallel machine's different processors are:

1. *grid layer data*: data relevant only to a certain layer or term in the grid domain,

2. *spectral layer data*: data relevant only to a certain layer or term in the spectral domain, and

3. *common data*: data identical for all layers (e.g., Legendre functions for transformation).

Common data is replicated across all involved processors and is therefore, locally accessibly. Spectral layer data is shared by all processors dealing with a certain layer. The grid layer data is decomposed along constant μ's and accessed locally by the processor to which this range of μ's has been assigned. As a result, no movement of grid data is necessary during model computation, whereas spectral data is shared frequently.

2.3.1 Layer Parallelization

Grid based global models are typically decomposed (e.g., see [WAB+93]) using a two-dimensional latitude/longitude domain decomposition, where each subdomain consists of several neighboring vertical columns extending from the earth's surface to the top layer of the atmosphere addressed by the model. This decomposition is used for a variety of reasons, including simulation of only few model layers or attempts to simulate processes that are strongly coupled in the vertical
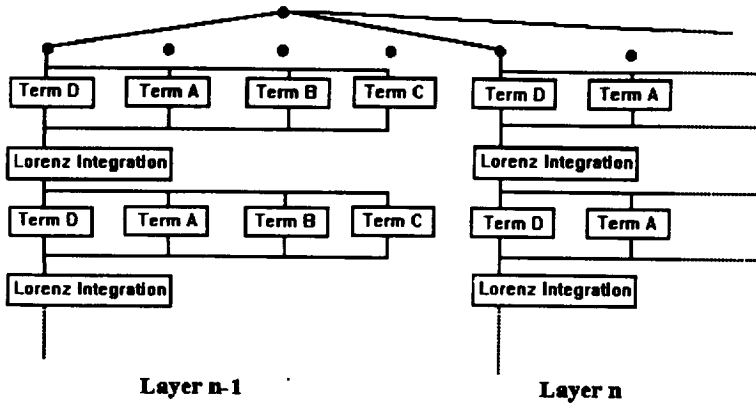
Figure 3: Schematic representation of term parallelization. The dots represent processors and the lines represent spectral data exchange.

dimension. The specific atmospheric processes investigated by our group do not exhibit a strong vertical coupling. Furthermore, when using a spectral transport model, vertical coupling is small, whereas the coupling and therefore, data sharing in the horizontal direction is extensive. This suggests that a layer-based parallelization is a first appropriate parallelization step for TRANS21 and TRANS42.

A schematic description of layer parallelization applied to TRANS21 appears in Figure 2. Here, a single computational thread performs all necessary calculations for an entire layer, with each layer thread mapped to its own processor. Each such processor contains a copy of all data and thread code, and it also contains the entire layer's grid and spectral data. Exchange of information between layers is necessary only for neighboring layers for terms C and D in Equation 7 (vertical derivations). Layer processors also compute the time step integration. Synchronization between threads is necessary only before each time step integration, in order to ensure that all layers use updated information before taking each time step. This barrier-like thread synchronization is programmed using multiple mutex locks, resulting in some small communication overhead between layer processors and therefore, resulting in almost linear speedup and high computational efficiency, as demonstrated by the measurements in Section 3.3 below.

2.3.2 Term Parallelization

Term parallelization exploits the fact that for each layer, the terms A, B, C, and D in Equation 7 can be calculated independently and concurrently, resulting in a parallel program described schematically in Figure 3. The implementation of this parallelization exploits the ability of the Cthreads parallel programming library used in this research to dynamically create and delete computational threads with comparatively low overheads. Such threads, henceforth called *helper threads*, are executed on additional processors not currently used in layer computations. An additional optimization used in our implementation is to pre-create helper threads, pre-map them to the appropriate processors, then simply wake up such threads when they are needed, using the low-level conditional wait and signal primitives offered by the Cthreads library[Muk91, GMS94]. Such wakeups are per-
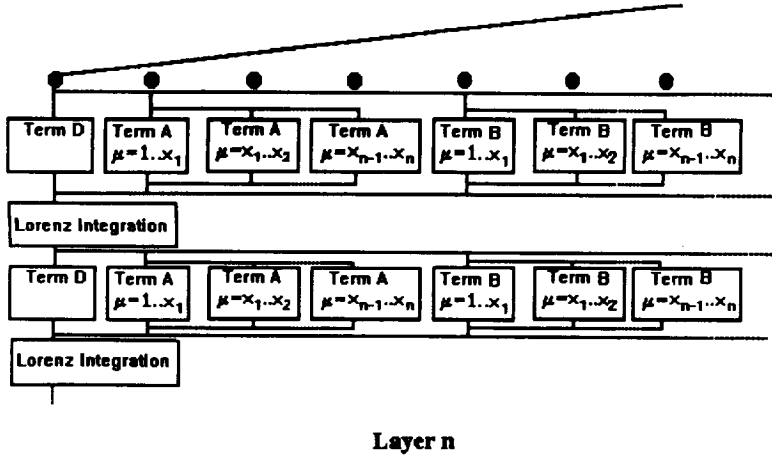
Figure 4: Schematic representation of μ parallelization. The dots represent processors and the lines represent spectral data exchange.

formed whenever layer processors have reached the point in time when terms A, B, and C have to be computed. Layer processors compute the term D themselves, while waiting for the completion of helper threads computing A, B, and C.

The time integration of the transport equation implies the use of spectral information from all terms involved, which therefore, results in the exchange of spectral data among processors during time integration calculations. Furthermore, since term computations differ in length, the computational loads of helper threads are not balanced. Measurements shown in Section 3.3 demonstrate the relatively low cost of spectral data exchange. It also shows that the load balancing problem should be addressed in order to attain significant speedups on large-scale parallel machines. Our future work will demonstrate that the addition of model computations involving constituent chemistries can lead to more balanced computational loads when using helper threads.

2.3.3 μ Parallelization

The most time consuming steps in terms A,B and C of Equation 7 are the various transformations between spectral and grid domain performed as part of these computations. In these transformations and also between different transformations, computations must be performed over all values of μ (latitudes) (see the Appendice A for more detail). A third level of parallelization employed in the TRANS codes is the decomposition of each term's computation into multiple computations, each addressing a different region of μ values. A schematic representation of this parallelization appears in Figure 4. This parallelization strategy possesses the potential for the most significant level of parallelization in our chosen approach. This strategy is also chosen in most other attempts to parallelize spectral models (see Section 3.7). Theoretically, μ calculations can be spread over the same number of processors as there are latitudes used in the model (actually, most transport models explicitly compute values for only half the number of total latitudes, because corresponding latitudes in the northern and southern hemispheres can be transformed together). As shown by experimental evaluation in Section 3.3, excessively small regions of μ values result in increased communication overheads that prevent increases in speedup even on the fairly tightly coupled KSR

8

shared memory multiprocessors employed in our research. However, the use of spectral models with higher resolution increase the usefulness of this parallelization strategy due to resulting increases in the lengths of term computations.

2.4 Implementation Characteristics

Two characteristics of the spectral model's implementation not mentioned earlier concern program portability, extensibility, and its broader use within a project addressing interactive parallel codes.

The TRANS models' implementation is portable to any shared memory parallel machine, currently including SUN Sparcstations, SGI uni- and multiprocessors, and the KSR-1 and KSR-2 supercomputers. This portability is achieved by use of the Cthreads portable parallel programming library described in detail in several publications, including [Muk91, GMS94]. This library hides underlying differences in parallel machines and operating systems by providing a standard set of constructs for creating and controlling parallel execution threads. For example, for thread synchronization, mutex locks offered by the library are layered directly on low-level synchronization hardware offered by the KSR-2 supercomputer, but must employ the operating system-provided 'lock' synchronization constructs existing on the SGI machine. As a result, the performance of mutex locks differs across both machines, but the interface and functionality provided to application programmers does not change. Model portability to non-shared memory machines is a topic of research that is addressed in other work by our research group (e.g., see [GMSS94, CMS93, ES94]).

Also important is the model's extensibility to include additional code modules, such as modules performing chemical calculations for specific constituents. While the implementation of TRANS does not offer a uniform or self-contained framework for inclusion of additional or complementary code modules as described in other research[FW94, ES94], TRANS attains limited extensibility and more importantly, the ability to interact on-line with other programs potentially running on different machines by using a uniform format for exchange of binary input and output files. This format is described in detail in [Eis94]. Briefly, it permits programmers to define the formats of expected input and output data used by their programs, then provides support for translation of these formats across the language and machine barriers existing between those programs. Based on this format, the TRANS model's implementation currently shares its input and output files between different machines (currently including SGI machines, SUN Sparcstations, IBM RS/6000, and the KSR-1 and KSR-2 machines) and different languages (currently including FORTRAN and C programs). Such sharing is important for several reasons. First, TRANS input files may consist of synthetic or observational input data, the latter resulting from the UKMO satellite data sets preprocessed by Fortran 'cleanup' and data interpolation programs. Second, TRANS output is either stored, post-processed for interpolation to different grid sizes and stored, or post-processed and then input to on-line data visualizations currently employing SGI Explorer-based visualization tools. In addition, selected program attributes are directly captured by the Falcon on-line monitoring system described in more detail in [EGSM94]. The purpose of such on-line monitoring and data visualization is to permit programmers to inspect selected program output and even individual program variables, based on which they can then 'steer' the program's execution toward more useful data domains, experimental outcomes, play 'what if' games, and understand program behavior with different values for the behavior and concentrations of atmospheric constituents. If such experimentation is possible during program execution, the current excessively long cycle from data input, to model computation, to data output and display can be reduced in time, thereby permitting experimental scientists to avoid time-consuming mistakes or simply to evaluate more alternatives in their scientific endeavors.
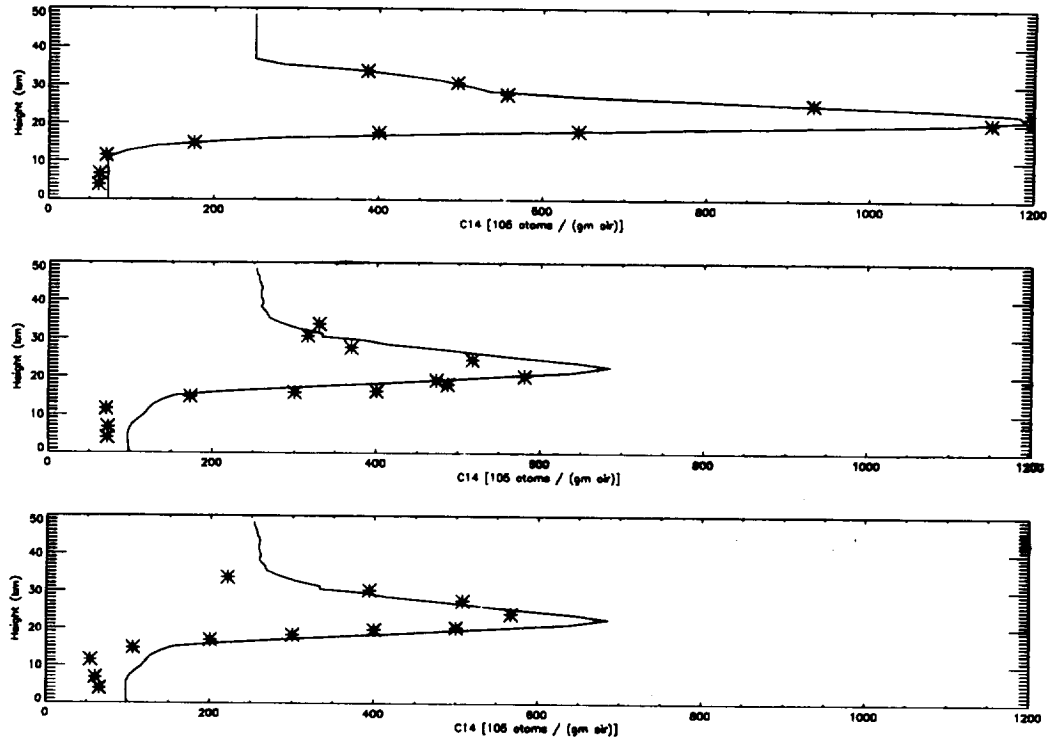
Figure 5: Carbon 14 distribution at 70°N for October 1963 (top), April 1964 (middle) and October 1964. The solid lines represent model results and the asterisks represent measurements given by Johnston. Note that observed windfields from 1993/94 are used for the simulations. Therefore, model results from 'April 1963' and 'October 1964' in the figure correspond to model output generated 6 months and 12 month after model initialization derived with windfields from 1993/94, respectively.

3 Model and Implementation Evaluation

3.1 Model Validation

The parallelized transport model (TRANS) has been validated and compared with the sequential Fortran version of the same model, called STRAT, which is described in detail in [CAPP75, CAP80]. For further validation, we have also simulated the distribution of radioactive Carbon 14 ^{14}C in the atmosphere after the nuclear bomb tests in the 1950s and 1960s. A detailed outline of this experiment appears in [RTBW94]. Since natural sources and sinks of ^{14}C are negligible, simulation of the excess ^{14}C after the bomb tests and comparison with observational data provides a reliable test of the TRANS model's ability to transport material correctly. For this validation, input data is taken from Johnston [Joh89] for October 1963. This input data is shown in conjunction with latitudinally averaged profiles of observations and TRANS model calculations in Figure 5 for April 1964 and October 1965.

3.2 Model Performance Compared to Sequential Codes

For normative comparison, the performance of the C version of TRANS21 is compared with the Fortran version of STRAT for a three day run in Table 1. Measurements are shown for a Silicon

Machine	version	time seconds
RS6000	STRAT	518
SGI	STRAT	662
SGI	TRANS21	535
KSR1	TRANS21	1511
KSR2	TRANS21	780

Table 1: Sequential execution times of model versions on different machines, for 37 levels and 3 days simulation time.

Graphics INDIGO2 machine, an IBM RS/6000 workstation, and individual processors of the KSR1 and KSR2 multiprocessors employed in our research. These results demonstrate that the performance of the TRANS model's implementation with Cthreads exceeds the performance of STRAT on the SGI machines (probably due to higher efficiency in the C code implementation), whereas TRANS performance on the KSR machine's comparatively slower processors is less than that of the STRAT model on either the SGI or RS6000 machines.

3.3 Performance of the Parallel TRANS Model

3.3.1 Issues in Model Parallelization

The experimental evaluations presented in this section address several specific issues concerning parallel atmospheric modeling:

- Do spectral solution methods for transport models result in computational overheads making them ill-suited for parallelization? More specifically, what are the performance effects of data movement during spectral model computations on modern multiprocessor machines?

- Can parallelism attained by layer, term, and μ parallelization be scaled to large parallel machines? This typically requires the attainment of sufficient locality of access to model code and data.

- What are the I/O demands of atmospheric model computations based on spectral solution methods, when acquiring input data on-line from file-based satellite data, when producing and displaying output data during model computation, while also using the restart files commonly used in large-scale scientific codes?

3.3.2 Experimental Results

All measurements shown in this section are attained by running the TRANS21 or TRANS42 programs for two simulation days. As described in Section 2.2, this corresponds to 2496 transformations from spectral to grid or grid to spectral representations for each layer. The simulations are performed with a varying number of layers in order to adjust the model's parallelism to the size of the underlying KSR machine.

Measurements are performed on the KSR2 supercomputer, which is a shared memory, cache-only (COMA) architecture with an interconnection network that consists of hierarchically interconnected rings, each of which can support up to 32 nodes or 34 rings (the largest machine delivered to date consists of 256 processors). Each node consists of a 64-bit processor, 32 MBytes of main memory used as a local cache, a higher performance 0.5 MBytes sub-cache, and a ring interface. CPU
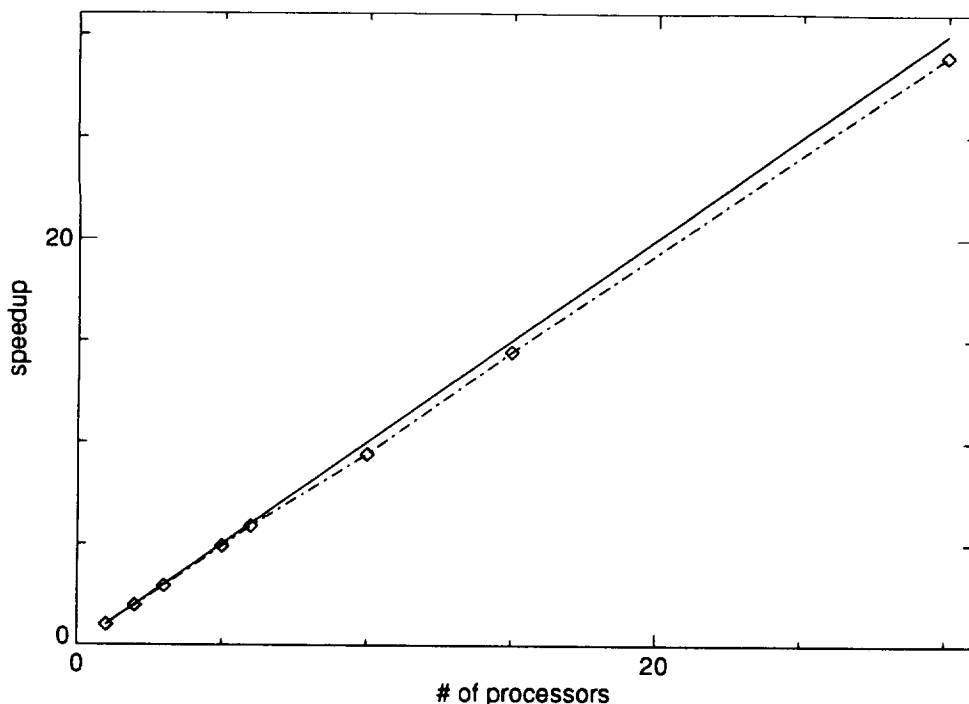
Figure 6: Measured speedup (dashed line with diamonds) for layer parallelization of TRANS21 on the KSR2, using 30 layers, versus idea speedup (solid line).

clock speed is 40 MHz, with peak performance of 80 MFlops per node, an access time to the subcache of 2 processor cycles (with a 64-byte cache line), an access time of 18 processor cycles to local memory, and an access time of 126 cycles to remote memory using a 128-byte cache line. Therefore, severe penalties exist concerning accesses to sub-cache, cache, and remote memory. Such penalties increase when additional rings exist in the memory access hierarchy. At the lowest level, the parallel programming model offered by the KSR's OSF Unix operating system is one of kernel-level threads (Pthreads) which offer constructs for thread fork, thread synchronization, shared memory between threads, and others. As stated earlier, TRANS is implemented with the Cthreads parallel programming library, which is layered on KSR Unix Pthreads.

Layer Parallelization. The first results presented in Figure 6 verify that layer parallelization of the transport code results in excellent speedup for a moderate number of processors. Measured speedup for both TRANS21 and TRANS42 for computations involving 30 layers appear in Figure 6. The same experiments run on different parallel machines, a GP1000 BBN Butterfly and an SGI Challenge multiprocessor, exhibit similar performance behaviors. For brevity, such results are not shown here.

As stated earlier, these measurements simply verify that data exchanges in the transport schemes of spectral transport models occur mainly horizontally, not vertically. This is to some extent also true for grid based models, therefore the amounts of data exchanged vertically depend on the specific physical or chemical processes being modeled rather than the transport method being used. Examples of such processes exhibiting excessive vertical data exchanges include radiation modeling, cloud modeling, etc. (e.g. [DFW+93]) Furthermore, computational loads are balanced

equally across layers, resulting in near linear speedup.

As described in Section 2.2 and as commonly done in climate models, the scientific version of TRANS will use a total of 37 atmospheric layers. This moderate number of layers would limit the scalability of a layer-parallelized code to small-scale parallel machines. This implies that additional parallelization is required to enable the TRANS code to take advantage of modern, large-scale parallel machines.

Term Parallelization. The second parallelization strategy employed in TRANS is the independent and concurrent computation of the different terms occurring in Equation 7. Several issues arise, including:

- dynamic vs. static creation and allocation of threads performing term calculation, resulting in implications concerning the locality of the grid data being accessed in term computations,

- performance limitations due to overheads arising from concurrent term computation, such as additional thread synchronization and additional interprocessor communications, and

- potential load imbalances caused by differences in term execution times.

Measured speedup results for the term parallelization versions of TRANS21 on the KSR1 and KSR2, and for TRANS42 on the KSR2 are depicted in Figure 7 for a model run with 14 vertical layers. It is apparent from these results that initial near-linear speedup (up to 14 processors) due to layer parallelism degrades to less than linear speedup due to unequal loads resulting from the concurrent computation of different terms. Specifically, computational efficiency drops to about 0.6 when 56 processors execute a 14 layer model, with the 4 terms A..D in Equation 7 being calculated in parallel. Reductions in parallel efficiency are due to three reasons. First, some additional communications result from the necessary exchange of spectral data between parallel term computations (grid data need not be exchanged since it is decomposed across term processors and allocated locally). Secondly, and more importantly, load imbalances result from differences in the execution times of terms. Thirdly, some additional overhead results from the necessary synchronization required before and after terms are computed.

The measurements depicted in Figure 8 demonstrate the primary contribution of load imbalance to the reduction in parallel efficiency. In these measurements, monitoring support available in Cthreads (see [GEK+94]) is used to measure term execution times for a three-layer run of the STRAT42 model. These measurements assume locally resident grid data. Load imbalance and reductions in parallel efficiency are further aggravated when such locality of grid data is not assured (see Section 3.4).

μ **Parallelization.** Measured program speedup with the independent and concurrent calculation of latitude bands inside the different terms is depicted in Figure 9. The simulations measured here use three vertical layers with a varying number of processors. In these simulations, each term has assigned to it the same number of 'help processors'. Changes in this assignment can be used to further tune program performance. The number of latitudes calculated by each help processor (N^μ) is approximately given by:

$$N^\mu \approx \frac{n^\mu}{\frac{\frac{n^p}{n^l}-1}{3}} \tag{1}$$

where n^μ = the total number of μ's divided by two, n^p is the number of processors used and n^l is the number of layers simulated. This means, for example, that the model computation measured in
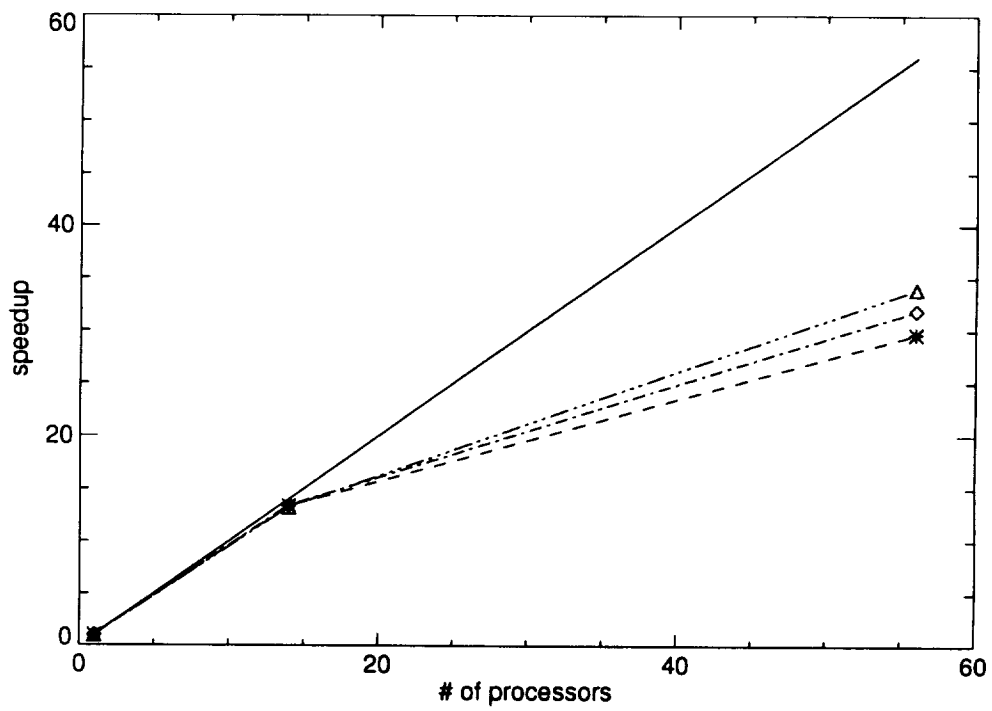
13

Figure 7: Measured speedup (14 layers) for term parallelization of TRANS21 on the KSR1 (asterisks) and the KSR2 (diamonds) and of TRANS42 on the KSR2 (triangles).
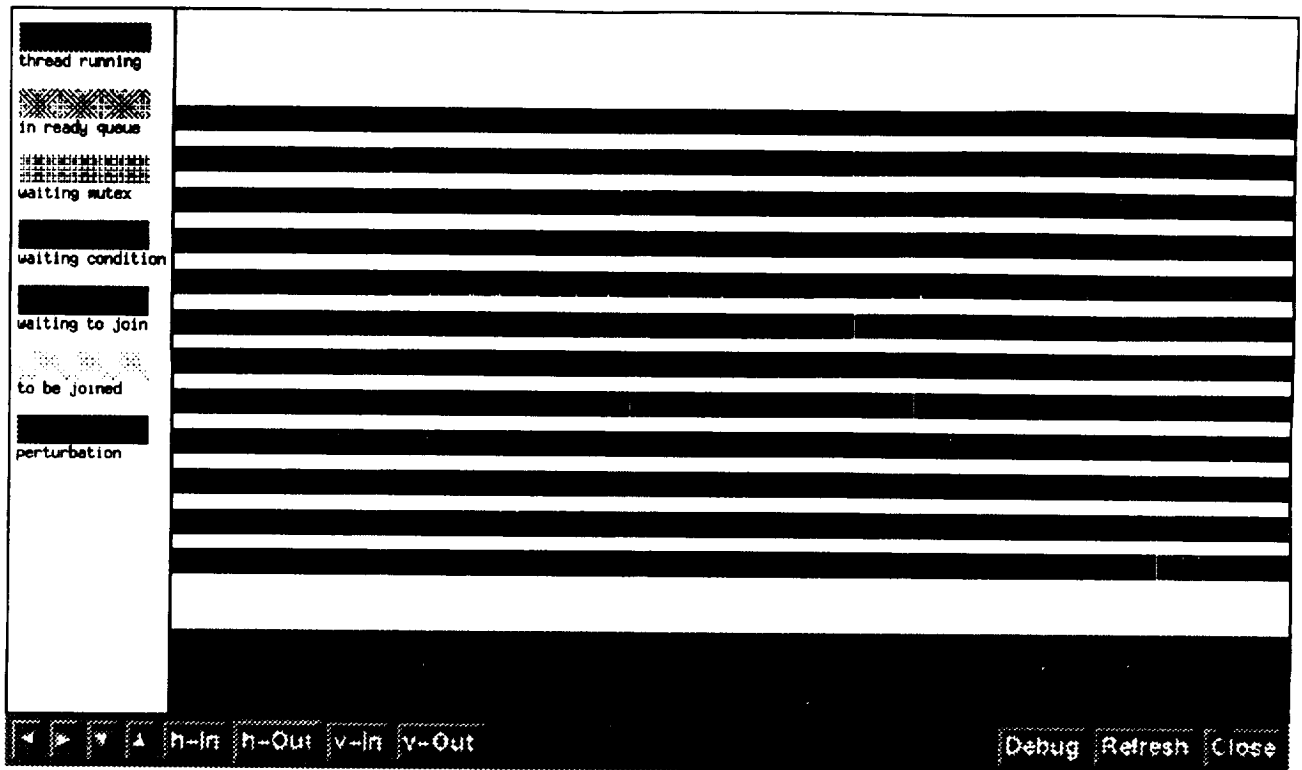
Figure 8: Monitoring of term parallelization of STRAT42 for 3 layers.

Figure 9 for TRANS21 (32 latitudes) uses 58 processors to calculate between 3 and 4 latitudes per help processor, whereas in TRANS42 (64 latitudes) with the same amount of processors, between 6 and 7 latitudes are calculated by each such processor.

With μ parallelization, two additional performance issues must be considered:

- *load imbalances* – caused by different numbers of latitudes computed by help processors, or caused by differences in term execution times; the latter can be corrected in principle by assigning different numbers of helper threads to each term calculation; and

- *granularity* – excessively small amounts of work performed by help processors, where computational overheads due to help processor use outweigh the benefits attained from additional parallelism.

The performance of model runs with μ parallelization is depicted in Figure 9. From these measurements, it appears that the best granularity of computation for helper processes is the calculation of approximately 3 latitudes per helper; no further performance gains are attained when computing a smaller number of latitudes per helper processor. This conclusion is not valid. Instead, Section 3.4 shows that speedup is primarily impeded by load imbalances.

3.4 Evaluation of Overheads due to Load Imbalance and Communication

Previous measurements shown in this section may be interpreted as limitations in parallelism due to the data exchanges or communication required in parallel spectral transport models. The purpose of this section is to investigate the various causes of and solutions to performance impediments.
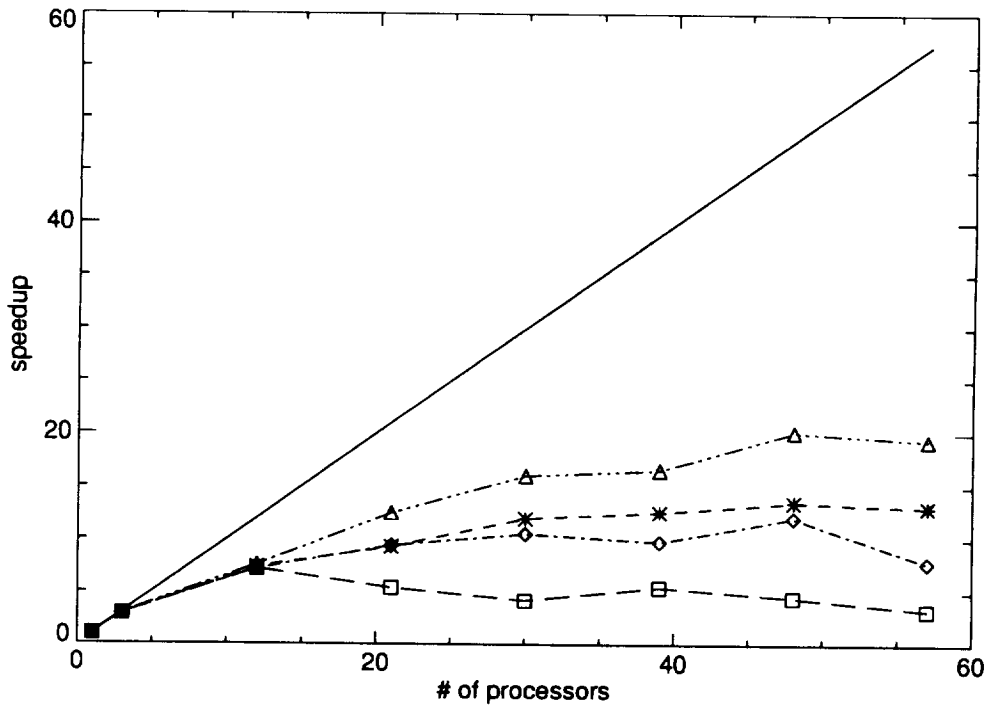
15

Figure 9: Measured speedup for μ parallelization versions of TRANS21 on the KSR1 (asterisks), TRANS21 on the KSR2 (diamonds) and TRANS42 on the KSR2 (triangles), with simulations performed for 2 days and 3 layers. Speedup results for a dynamic helper version of STRAT21 are also shown (squares).
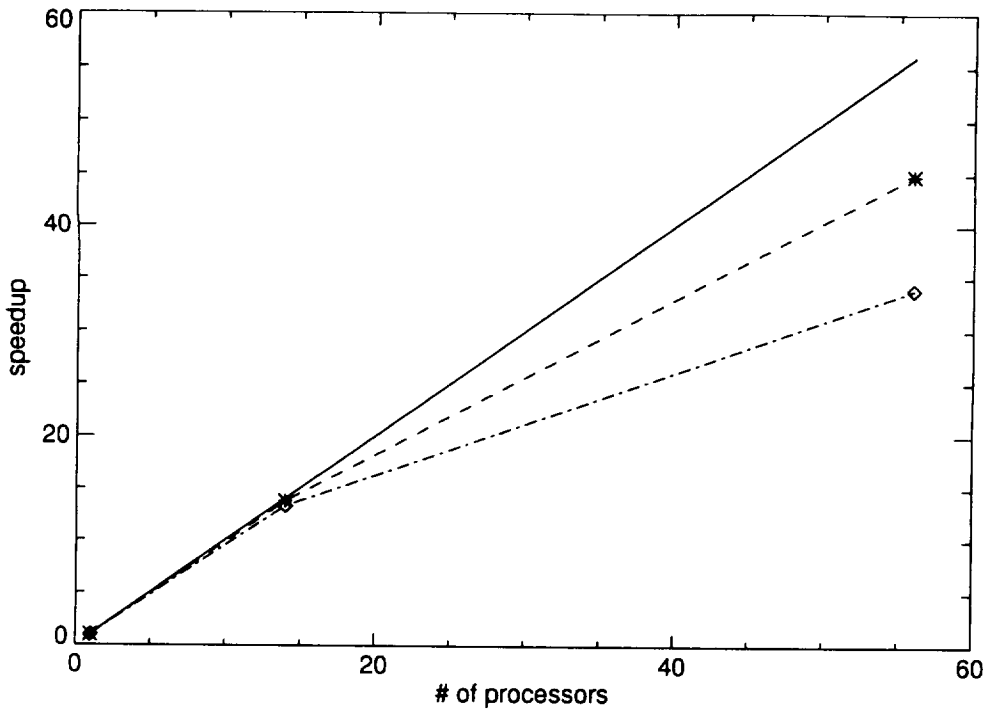
Figure 10: Measured speedup for actual TRANS42 on the KSR2 (diamonds) compared to a synthetic version of TRANS42 (asterisks) for 14 layers.

Load imbalance. As shown in Figure 8, the somewhat sharp drop in computational efficiency when using term parallelization is due to load imbalances arising from differences in term execution times. To further understand this phenomenon, we have created an artificially load-balanced (or synthetic) version of the TRANS model (ie., term computations are artificially increased in length when needed). The resulting speedup for 14 layers is shown in Figure 10, resulting in improvements in efficiency from about 0.61 to more than 0.8.

These measurements also explain the results depicted in Figure 9, where μ parallelization exhibits limited speedup apparently due to the granularities of helper thread computations. Actual speedup impediments are again due to the load imbalance in term parallelization, which results in similar load imbalances for μ parallelization, since each helper processor computes different numbers of latitudes.

Load balance can be improved in several ways, including:

1. The addition of chemical source and sink calculations to threads computing terms such that total thread computation times are similar. This will lead to performance results like those shown in Figure 10.

2. The concurrent calculation of the transport for several species, again performed by helper processors such that loads are balanced during term computation.

3. The assignment of different numbers of helper processors to threads performing term computations.

17

Concerning 1, many of the species transported in the atmosphere and being investigated by scientists are also undergoing chemical changes. The computations required to model such changes can be distributed across helper processors (or even helper threads) and therefore, improve imbalanced loads. In addition, these chemical changes are highly non-linear in nature and therefore, have to be calculated in the grid domain. This implies that their computation by helper processors (rather than by additional processors) is important since such processors already have locally available the required grid information, thereby avoiding additional inter-processor exchanges of grid data. However, the assignment of chemical computations to helper processors to correct load imbalances has to be performed for each model run, depending on the actual chemistry being performed, the number of simulated species, etc. It is important, therefore, to construct TRANS such that the number of helper threads, the computations performed by them, and the assignment of threads to processors is easily varied.

Communication overheads. We have constructed a version of TRANS that is able to allow any number of helper threads and processors to first acquire and then execute any number of term or chemistry computations, called TRANS-D (D for dynamic helpers). In this version, some limited number of helper threads is created at the time of program initialization, but no work is assigned to helpers at that time. Instead, each helper retrieves a work description from a global 'work queue' into which previous computations deposit work items (e.g., a term computation). One interesting insight from the use of TRANS-D is that the use of dynamic helpers can severely degrade program performance unless work items are assigned to helpers such that the locality of the grid data required for such work is maintained. This is shown in Figure 9, where the dynamic helper version results in poor performance due to grid data movement caused helper execution on any available processor (rather than on the processor possessing the appropriate grid data). Formulations of the suitable restrictions of mappings of threads to processors have been widely explored in the literature (e.g., see early work described in [SJ84]), but they have not been added to TRANS-D.

The exchange of spectral information among different helper threads does not affect program performance, as readily seen from the improvements in speedup attained when balancing helper loads (see Figure 10) while also exchanging spectral data among helpers. To quantify the effects of spectral data exchange among helper processors, term parallelization version of TRANS has been run such that spectral information exchanges are suppressed, which produces incorrect simulation results. Timings of such incorrect simulations exhibit only minor differences to timings of standard TRANS, thereby indicating that spectral data exchange is not a limiting factor in program performance.

3.5 Input analysis

Atmospheric models can produce and consume large amounts of data. Therefore, application performance can be severely affected by input/output performance. As mentioned in Section 2.2, the TRANS model requires the input of windfield information (two real numbers for every spectral point in the model) for every simulation day (e.g., every 12 time steps) for each layer. For high input performance, we distribute such windfield information across multiple files (one per level) prior to each model run, where a variable number of input threads can read such files prior during model execution, and where model computations are programmed to proceed whenever input is available. In addition, 2 days of windfield information are input prior to model computation, so that model computation can proceed concurrently with additional windfield inputs. Model output is also performed concurrently, using the concurrently executing layer processors. In Table 2, total

Machine	num of procs	I/O time seconds	speedup	total run time seconds	percentage of total
SGI	1	0.64	1	98.3	0.65
KSR2	1	5.12	1	140.4	3.65
KSR2	2	2.58	1.98	140.4	1.84
KSR2	5	1.12	4.57	140.4	0.64
KSR2	10	2.40	2.13	140.4	1.71

Table 2: Execution times of inputs for TRANS21 for 2 simulations days and 10 layers, using a simulation run on a single processor. The last column shows the relationship between input time and total simulation time as a percentage of total execution time.

input time is evaluated for a variable number of input threads for a TRANS21 run for 10 layers and 2 days of simulation time. From this table, it is apparent that the KSR2 offers limited parallelism for I/O. Specifically, our machine has 7 input/output adapters, so that input time improves for up to 5 threads, then degrades for 10 threads concurrently reading input files. Similar levels of concurrency can be attained for data output.

3.6 Discussion

The performance results presented in this section demonstrate several properties of the parallelized spectral transport code relevant to high performance computing:

- Significant performance gains are attained even when parallelizing only the transport component of a global atmospheric modeling code. Such performance gains can be expected to extend to large parallel machines offering hundreds of processors, and they should increase for more complex atmospheric computations that include chemical modeling, etc.

- Parallelization across different atmospheric layers is advantageous for transport processing, but limits parallelism due to most models' use of a moderate number of layers (e.g., 37 layers). This necessitates a parallelization approach relying on multiple methods for program and data decomposition, such as term and μ parallelization. Models that require extensive computation integrating over vertical atmospheric columns[WF94] may prefer μ to layer parallelization, but they still remain subject to the other performance issues studied in this paper, including load balancing, the minimization of grid data exchange, etc. The use of additional methods of parallelization (e.g., FFT parallelization) are likely to further increase the degree of parallelism available in global atmospheric models. Sample additional parallel activities pursued by our group include the introduction of additional terms during term parallelization, or the experimentation with alternative advection schemes (e.g., the semi-lagrangian transport of water can be parallelized, independently of spectral transport, in the grid domain).

- The global sharing of spectral data results in few additional runtime overheads, while the sharing of grid data results in significant performance penalties on any parallel machine subject to restrictions in communication latencies and bandwiths.

- Load imbalances result in large performance penalties for larger number of processors. Such imbalances may be removed by pursuit of mixed parallelization strategies (e.g., term with μ parallelization) and/or by additional consideration of specific properties (e.g., chemical properties) of atmospheric constituents.

The performance results presented in this section are gained by experimentation with a Kendall Square KSR2 supercomputer. While this machine offers a shared memory model, the performance insights attained with it also extend to non-shared memory machines like the Intel Paragon or the Cray T3D, because the penalties for remote vs. local memory access on the KSR machine approximate those of distributed memory platforms (ie., penalties of up to 1:100 for local cache vs. remote memory accesses). Furthermore, the threads platform used in our experimentation is easily ported to other shared memory parallel machines (e.g., Cthreads are already running on the SGI multiprocessor machines).

3.7 Related Research

Research related to our efforts falls into two categories:

- the parallelization of weather forecast models, as performed prior to our work by research groups at the European Center for Medium-Range Weather Forecast, using spectral weather models [DS89], and

- the parallelization of climate models, as performed concurrently with our work at Oakridge National Laboratories in the U.S., using NCAR's CCM2 model [HBB+92].

Both of these research groups employ distributed rather than shared memory machines in their research. However, differences in their results to the work presented in this paper are due primarily to differences in parallelization approaches. Specifically, both of these research groups parallelize the "shallow water model" equation for one plane, requiring the solution of three equations for three unknowns, whereas the amount of computation being parallelized in our work is less: we are parallelizing only one equation with one unknown. This implies that the resulting ratio of local computation to communication in the models parallelized elsewhere is more advantageous than in the TRANS model. Therefore, the detailed study of spectral transport parallelization with the TRANS model can be considered a necessary prerequisite or element of any parallelization effort involving large-scale atmospheric codes.

Specific comments on each paper describing related work are presented next. In the European models, Barros et al. [BK90] investigate spectral and grid based parallelized solutions to Helmholtz-type equations on an iPSC/2 hypercube with 32 nodes. μ parallelization of the Laplace transformation is employed (as in our work), comparing two types of data exchanges between processors: (1) the rotation approach, where data is communicated stepwise between processors, with each processor removing from a message the information it requires and adding to the message the information required by its neighbors, vs. (2) the transposition approach, where all data is exchanged in chunks among the processors requiring it. Both communication approaches result in parallel efficiencies exceeding 90%, with the transposition approach being slightly more efficient. The disadvantage of this approach is that parallelization is limited to a total of $J/2$ processors, where J =number of latitudes. Detailed studies of alternative communication approaches are not part of our work with the TRANS model, but may constitute an interesting extension of our research even on shared memory machines like the KSR due to the machine's NUMA properties.

Further parallelizations by this research group concern a large-scale nCube/2, where they attain an efficiency of 86% on 86 processors using the 2-D ECMF weather forecast model [GJS93], and efficiencies of about 85% on 8 processors on an IBM SP1 machine with the 3-D ECMF weather forecast model [GJS94].

In the U.S., Jacob et al. [JH89] solve the shallow water model in a plane on a 20-processor Encore Multimax shared memory machine, comparing finite difference vs. spectral transport solution

methods. Mixed parallelization methods are used, where the three different equations of the shallow water model are computed concurrently, while also using μ parallelization when solving each of the three unknowns. Efficiencies of up to 96% are achieved for 20 processors.

Concurrent with our work, Worley et al. at Oakridge[WD92] solve the shallow water model in one plane on a Intel iPSC/860 with 128 nodes, using μ parallelization for the Laplace transformation and the aforementioned rotational data exchange. Efficiencies of .31 and .35 are attained for T21 on 16 processors and for T42 on 32 processors. Initial results are limited to using up to a maximum of 32 processors for the T42 model. These restrictions are removed in [WWD92], where additional parallelism is attained by parallelizing the FFT computations performed across the data in vertical atmospheric patches. This mixed parallelization approach results in a composite efficiency of about 0.1 for T21 on 64 processors and of about 0.15 for T42 on 128 processors. Load imbalances appear the primary causes of reduced efficiencies on large-scale machines. Additional measurements appear in [WF94], where results are attained and compared on a 1024 processor NCube/2 machine, a 128 processor iPSC/860, a 512 processor Paragon, and a 512 processor Connection Machine. Specific optimizations address each of the machines being used.

In [FW94], Oakridge investigators compare several possible parallelization of the spectral method for solving the Shallow Water Equation, including (1) parallelization of the Laplace transformation using the (a) transpose technique versus the (b) rotation technique, (2) parallelization of the FFT using the (a) transpose versus (b) rotation techniques. This parallelization uses a model exhibiting several atmospheric layers but does not parallelize across layers. In [DFW+93],the same methods are used for parallelization of NCAR's CCM2 model on an Intel iPSC/860 with 128 processors. An additional transport scheme has to be employed for computing the advection of moisture fields (the spectral approach does not appear to work well for this purpose). Instead, CCM2 uses a semi-Lagrangian transport scheme, which results in the use of two transport schemes in the model being parallelized. The authors again use μ parallelization, assigning overlapping regions to processors. Such an assignment scheme could be explored with the TRANS model as well.

Acknowledgements

References

[ABB+94] J.J. Ambrosiano, J. Bolstad, A.J. Bourgeois, J.C. Brown, and B. Chan. High-performance climate modeling using a domain and task decomposition message-passing approach. pages 397–405. IEEE, August 1994.

[BK90] S. R. M. Barros and T. Kauranne. Spectral and multigrid spherical helmholtz equation solvers on distributed memory parallel computers. In *Fourth Workshop on use of parallel pocessors in meteorology*, pages 1–27. European Centre for Mefium-Range Weather Forecasts, November 1990.

[CAP80] Derek M. Cunnold, Fred Alyea, and R. Prinn. Preliminary calculations concerning the maintenance of the zonal mean ozone distribution in the northern hemisphere. *Pure Appl. Geophys.*, 118:329–354, 1980.

[CAPP75] Derek M. Cunnold, Fred Alyea, N. Philips, and R. Prinn. A three-dimensional dynamical-chemical model of atmospheric ozone. *J. Atmos. Sci.*, 32:170–194, 1975.

[CMS93]   Christian Clemencon, Bodhisattwa Mukherjee, and Karsten Schwan. Distributed shared abstractions (dsa) on large-scale multiprocessors. In *Proc. of the Fourth USENIX Symposium on Experiences with Distributed and Multiprocessor Systems*, pages 227–246. USENIX, September 1993. Also as TR# GIT-CC-93/37.

[DFW+93]   J. B. Drake, R.E. Flanery, D.W. Walker, I.T. Foster P.H. Worley, J.G. Michalakes, R.L. Stevens, J.J. Hack, and D.L. Williamson. The message passing version of the parallel community climate model. In G-R. Hoffman and T. Kauranne, editors, *Parallel Supercomputing in Atmospheric Science*, pages 500–513. World Scientific, 1993.

[DS89]   D. Dent and A. Simmons. The ecmwf multi tasking weather prediction model. *Computer Physics Reports*, 11:153–194, 1989.

[EGSM94]   Greg Eisenhauer, Weiming Gu, Karsten Schwan, and Niru Mallavarupu. Falcon – toward interactive parallel programs: The on-line steering of a molecular dynamics application. Technical Report GIT-CC-94-08, Georgia Institute of Technology, College of Computing, Atlanta, GA 30332-0280, 1994. also in High-Performance Distributed Computing (HPDC-3).

[Eis94]   Greg Eisenhauer. Portable self-describing binary data streams. Technical Report GIT-CC-94-45, College of Computing,Georgia Institute of Technology, Atlanta, GA 300332, 1994. *(anon. ftp from ftp.cc.gatech.edu)*.

[ES94]   Greg Eisenhauer and Karsten Schwan. Md - – a flexible framework for high-speed parallel molecular dynamics. In Adrian Tentner, editor, *High Performance Computing, Proceedings of the 1994 SCS Simulation Multiconference*, pages 70–75. Society for Computer Simulation, Society for Computer Simulation, April 1994.

[FW94]   I.T. Foster and P.H. Worley. Parallel algorithms for the spectral transform method. Technical Report ORNL/TM-12507, Oak Ridge National Laboratory, April 1994.

[GEK+94]   Weiming Gu, Greg Eisenhauer, Eileen Kraemer, Karsten Schwan, John Stasko, Jeffrey Vetter, and Nirupama Mallavarupu. Falcon: On-line monitoring and steering of large-scale parallel programs. Technical Report GIT-CC-94-21, Georgia Institute of Technology, College of Computing, Atlanta, GA 30332-0280, April 1994. Submitted to Frontiers 95.

[GJS93]   U. Gaertel, W. Joppich, and A. Schueller. Parallelizing the ecmwf's weather forecast program: the d case. *Parallel Computing 19*, 19:1427–1429, 1993.

[GJS94]   U. Gaertel, W. Joppich, and A. Schueller. Medium-range weather forecast on parallel systems. pages 388–391. IEEE, August 1994.

[GMS94]   Kaushik Ghosh, Bodhisattwa Mukherjee, and Karsten Schwan. Experimentation with configurable, lightweight threads on a ksr multiprocessor. In *In the Proceedings of the First International Workshop on Parallel Processing, Bangalore, India*, December 1994. Expanded version available as Georgia Tech TR# GIT-CC-93/37.

[GMSS94]   Ahmed Gheith, Bodhi Mukherjee, Dilma Silva, and Karsten Schwan. Ktk: Kernel support for configurable objects and invocations. In *Second International Workshop on Configurable Distributed Systems*, pages 236–240. IEEE, ACM, March 1994.

[GVS94]   Weiming Gu, Jeffrey Vetter, and Karsten Schwan. An annotated bibliography of interactive program steering. *ACM SIGPLAN Notices*, 29(9):140–148, Sept. 1994.

[Hau40]   B. Haurwitz. The motion of atmospheric disturbances on the spherical earth. *Journal of Mar. Res.*, 3:254–267, 1940.

[HBB$^+$92]   J.J. Hack, B.A. Boville, B.P. Briegleb, J.T. Kiehl, P.J. Rasch, and D.L. Williamson. Description of the ncar community climate model (ccm2). NCAR Tech. Note TN-382+STR, National Center for Atmospheric Research, 1992.

[JE45]   E. Jahnke and F. Emde. Tables of functions. *Dover Publications Inc.*, 1945.

[JH89]   R. Jakob and J. Hack. Parallel mimd programming for global models of atmospheric flow. *ACM*, pages 106–112, 1989.

[Joh89]   H.S. Johnston. Evaluation of excess carbon 14 and strontium 90 data for suitability to test two-dimensional stratospheric models. *J. Geophys. Res.*, 94:18485–18493, 1989.

[KHYK61]   S. Kubota, M. Hirose, Y.Kichuchi, and Y. Kurihara. Barotropic forecasting with the use of surface spherical harmonic representation. *Pap. Meteorol. Geophys.*, 12:199–215, 1961.

[Lor71]   E.N. Lorenz. An n-cycle time-differencing scheme for stepwise numerical integration. *Monthly Weather review*, 99(8):644–648, 1971.

[Muk91]   Bodhisattwa Mukherjee. A portable and reconfigurable threads package. In *Proceedings of Sun User Group Technical Conference*, pages 101–112, June 1991.

[Pla60]   G.W. Platzmann. The spectral form of the vorticity equation. *J. Meteorol.*, 17:635–644, 1960.

[Pra92]   M. Prather. Special numerical experiment: Simulation of cfcl3 as a test for 3-d atmospheric models. Technical report, Report for WCRP Workshop on Long-Range transport of trace gases, 1992.

[RTBW94]   P.J. Rasch, X. Tie, B.A. Boville, and D.L. Williamson. A three-dimensional transport model for the middle atmosphere. *J. Geophys. Res.*, 99:999–1017, 1994.

[SFG$^+$91]   Karsten Schwan, Harold Forbes, Ahmed Gheith, Bodhisattwa Mukherjee, and Yiannis Samiotakis. A cthread library for multiprocessors. Technical report, College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, GIT-ICS-91/02, Jan. 1991.

[Sil54]   I.S. Silberman. Planetary waves in the atmosphere. *J. Meteorol.*, 11:27–34, 1954.

[SJ84]   Karsten Schwan and Anita K. Jones. Specifying resource allocation for the cm* multiprocessor. *IEEE Software*, 3(3):60–70, May 1984.

[SO93]   R. Swinbank and A. O'Neill. A stratosphere - troposphere data assimilation system. Climate Research Technical Note CRTN 35, Hadley Centre Meteorological Office, London Road Bracknell Berkshire RG12 2SY, March 1993.

[WAB+93] M.F. Wehner, J.J. Ambrosiano, J.C. Brown, W.P. Dannevik, P.G. Eltgroth, A.A. Mirin, J.D. Farrara, C.C. Ma, C.R. Mechoso, and J.A. Spahr. Toward a high performance distributed memory climate model. *IEEE*, pages 102 – 113, 1993.

[WD92] P.H. Worley and J.B. Drake. Parallelizing the spectral transform method. *Concurrency: Practice and Experience*, 4(4):269–291, 1992.

[WF94] P.H. Worley and I.T. Foster. Parallel spectral transform shallow water model: A runtime-tunable parallel benchmark code. In *SHPCC '94*, pages 207-214. IEEE Computer Society, 1994.

[WP86] W.M. Washington and C.L. Parkinson. *An introduction to three-dimensional climate modeling*. Oxford University Press, 1986.

[WWD92] D.W. Walker, P.H. Worley, and J. B. Drake. Parallelizing the spectral transform method - part ii. *Concurrency: Practice and Experience*, 4(7):509-531, October 1992.

A Mathematical Model Description

We first describe the governing equation for the global transport of any atmospheric constituent. Next, a short overview is presented of the spectral approach to solving this second order differential equation.

Equations for Global Transport. The continuity equation for any atmospheric constituent Y with mixing ratio X^2 is given as:

$$\frac{\partial X}{\partial t} = -\vec{v} \cdot \nabla X \tag{2}$$

where $\vec{v}$ is the wind velocity which can be written as:

$$\vec{v} = (\vec{u}, W) \tag{3}$$

where $\vec{u}$ is the horizontal part of the windfield and W is the vertical advection velocity.

By expanding the horizontal wind velocity $\vec{u}$ in terms of a horizontal stream function ψ and a velocity potential χ ($\vec{k}$ is the vertical unit vector and the $\vec{\nabla}$ is taken on a constant pressure surface):

$$\vec{u} = \vec{k} \times \nabla \psi - \nabla \chi \tag{4}$$

it can be shown, that the continuity equation for the transported species Y can be expressed as:

$$\frac{\partial X}{\partial t} = -(\vec{k} \times \nabla \psi) \cdot \nabla X + \nabla \chi \cdot \nabla X - W\frac{\partial X}{\partial Z} \tag{5}$$

where W is the vertical advection velocity, $Z = -\ln(P)$, and P is the ratio of pressure to 1000 mbar.

---

[2]Concentrations of atmospheric trace gases are usually expressed as a ratio between the number of molecules of species Y and the number of air molecules in a given volume. This is called the mixing ratio of species Y and is usually given in parts per million (ppm), parts per billion (ppb), or parts per trillion (ppt).

To take into account the diffusive character of the transport of any species in the atmosphere a vertical and horizontal diffusion term has to be added:

$$\frac{\partial X}{\partial t} = -(\vec{k} \times \nabla \psi) \cdot \nabla X + \nabla \chi \cdot \nabla X - W \frac{\partial X}{\partial Z} + \frac{\partial}{\partial P} (\frac{K_d P}{H_0^2} \frac{\partial X}{\partial Z}) + A_H \nabla^2 X \qquad (6)$$

where A_H and K_z are horizontal and vertical diffusion coefficients and H_0 is the scale height ($\approx 8km$).

By using spherical coordinates ($\lambda =$ longitude, $\mu = \sin(\phi)$, $\phi = latitude$, $Z = -\ln(P)$) and by making the equation dimensionless, the final prognostic equation for species Y can be written as:

$$\frac{\partial X}{\partial t} = \overbrace{\frac{\partial X}{\partial \lambda} \frac{\partial \psi}{\partial \mu} - \frac{\partial \psi}{\partial \lambda} \frac{\partial X}{\partial \mu}}^{A} +$$

$$\overbrace{\frac{1}{(1-\mu^2)} \frac{\partial \chi}{\partial \lambda} \frac{\partial X}{\partial \lambda} + (1-\mu^2) \frac{\partial \chi}{\partial \mu} \frac{\partial X}{\partial \mu}}^{B} -$$

$$\overbrace{W \frac{\partial X}{\partial Z}}^{C} + \overbrace{\frac{1}{H_0^2 P} \frac{\partial}{\partial Z}(K_z P \frac{\partial X}{\partial Z}) + A_H(\frac{1}{(1-\mu^2)} \frac{\partial^2 X}{\partial \lambda^2} + \frac{\partial}{\partial \mu} \mu^2 \frac{\partial X}{\partial \mu})}^{D} \qquad (7)$$

For further referencing the different terms have been named A ... D
A more detailed description of the spectral method used in our model to carry out this integration appears in the next section.

The Spectral Approach to Solving the Transport Equation. We will not discuss the spectral method in depth, but instead provide a brief overview. More detailed information can be found in several publications, including[Hau40, Sil54, Pla60, KHYK61, WP86, FW94].

Any variable $F(\lambda, \mu, t)$ in a 2 dimensional spherical surface can be approximated by an expansion into a set of orthogonal spherical basis functions, called spherical harmonics:

$$F(\lambda, \mu, t) = \sum_{l=-lMax}^{lMax} \sum_{n=|l|}^{nMax(l)} f_{l,n}(t) Y_{l,n}(\lambda, \mu) \qquad (8)$$

The complex quantities $f_{l,n}(t)$ are called expansion coefficients. The orthogonal spherical harmonics $Y_{l,n}(\lambda, \mu)$ are given as[JE45]:

$$Y_{l,n}(\lambda, \mu) = e^{il\lambda} P_{l,n}(\mu) \qquad (9)$$

where $P_{l,n}(\mu)$ are the normalized Legendre's associated function of the first kind. By setting $nMax(l) = lMax$ a triangular spectral truncation is used. For applying a unaliases transform λ^{Max} (number of longitudes) has to satisfy:

$$\lambda^{Max} \geq 3nMax \qquad (10)$$

To simplify the Fast Fourier Algorithm involved in the transformations usually λ^{Max} is chosen to be the smallest power of two satisfying this equation. In addition the number of latitudes is usually set be half the number of longitudes. Therefore $nMax$ can be used to characterize the horizontal

resolution of a model, eg. T42 means a triangular truncation with $nMax = 42$, $\lambda^{Max} = 128$ and the number of latitudes is 64. One of the necessary characteristics of spherical harmonics to be useful as a basis for an expansion is their orthogonality:

$$\int_{-1}^{1} \int_{0}^{2\pi} Y_{l,n}(\lambda,\mu) Y_{l',n'}^{*}(\lambda,\mu) d\lambda d\mu = \begin{cases} 4\pi & \text{if n=n' and l=l'} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

where $Y_{l',n'}^{*}(\lambda,\mu)$ represents the complex conjugate of $Y_{l,n}(\lambda,\mu)$. By adding a weight function $w(\mu)$, this orthogonality holds also if instead of the two integrals only finite sums over μ and *lambda* are executed. Using this orthogonality the expansion coefficients $f_{l,n}(t)$ is easily calculated (if $F(\lambda,\mu,t)$ is known)

$$\sum_{\mu,\lambda} F(\lambda,\mu,t) Y_{l',n'}^{*}(\lambda,\mu) =$$

$$\sum_{\mu,\lambda} \left(\sum_{l=-lMax}^{lMax} \sum_{n=|l|}^{nMax} f_{l,n}(t) Y_{l,n}(\lambda,\mu) w(\mu) \right) Y_{l',n'}^{*}(\lambda,\mu) w(\mu) =$$

$$f_{l',n'}(t) \text{ for each } l' \text{ and } n' \tag{12}$$

The spectral method of solving the transport equation takes advantage of the fact that terms like $\frac{\partial F}{\partial \mu}$ and $\frac{\partial F}{\partial \lambda}$ are much easier and more accurately calculated in the spectral domain (compared with a grid based finite difference scheme):

$$\frac{\partial F}{\partial \lambda} = \left(\sum_{l=-lMax}^{lMax} \sum_{n=|l|}^{nMax} il f_{l,n}(t) Y_{l,n}(\lambda,\mu) \right) \tag{13}$$

$$\frac{\partial F}{\partial \mu} = \left(\sum_{l=-lMax}^{lMax} \sum_{n=|l|}^{nMax} f_{l,n}(t) e^{il\lambda} \frac{\partial P_{l,n}(\mu)}{\partial \mu} \right) \tag{14}$$

where $\frac{\partial P_{l,n}(\mu)}{\partial \mu}$ are known functions od μ (time independent).

GEORGIA INSTITUTE OF TECHNOLOGY

PAGE 02

PRINCIPAL INVESTIGATOR	K SCHWAN		CENTER NO.	246R80880AO	ACCOUNT NO.	C-36-X32
STATUS AT END OF	AUGUST	1994			DEPARTMENT	COLL COMP
SPONSOR	NASA GTRC WASHINGTON					
AWARD NUMBER	NAGW-3886		RF CENTER NO.	0064145100O	RESTRICTED FUND	RF-43165
EFFECTIVE DATE	04-01-94		BILLING GROUP GTRC		EXPIRATION DATE	03-31-97

	MONTH	FISCAL YEAR	TOTAL CONTRACT
PERSONAL SERVICES			
BUDGET			20,850.25
EXPENDED	2,423.66	12,387.32	19,325.30
ENCUMBERED	2,423.66	2,423.66	2,423.66
FREE BALANCE	-2,423.66		-898.71
FRINGE BENEFITS			
BUDGET			8,495.94
EXPENDED	.00	1,862.38	1,862.38
ENCUMBERED	.00	.00	.00
FREE BALANCE			6,633.56
MATERIALS AND SUPPLIES			
BUDGET			-200.00
EXPENDED	.00	.00	.00
ENCUMBERED	.00	44.50	44.50
FREE BALANCE			-244.50
TRAVEL			
BUDGET			5,000.00
EXPENDED	.00	.00	.00
ENCUMBERED	.00	1,130.00	1,130.00
FREE BALANCE			3,870.00
COMPUTER			
BUDGET			5,000.00
EXPENDED	.00	.00	504.38
ENCUMBERED	.00	.00	.00
FREE BALANCE			4,495.62
TOTAL DIRECT CHARGES			
BUDGET			39,146.19
EXPENDED	2,423.66	14,249.70	21,692.06
ENCUMBERED	-2,423.66	3,598.16	3,598.16
FREE BALANCE			13,855.97

GEORGIA INSTITUTE OF TECHNOLOGY

PRINCIPAL INVESTIGATOR	K SCHWAN		CENTER NO.	246R8O880AO	ACCOUNT NO.	C-36-X32
STATUS AT END OF	AUGUST	1994			DEPARTMENT	COLL COMP

SPONSOR   NASA GTRC WASHINGTON

AWARD NUMBER   NAGW-3886   RF CENTER NO. OO64145IOOO   RESTRICTED FUND RF-43165

EFFECTIVE DATE 04-01-94   BILLING GROUP GTRC   EXPIRATION DATE 03-31-97

	MONTH	FISCAL YEAR	TOTAL CONTRACT

INDIRECT CHARGES

BUDGET			14,485.13
EXPENDED	969.46	5,699.88	8,453.54 RATE OF 40.0 BASE OF 4
ENCUMBERED	-969.46	1,439.27	1,439.27
FREE BALANCE			4,592.32

TOTAL

BUDGET			53,631.32
EXPENDED	3,393.12	19,949.58	30,145.60 THIS ACCOUNT IS 61.1 EXPENDED
ENCUMBERED	-3,393.12	5,037.43	5,037.43
FREE BALANCE			18,448.29