Project #: C-36-X32  Cost share #:  Rev #: 8
Center #: 10/24-6-R8088-0A0  Center shr #:  OCA file #:  Active
Contract #: NAGW-3886  Mod #: BUDGET REVISION  Document: GRANT
Prime #:  Contract entity: GTRC
Subprojects ?: Y  CFDA: 43.002
Main project #:  PE #:  

Project unit: COMPUTING  Unit code: 02.010.300
Project director(s): SCHWAN K  (404)894-2589

Sponsor/division names: NASA  / HEADQUARTERS/WASHINGTON, DC
Sponsor/division codes: 105  / 002

Award period: 940401 to 970331 (performance)  970630 (reports)

Sponsor amount  New this change  Total to date
Contract value  0.00  449,909.00
Funded  0.00  294,393.00
Cost sharing amount  0.00

Does subcontracting plan apply?: N

Title: PARALLELIZATION & VISUAL ANALYSIS OF DYNAMICAL/CHEMICAL ATMOSPHERE MODELS

PROJECT ADMINISTRATION DATA

OCA contact: Anita D. Rowland  894-4820
Sponsor technical contact  Sponsor issuing office
GLENN H. MUCKLOW, CODE ST  ADRIENE WOODIN
(202)358-2235  (202)358-0508

NASA HEADQUARTERS  SAME
300 E STREET SW
WASHINGTON, DC 20546

Security class (U,C,S,TS): U  ONR resident rep. is ACO (Y/N): Y
Defense priority rating :  supplemental sheet
Equipment title vests with: Sponsor  REF. GENERAL PROVISION 1260.408, GRANTS HANDBOOK.
Administrative comments -  TRANSFER OF FUNDS ($29118) TO SUBPROJECT.
Closeout Notice Date 05-DEC-1997

Project Number C-36-X32

Center Number 10/24-6-R8088-0A0

Project Director SCHWANS, KARSTEN

Project Unit COMPUTING

Sponsor NASA/HEADQUARTERS/WASHINGTON, DC

Division Id 3382

Contract Number NAGW-3886

Prime Contract Number

Title PARALLELIZATION & VISUAL ANALYSIS OF DYNAMICAL/CHEMICAL ATMOSPHERE MODELS

Effective Completion Date 30-SEP-1997 (Performance) 31-DEC-1997 (Reports)

Closeout Action:

<table>
<thead>
<tr>
<th>Y/N</th>
<th>Date Submitted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

Comments

Distribution Required:

<table>
<thead>
<tr>
<th></th>
<th>Y/N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Director/Principal Investigator</td>
<td>Y</td>
</tr>
<tr>
<td>Research Administrative Network</td>
<td>Y</td>
</tr>
<tr>
<td>Accounting</td>
<td>Y</td>
</tr>
<tr>
<td>Research Security Department</td>
<td>N</td>
</tr>
<tr>
<td>Reports Coordinator</td>
<td>Y</td>
</tr>
<tr>
<td>Research Property Team</td>
<td>Y</td>
</tr>
<tr>
<td>Supply Services Department/Procurement</td>
<td>Y</td>
</tr>
<tr>
<td>Georgia Tech Research Corporation</td>
<td>Y</td>
</tr>
<tr>
<td>Project File</td>
<td>Y</td>
</tr>
</tbody>
</table>

NOTE: Final Patent Questionnaire sent to PDPI
July 10, 1998

NASA Washington Headquarters  
Financial Management Division  
CODE BFH-7  
Washington, DC 20546

Reference: NASA – Research Grant NAGW-3886 - Federal Cash  
Transactions Report (SF-272)

Dear Madam/Sir:

Enclosed is the final Federal Cash Transactions Report (SF-272) for NASA - Washington Headquarters  
Research Grant NAGW-3886 with Georgia Tech Research Corporation/Georgia Institute of Technology.  
This report is for the period April 1, 1994 through September 30, 1997. Should you have questions, please  
contact Dale C. Turner of this office, (404) 894-5521.

Sincerely,

[Signature]

Director

CTD/dct

Enclosures

c:  
Mr. Eric Trevena, COC, 0280  
Ms. Danielle Herrmann, OCA, 0420  
Ms. Wanda Simon, OCA, 0420  
NASA Headquarters, Headquarters Acquisition Division,  
Code HWB-1, Attn: Closeout Unit, Washington, DC 20546  
ONR RR, Atlanta Regional Office, 101 Marietta Tower, Suite 2805,  
Atlanta, GA 30323
1. Federal sponsoring agency and organizational element to which this report is submitted

NASA - WASHINGTON HEADQUARTERS

2. RECIPIENT ORGANIZATION
Name: GEORGIA TECH RESEARCH CORPORATION
Number and Street: 400 10TH STREET, N W - ROOM 270
City, State and ZIP Code: ATLANTA, GA 30332-0415

3. FEDERAL EMPLOYER
IDENTIFICATION NO. 58-0603146

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a. Cash on hand beginning of reporting period</td>
<td>$0.00</td>
<td></td>
</tr>
<tr>
<td>b. Letter of credit withdrawals</td>
<td>449,908.40</td>
<td></td>
</tr>
<tr>
<td>c. Treasury check payments</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>d. Total receipts (Sum of lines b and c)</td>
<td>449,908.40</td>
<td></td>
</tr>
<tr>
<td>e. Total cash available (sum of line a and d)</td>
<td>449,908.40</td>
<td></td>
</tr>
<tr>
<td>f. Gross disbursements</td>
<td>449,908.40</td>
<td></td>
</tr>
<tr>
<td>g. Federal share of program income</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>h. Net disbursements (Line f minus line g)</td>
<td>449,908.40</td>
<td></td>
</tr>
<tr>
<td>i. Adjustments of prior periods</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>j. Cash on hand end of period</td>
<td>$0.00</td>
<td></td>
</tr>
</tbody>
</table>

11. STATUS OF FEDERAL CASH

12. THE AMOUNT SHOWN ON LINE 11J, ABOVE, PRESENTS CASH REQUIREMENTS FOR THE ENSUING 0 Days

13. OTHER INFORMATION
a. Interest Income $ |

14. REMARKS (Attach additional sheets of plain paper, if more space is required)

Questions concerning this report should be directed to: Dale C. Turner (404) 894-5521.

15. CERTIFICATION

I certify to the best of my knowledge and belief that this report is true in all respects and that all disbursements have been made for the purpose and conditions of the grant or agreement.

AUTHORIZED CERTIFYING OFFICIAL
SIGNED: CHARLES T. DUFFY, DIRECTOR GRANTS AND CONTRACTS ACCOUNTING

DATE REPORT SUBMITTED 7/10/98

THIS SPACE FOR AGENCY USE

STANDARD FORM 272 (7-76) Prescribed by Office of Management and Budget Cir. No. A-110
October 11, 1994

Glenn Mucklow
Information Systems Research and Technology
NASA Headquarters, 300 E Street, SW
Washington, DC 20546
USA

Dear Glenn:

Included is our first annual report. We are not including additional copies of the materials forwarded to you last July. Any comments on the new report and materials are appreciated.

Best wishes,

Karsten Schwan
Associate Professor
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

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$^2$, 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 (i.e., 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 (e.g., 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 (i.e., 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-
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 of 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 $X$ 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 $\lambda = \text{longitude}, \mu = \sin(\phi)$ ($\phi = \text{latitude}$) 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 $\frac{\partial F}{\partial \lambda}$ and $\frac{\partial F}{\partial \mu}$ 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 $\frac{\partial F}{\partial \mu}$ and $\frac{\partial F}{\partial \lambda}$ 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 ($IMax = 42$ 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).
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 remaining 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 $l_{Max} = 21$ 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 $l_{Max} = 21$ (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:
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 accessible. 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
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 $\mu$ 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 $\mu$ 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 $\mu$ (latitudes) (see the Appendix 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 $\mu$ 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, $\mu$ 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 $\mu$ values result in increased communication overheads that prevent increases in speedup even on the fairly tightly coupled KSR
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, cur­
crently 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 program­
mming 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 mu­
tex 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
Table 1: Sequential execution times of model versions on different machines, for 37 levels and 3 days simulation time.

<table>
<thead>
<tr>
<th>Machine</th>
<th>version</th>
<th>time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS6000</td>
<td>STRAT</td>
<td>518</td>
</tr>
<tr>
<td>SGI</td>
<td>STRAT</td>
<td>662</td>
</tr>
<tr>
<td>SGI</td>
<td>TRANS21</td>
<td>535</td>
</tr>
<tr>
<td>KSR1</td>
<td>TRANS21</td>
<td>1511</td>
</tr>
<tr>
<td>KSR2</td>
<td>TRANS21</td>
<td>780</td>
</tr>
</tbody>
</table>

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
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:

\[ N^μ \approx \frac{n^μ}{\frac{n^p}{n^l} - 1} \]  

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
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 \( \mu \) 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 \( \mu \) 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.
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 (i.e., 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 $\mu$ 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 $\mu$ 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.
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
### 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.

<table>
<thead>
<tr>
<th>Machine</th>
<th>num of proc</th>
<th>I/O time (seconds)</th>
<th>speedup</th>
<th>total run time (seconds)</th>
<th>percentage of total</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGI</td>
<td>1</td>
<td>0.64</td>
<td>1</td>
<td>98.3</td>
<td>0.65</td>
</tr>
<tr>
<td>KSR2</td>
<td>1</td>
<td>5.12</td>
<td>1</td>
<td>140.4</td>
<td>3.65</td>
</tr>
<tr>
<td>KSR2</td>
<td>2</td>
<td>2.58</td>
<td>1.98</td>
<td>140.4</td>
<td>1.84</td>
</tr>
<tr>
<td>KSR2</td>
<td>5</td>
<td>1.12</td>
<td>4.57</td>
<td>140.4</td>
<td>0.64</td>
</tr>
<tr>
<td>KSR2</td>
<td>10</td>
<td>2.40</td>
<td>2.13</td>
<td>140.4</td>
<td>1.71</td>
</tr>
</tbody>
</table>

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 \( \mu \) parallelization. Models that require extensive computation integrating over vertical atmospheric columns[WF94] may prefer \( \mu \) 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 \( \mu \) 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 (i.e., 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. 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 $\mu$ 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 $\mu$ 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 $\mu$ parallelization, assigning overlapping regions to processors. Such an assignment scheme could be explored with the TRANS model as well.

Acknowledgements

The research for this work was supported in part by funds from the following organisations and projects: UARS Grant G35X05UARS NASA Grant from the National Science Foundation under Grant ATM-8905901 In addition we would like to thank Greg Eisenhauer and Weiming Gu.

References


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 $X$ is given as:

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

(2)

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

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

(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 $\psi$ and a velocity potential $\chi$ ($\vec{k}$ is the vertical unit vector and the $\vec{\nabla}$ is taken on a constant pressure surface):

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

(4)

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

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

(5)

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

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:

\[
\frac{\partial X}{\partial t} = -(\vec{k} \times \nabla \psi) \cdot \nabla X + \nabla \cdot \nabla X - W \frac{\partial X}{\partial Z} + \frac{\partial}{\partial P} \left( \frac{K_p}{H_0^2} \frac{\partial X}{\partial Z} \right) + A_H \nabla^2 X
\]

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

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

\[
\frac{\partial X}{\partial t} = \frac{A}{\partial \lambda \partial \mu} - \frac{\partial \psi}{\partial \lambda} \frac{\partial X}{\partial \mu} + \frac{B}{(1 - \mu^2)} \frac{\partial \chi}{\partial \lambda} \frac{\partial X}{\partial \mu} + \frac{C}{(1 - \mu^2)} \frac{\partial \chi}{\partial \mu} \frac{\partial X}{\partial \mu} - \frac{D}{W \frac{\partial X}{\partial Z} + \frac{1}{H_0^2} \frac{\partial X}{\partial Z} + A_H \frac{1}{(1 - \mu^2)} \frac{\partial^2 X}{\partial \mu^2} + \frac{\partial \mu^2}{\partial \mu}}
\]

For further referencing the different terms have been named \( A \ldots D \). 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:

\[
F(\lambda, \mu, t) = \sum_{l=-\text{Max}}^{\text{Max}} \sum_{n=|l|}^{n=\text{Max}(l)} f_{l,n}(t) Y_{l,n}(\lambda, \mu)
\]

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

\[
Y_{l,n}(\lambda, \mu) = e^{il\lambda} P_{l,n}(\mu)
\]

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

\[
\lambda^{\text{Max}} \geq 3n\text{Max}
\]

To simplify the Fast Fourier Algorithm involved in the transformations usually \( \lambda^{\text{Max}} \) 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 \( n\text{Max} \) can be used to characterize the horizontal
resolution of a model, eg. T42 means a triangular truncation with $nMax = 42$, $\lambdaMax = 128$ 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:

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

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 $\mu$ and $\lambda$ are executed. Using this orthogonality the expansion coefficients $f_{l,n}(t)$ is easily calculated (if $F(\lambda, \mu, t)$ is known)

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

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

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

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):

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

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

where $\frac{\partial P_{l,n}(\mu)}{\partial \mu}$ are known functions of $\mu$ (time independent).
### PERSONAL SERVICES

<table>
<thead>
<tr>
<th>MONTH</th>
<th>BUDGET</th>
<th>EXPENDED</th>
<th>ENCUMBERED</th>
<th>FREE BALANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2,423.66</td>
<td>2,423.66</td>
<td>2,423.66</td>
<td>-898.71</td>
</tr>
<tr>
<td>TOTAL CONTRACT</td>
<td>20,850.25</td>
<td>19,325.30</td>
<td>2,423.66</td>
<td></td>
</tr>
</tbody>
</table>

### FRINGE BENEFITS

<table>
<thead>
<tr>
<th>MONTH</th>
<th>BUDGET</th>
<th>EXPENDED</th>
<th>ENCUMBERED</th>
<th>FREE BALANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>.00</td>
<td>1,862.38</td>
<td>.00</td>
<td>1,862.38</td>
</tr>
<tr>
<td>TOTAL CONTRACT</td>
<td>8,495.94</td>
<td>.00</td>
<td>.00</td>
<td></td>
</tr>
</tbody>
</table>

### MATERIALS AND SUPPLIES

<table>
<thead>
<tr>
<th>MONTH</th>
<th>BUDGET</th>
<th>EXPENDED</th>
<th>ENCUMBERED</th>
<th>FREE BALANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>.00</td>
<td>.00</td>
<td>44.50</td>
<td>-244.50</td>
</tr>
<tr>
<td>TOTAL CONTRACT</td>
<td>-200.00</td>
<td>.00</td>
<td>.00</td>
<td></td>
</tr>
</tbody>
</table>

### TRAVEL

<table>
<thead>
<tr>
<th>MONTH</th>
<th>BUDGET</th>
<th>EXPENDED</th>
<th>ENCUMBERED</th>
<th>FREE BALANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>.00</td>
<td>.00</td>
<td>1,130.00</td>
<td>1,130.00</td>
</tr>
<tr>
<td>TOTAL CONTRACT</td>
<td>5,000.00</td>
<td>.00</td>
<td>.00</td>
<td></td>
</tr>
</tbody>
</table>

### COMPUTER

<table>
<thead>
<tr>
<th>MONTH</th>
<th>BUDGET</th>
<th>EXPENDED</th>
<th>ENCUMBERED</th>
<th>FREE BALANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>.00</td>
<td>.00</td>
<td>504.38</td>
<td>.00</td>
</tr>
<tr>
<td>TOTAL CONTRACT</td>
<td>5,000.00</td>
<td>4,495.62</td>
<td>.00</td>
<td></td>
</tr>
</tbody>
</table>

### TOTAL DIRECT CHARGES

<table>
<thead>
<tr>
<th>MONTH</th>
<th>BUDGET</th>
<th>EXPENDED</th>
<th>ENCUMBERED</th>
<th>FREE BALANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2,423.66</td>
<td>14,249.70</td>
<td>3,598.16</td>
<td>13,855.97</td>
</tr>
<tr>
<td>INDIRECT CHARGES</td>
<td>MONTH</td>
<td>FISCAL YEAR</td>
<td>TOTAL CONTRACT</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------</td>
<td>-------------</td>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>BUDGET</td>
<td></td>
<td></td>
<td>14,485.13</td>
<td></td>
</tr>
<tr>
<td>EXPENDED</td>
<td>969.46</td>
<td>5,699.88</td>
<td>8,453.54</td>
<td></td>
</tr>
<tr>
<td>ENCUMBERED</td>
<td>-969.46</td>
<td>1,439.27</td>
<td>1,439.27</td>
<td></td>
</tr>
<tr>
<td>FREE BALANCE</td>
<td></td>
<td></td>
<td>4,892.32</td>
<td></td>
</tr>
</tbody>
</table>

| TOTAL                 |       |             | 53,631.32      |
| BUDGET                |       |             | 30,145.80      |
| EXPENDED              | 3,393.12| 19,949.58   | THIS ACCOUNT IS EXPENDED |
| ENCUMBERED            | -3,393.12| 5,037.43   | 5,037.43       |
| FREE BALANCE          |       |             | 18,448.29      |
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

January, 1996

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

List of Investigators:

Karsten Schwan, College of Computing
Fred Alyea, Earth and Atmospheric Sciences
M. William Ribarsky, College of Computing
Mary Trauner, Information Technology
Greg Eisenhauer, student, College of Computing
Yves Jean, student, College of Computing
Weiming Gu, student, College of Computing (PhD, 12/95)
Ray Wang, postdoc, Earth and Atmospheric Sciences
Jeffrey Waldrop, student, College of Computing
Beth Schroeder, student, College of Computing
Jeremy Heiner, student, College of Computing

List of associated investigators (not funded by NASA):

Thomas Kindler, student, Earth and Atmospheric Sciences (PhD, 12/95)
Dilma Silva, student, College of Computing
Jeffrey Vetter, student, College of Computing (funded by separate NASA GSRP grant.)
Vernard Martin, student, College of Computing
I. Scientific Application and Analysis

Scientific Progress

The three-dimensional, spectral transport model used in the current project was first successfully integrated over climatological time scales by Dr. Guang Ping Lou for the simulation of atmospheric N$_2$O using the United Kingdom Meteorological Office (UKMO) 4-dimensional, assimilated wind and temperature data set. A non-parallel, FORTRAN version of this integration using a fairly simple N$_2$O chemistry package containing only photo-chemical reactions was used to verify our initial parallel model results. The integrations reproduced the gross features of the observed stratospheric climatological N$_2$O distributions but also simulated the structure of the stratospheric Antarctic vortex and its evolution. A paper describing this work was presented at the Spring, 1994 AGU meeting (Lou, et al, 1995) and an enlarged version suitable for publication is currently in preparation (Lou, et al, 1996).

Subsequently, Dr. Thomas Kindler, who produced much of the parallel version of our model, enlarged the N$_2$O model chemistry package to include N$_2$O reactions involving O($^1$D) and also introduced assimilated wind data from NASA as well as UKMO. Initially, transport calculations without chemistry were run using Carbon-14 as a non-reactive tracer gas with the result that large differences in the transport properties of the two assimilated wind data sets were apparent from the resultant Carbon-14 distributions. Subsequent calculations for N$_2$O, including its chemistry, with the two input winds data sets with verification from UARS satellite observations have refined the transport differences between the two such that the model’s steering capabilities could be used to infer the correct climatological vertical velocity fields required to support the N$_2$O observations. During this process, it was also discovered that both the NASA and the UKMO data contained spurious values in some of the higher frequency wave components, leading to incorrect local transport calculations and ultimately affecting the large scale properties of the model’s N$_2$O distributions, particularly at tropical latitudes. Subsequent model runs with wind data that had been filtered to remove some of the high frequency components produced much more realistic N$_2$O distributions. A paper presenting these results and data limitations was given at the Fall, 1995 AGU meeting (Kindler, et. al, 1995) and, in more detail, in Dr. Kindler’s Ph.D. Thesis at Georgia Tech (Kindler, 1995).

During the past few months, the UKMO wind data base for a complete two-year period was processed into spectral form for model use. This new version of the input transport data base now includes complete temperature fields as well as the necessary wind data. This was done to facilitate advanced chemical calculations in the parallel model which often depend upon temperature. Additional UKMO data is being added as it becomes available.

How Interactive Steering Contributes To Scientific Applications

When combining on-line visualization and steering tools with our group’s parallel version of a global spectral atmospheric transport model, users have the unique ability to compare model results with observational data during the model run. Should discrepancies between model results and observations occur, model execution can be stopped, rolled back in time, model parameters may be changed, whereupon the user can then rerun the model with new parameter settings. Our experiences with this new approach in model validation are quite positive. Specifically, when applying these interactive validation methods to the scientifically relevant problem of simulating the global distribution and transport of Nitrous Oxide (N$_2$O), interesting scientific outcomes result from a comparison of results using
simulated windfields for transport versus using assimilated (measured) windfields for driving the transport inside the model. When comparing the results from using two different sets of assimilated windfields (NASA, UKMO), the model shows an underprediction of vertical mass transport in the equatorial area with the UKMO windfields and an overprediction of the vertical transport with the NASA winds. Online model interactions permitted us to adjust and “play with” vertical windfields to investigate in detail the sensitivity of the biased model results to changes in the vertical advection term. As a result and compared with observation data, model results were improved significantly for both windfield sets.

II. High Performance Computing: Implementation Summaries

Parallelization of the Atmospheric Modeling Code

The original atmospheric model was written in FORTRAN. To facilitate on-line monitoring, it was necessary to rewrite the code in C since the current FALCON system (which supports on-line monitoring) only supports the C programming language. An effort is underway to develop a version of FALCON that will work with other languages. (The FALCON system is described fully in the Infrastructure section.)

The parallelized model was targeted for two different computer architectures. The first approach was to target a shared memory machine model. A Kendall Square KSR 2 supercomputer was first chosen for this implementation. This shared memory model parallelizes the computations by both atmospheric level and by term in the Navier Stokes equation. When Kendall Square Research closed its doors, the model was ported to the Power Challenge Series of supercomputers manufactured by Silicon Graphics. Both machines run a variant of the UNIX operating system and support the shared memory paradigm so porting the software was of minimal effort.

The second approach was to target a distributed memory machine model called a message passing model. The machine used to implement the model was a group of three high end IBM workstations and an IBM SP-2. We used the MPI message passing library to communicate between the machines. Each processor was assigned work to do by a master process. Each of the slave processors calculated its portion of work, occasionally communicating with other processor to get needed information. Results were sent back to the master processor at the end of each timestep. In addition to the responsibility of gathering information for the entire application, the master processor also was given a portion of work to do. This distributed memory model parallelizes the application by level only in order to keep communication costs down.

On-line Program Monitoring and Steering

The monitoring is accomplished by inserting what we call “sensors” into the actual code at compile time. Sensors are declared earlier by the programmer and are used to gather interesting information about a program’s state at a particular moment. At run time, when the code encounters a “sensor”, the sensor will gather up whatever information it needs and will send that information to a centralized routing station called a “hub”. The hub is a program, usually running on another machine, that gathers information and stores or forwards it to other applications as necessary. For our system, the visualization system connects to the “hub” to request the monitoring information about the application. Although it is theoretically possible to have the application communicate directly to the visualization, the hub provides more functionality in that it can offload work to another processor and it
provides facilities for allowing an arbitrary number of applications to connect and access the same monitoring information without affecting the application in any way. Also, the hub allows us to easily provide a communications interface for the steering function that will accommodate multiple steerable components without unduly affecting the running time of the application.

To steer the application, special sensors are inserted into the application that check for steering commands from the hub. If a command is waiting to be received from a hub, then the sensor interprets the steering command and modifies the application state accordingly. In most cases, steering is accomplished by first stopping the application from proceeding further in its simulation, changing the program state, and then re-starting the application so that it may continue with the new state information. This start/modify/stop sequence is done to insure that all parts of the application are synchronized and have the necessary state information so that the model is in a consistent state and thus the calculations are consistent as well. (Monitoring and steering are described fully in the Infrastructure section.)

Our applications include monitoring sensors for the wind fields and for the concentrations of various (single) chemical constituents. A selectable 2D or 3D interactive visual interface allows the scientist to move through the data at each timestep using various projections as desired. Steering sensors allow the scientist to evaluate and test new values for the wind fields in conjunction with simple checkpoint and restart facilities which are required to assure stable and accurate simulation behavior. As mentioned above, the spurious wind values were discovered through the use of this interface. (The visualization system is described fully in the Visualization section.)

Parallel Code Performance Evaluation

This work concerns the parallel implementation of a grand challenge problem: global atmospheric modeling. The novel contributions of our work include: (1) a detailed investigation of opportunities for parallelism in atmospheric transport based on spectral solution methods, (2) the experimental evaluation of overheads arising from load imbalances and data movement for alternative parallelization methods, and (3) the development of a parallel code that can be monitored and steered interactively based on output data visualizations and animations of program functionality or performance. Code parallelization 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).

Performance measurements are performed on a 64-node KSR2 supercomputer. However, the parallel code has been ported to several shared memory parallel machines, including SGI multiprocessors, and it has now been ported to several modern platforms, including the IBM SP-2 machine, the SGI Powerchallenge, and workstations. Additional performance evaluation is ongoing on those machines.

III. Infrastructure

In order to enable our integrated approach to simulation and visualization/analysis, we have developed Falcon, a toolkit that collectively supports on-line monitoring, steering,
visualization, and analysis of parallel and distributed simulations. The general usefulness of
the toolkit is demonstrated by its diverse application to areas such as interactive molecular
dynamics simulation and interactive simulation of fault containment strategies in
telecommunication systems. It is anticipated that the Falcon toolkit will be available for
distribution on the WWW in the near future. The Falcon toolkit includes sensors and
probes, an on-line steering system, DataHub for routing information from a server to
multiple clients, and Portable Binary I/O (PBIO) for transmitting binary data between
machines.

Sensors, probes, and steering objects inserted in the simulation code are generated from
monitoring and steering specifications. Their partially analyzed monitoring information is
sent to graphical and visualization displays. Once steering decisions are made by the user,
changes to the application's parameters and states are made by Falcon's steering
mechanism which invokes the steering objects embedded in the application code.

Falcon's on-line steering component consists of a steering server on the target machine that
performs steering, and a steering client that provides the user interface and control facilities
remotely. The steering server is typically created as a separate execution thread of the
application to which local monitors forward only those monitoring event that are of interest
to steering activities. The steering client receives application run-time information from the
application, displays the information to the user, accepts steering commands from the user,
and enacts changes that affect the application's execution. Communication between
application and steering client and steering client and server is handled by the transmission
tool, DataHub.

DataHub is a transmission tool for routing messages between multiple clients where clients
can be broadly classified as applications, visualization/analysis/steering tools, or other
DataHubs. Messages are identified by their format names and registered with DataHub by
both senders and receivers. When a message is received, it is routed to those clients who
have registered their interest in receiving that message type. Communication is done either
through sockets or file I/O. The hub server can provide additional functionality such as
event reordering before data is routed to clients. DataHub and PBIO taken together provide
a flexible display system for attaching different types of graphical and visualization displays
to an application's execution. Graphics intensive clients, which run on high performance
front-end workstations to take advantage of better graphics and visualization support, can
be dynamically attached to and detached from the display system.

The program steering environment demands speed and compactness of binary data
transmission in a heterogeneous environment. These needs are met by Portable Binary I/O
(PBIO), a set of services for transmitting binary data between machines in heterogeneous
environments. PBIO provides a low overhead service by not requiring data to be translated
into a "standard" or "network" representation and portability by transferring data between
machines despite differences in byte ordering, sizes of datatypes, and compiler structure
layout differences.

Though PBIO uses a metaformat in which the actual formats of binary records could be
described, the representation of the metadata is hidden. Writers of data provide a
description of names, types, sizes, and positions of fields in records through calls to the
PBIO library. Readers provide similar information. No translation is done on the writer's
end; meta information describing the senders format is sent in the PBIO data stream. On
the reader's end, the format of the incoming data is compared with the format the reading
program expects. Where discrepancies exist, PBIO performs the appropriate translations.
IV. Visualization

We have integrated the Glyphmaker visualization system, including modules developed within the Iris Explorer environment, with the Falcon steering system and the atmospheric model. As the model generates timesteps, the visualization is updated in an on-line fashion. Additions to the visualization capability include modules to immediately display the data or to pass it along to PV-Wave for alternative visualizations and analysis.

By direct manipulation steering we mean that we can interact directly with visualizations of atmospheric simulations to alter the future course of the simulations. We do this, for example, by scaling, rotating, translating, or inputting data for graphical objects bound to the data. Thus we could use the conditional box (a tool from Glyphmaker) to define spatial regions in the data where one could change chemical concentrations or other parameters. We could also employ data probes from Glyphmaker to locate localized behavior of interest and to adjust parameter values where desired. We have extended the rendering module in Glyphmaker to support these direct manipulation capabilities. (The standard rendering module in Explorer is not nearly flexible enough for our purposes, so we have made extensive modifications to it using SGI Inventor.) Our direct manipulation techniques involve interactions with both 3D and 2D representations of the data. This hybrid approach is attractive because it recognizes that while new and innovative methods are necessary to explore spatially complex and multidimensional data or to control simulations that produce these data, familiar tools such as 2D plots are succinct ways of expressing user intent.

In our current version of the visualization/analysis tools, we have added a graphical steering mechanism to our Glyphmaker visualization system. The system allows the user to select from a set of geometric forms. The user can then deform the geometry to encompass a desired spatial region, within which one can change parameters in the atmospheric model. In addition to interactive control of position and deformation of the geometric steering objects, the user receives visual feedback from both the steering object’s geographic position and from the model datastructure indices. The visual feedback is enhanced by allowing the user to choose from a variety of projections (spherical or flat) with the graphical attributes of the geometric form adjusted to the type of projection.

The direct manipulation steering approach is a new and powerful way to control spatially complex and dynamic simulations, such as those from atmospheric models. It allows the user to do side-by-side probing and analyzing of the correlations in the data while being able to redirect the simulation in a spatially intuitive way to better understand how the physical processes evolve. It requires the capability for direct, quantitative probing of data that we have built into Glyphmaker through the formulation of elaborate data structures that always connect the visual representations to the original data, allowing investigation down to the individual datum. The data structures change and expand dynamically as new bindings between visual representations and data are made. The steering also requires a close coupling with the steering control and data transfer mechanisms provided by Falcon. The first stages of this integration has been completed and future development will require updating of the Falcon system to respond to new needs placed on it by enhancement of the visual interface as well as modification of the modes of visual interaction necessitated by improvements in the Falcon system.

The flexibility of the Glyphmaker system allows the use of the steering objects for analysis as well. The data elements within the region could be reclassified with their own glyphs (e.g., with different shapes or colors than the surroundings) so that their behavior could be highlighted and followed in detail. We have added the capability to take these selected data
and list any values or show their distribution in 2D plots. This is the process of mixing 3D visualizations with 2D quantitative analyses that we mentioned above.

During the reporting period, we have also worked closely with atmospheric scientists. We instrumented the atmospheric model with a mechanism for deferred steering. Our design allows model changes to be scheduled rather than applied immediately. This is necessary because the parallelized execution is kept efficient by minimal synchronization. Additionally we focused on steering the vertical windfields. The windfields are an important transport mechanism and are derived from observed data. Our steering system permits both human interactive control and automated input from weather data sources (satellites, etc.).

V. Funding and Indirect Effects

Our work should have broad relevance not only for atmospheric scientists but also for other members of the NASA community who must look at or control the production of large amounts of dynamic, multidimensional data. The direct manipulation steering mechanisms especially form a new set of tools for controlling simulations where the output is 3D, complicated, and dynamic. These tools are even more important for understanding atmospheric processes since they evidently depend on the detailed behavior of highly correlated variables. Thus the steering tools are of particular interest to the atmospheric scientists with which we work and should engender broader interest as well.

Specific impacts include the awarding of a grant from IBM to work on real-time visualization and on the development of hierarchical visualization schemes for navigating and exploring very large datasets. This research will use a heterogeneous environment consisting of an SP-2 multiprocessor machine and graphics workstation. In addition we have written proposals for the DOD Multidisciplinary University Research Initiative and for other programs based, in part, on this work. Finally the visualization/analysis and steering tools were presented during the site visit on a proposal from the College of Computing for an NSF Infrastructure Grant. The grant was subsequently awarded based, in part, on a demonstration of broad-based, innovative, and collaborative efforts using computing.

By leveraging the NASA grant, the investigators were able to acquire two major equipment grants from the National Science Foundation, entitled:


These grants are helping us direct our efforts into the following new directions.

The continuing merger of computer and communication technologies is leading to a new computing/communications infrastructure of unprecedented magnitude, enabling new applications with broad economic and social impact. Yet, such applications pose major challenges to researchers in Computer Science and in application domains. We are constructing an infrastructure consisting of computer and communication equipment to
support collaborative experimental research among five research projects that attack some of these challenges. These projects are closely integrated and build on each other. As concrete demonstrations of our joint work, we will realize systems from two different “driver” applications that integrate research results, system and networking software, and tools from each of the five projects.

The topic of our joint research is the realization of distributed laboratories, where individuals can interact with each other, and more importantly, with powerful, distributed computational tools as readily as if all were located in a single site. Our intent is to permit scientists, engineers, and managers at geographically distinct locations (including individuals “tele-commuting” from home) to combine their expertise in solving shared problems, by allowing them to simultaneously view, interact with, and steer sophisticated computations executing on high performance distributed computing platforms. The research results and tools resulting from these efforts will have broad application in many domains. However, we are using two specific applications to focus our efforts, and to help ensure that our results and software tools are properly integrated:

- A distributed laboratory for experimentation with high performance numeric computations for applications in molecular physics, atmospheric sciences, working with high performance atmospheric and pollution modelling, and manufacturing systems.
- A distributed laboratory for studying the behavior of future-generation, large-scale telecommunication networks through high performance parallel and distributed simulation models of wired and wireless networks, called the virtual telecommunication networks application.

The NASA application and some of the infrastructure developed with this grant will also be employed in a major new ARPA-funded effort:


VI. Papers and Presentations

This section provides a summary of papers and presentations given at a number of conferences and meetings around the country. Abstracts and descriptions are provided for detail and clarification.

1. AUG Spring, 1995 meeting.

Reference:


Abstract:

1 (now at General Services Corporation, Data Assimilation Office, GSFC, NASA, 7501 Forbes Blvd., Suite 200, Seabrook, MD 20706; ph: 301-805-6996; e-mail: glou@dao.gsfc.nasa.gov)
3-D Simulations of N₂O Transport and Antarctic Vortex Evolution

This study focuses on three areas: (a) the structure of the stratospheric Antarctic vortex and its evolution; (b) the transport of N₂O and dynamical forces that dominate these processes; (c) the climatology of the N₂O mixing ratio distribution and its driving factors. A 3-dimensional spectral chemical transport model was employed to simulate N₂O transport and study the driving forces that affect the processes. The dynamical driving fields are from the UKMO 4-dimensional assimilated data set. UARS CLAES N₂O mixing ratio are used for the N₂O initial conditions. Model results show that the N₂O distribution and transport closely resemble the CLAES measurements, especially at high latitudes. The correlation coefficients between CLAES N₂O temperatures, and model N₂O and temperatures are remarkably similar in terms of their meridional distributions. Diagnostic study and model simulation results reveal that while large-scale Eulerian mean vertical motion fields are upward inside the vortex, the mean residual circulation vertical velocity is downward. The monthly mean maximum sinking residual velocity is -0.40 cm/s at about 1.5 mb and -0.07 cm/s in the 30-90 mb layer inside the Antarctic vortex in September. The vortex first breaks in the upper stratosphere during September. Then the breaking process propagates downward to the 3-10 mb level in the middle of October. At the lower levels, 10-20 mb, the vortex breaks up in early November. These breaking processes continue to penetrate to lower levels at about 20-30 mb by late November. In the meridional transport of N₂O, eddy transport is the chief process. Especially at higher altitudes, there seems to be persistent eddy mixing going on at the middle latitudes during the early spring. However, the residual circulation transport dominates the long term vertical mixing. The bulge of the elevated N₂O mixing ratio in the tropical stratosphere is determined by the uplifting of mass by the residual circulation. During the Southern Hemisphere summer, the uplifting of N₂O by the residual circulation reaches above 1 ppb/day. The downward transport inside the vortex can exceed 2 ppb/day in the winter hemisphere. The climatological distribution of the N₂O mixing ratio follows the seasonal variations of the solar radiation. The bulge of the elevated N₂O shifts toward the summer hemisphere by up to 15 degrees in latitude. The slopes of the N₂O mixing ratios are sharper in the winter hemisphere and the surf zone is well defined in the middle latitudes on the zonal mean plots.


2. AUG Fall, 1995 Meeting

Reference:

Abstract:

A Comparison of CLAES N₂O Simulations using 3D Transport Models Driven by UKMO and GSFC Assimilated Winds

A three dimensional chemical model has been developed. The model has a vertical resolution of approximately 1.25 km (on-half a UARS layer) and is spectrally truncated at T21. In this paper we will compare N₂O simulations from two calculations in which the model is driven by the windfields provided by the assimilation models of UKMO and GSFC. The calculations were initialized on September 1, 1992 with a distribution based on UARS CLAES N₂O measurements and were run for 13 months. The zonal mean gradients of N₂O are found to steepen using the GSFC wind fields whereas they flatten out using the UKMO fields (as we have previously reported). Consequently the calculated atmospheric lifetime of N₂O changes from 180 years initially to less than 100 years and longer than 200 years respectively using the GSFC and UKMO winds. The budgets of N₂O in the two calculations will be compared in terms of contributions by the residual mean circulation and mixing along isentropes. The degree of isolation of the polar vortices and the extent of iteration between the tropics and the extratropics will also be examined using area mapping analyses.


3. Supercomputing '95, GII Testbed

Reference:


Abstract:

A Parallel Spectral Model for Atmospheric Transport Processes

Earth and atmospheric scientists at Georgia Tech have developed a global chemical transport model that uses assimilated windfields for the transport calculations. These models are important tools to answer scientific questions about the stratospheric-tropospheric exchange mechanism or the distribution of species such as chlorofluorocarbons, hydrochlorofluorocarbons, and ozone. This model uses a spectral approach common to global models to solve the transport equation for each species.

Ideally, in large-scale atmospheric simulations, the observational database should be closely coupled to the visualization/analysis process. In fact, there should be feedback in the form of steering between the latter and the simulation in order to yield more accurate representations of atmospheric processes and a significantly more focused investigation. Because the data have complicated 3D
structures and are highly time-dependent, the visualization approach must handle this dynamic data in a highly interactive fashion.

In this project, the researchers have combined all these aspects into a single, integrated approach. This has required a collaborative, interdisciplinary process involving atmospheric scientists and experts in high-performance parallel computing, visualization, and user interfaces. The process used here could serve as a template for building highly effective and powerful applications (and tools supporting them), a process where the developer comes away with a deeper understanding of user needs.

Discussion:

This application was accepted for execution over the GII testbed and visualization on the I-Way Wall.

A working prototype of the distributed memory model with visualization and minor steering was exhibited. The atmospheric transport model was running on 32 nodes of the IBM SP-2 supercomputer at the Cornell Theory Center. On-line monitoring data was shipped over a dedicated ATM network to San Diego to an SGI Challenge server which acted as a centralized resource manager and router (the Datahub.) The custom visualization was running on an SGI Onyx connected to the Wall.

The prototype allowed for the user to interactively (while the model is running) view both the wind fields and the N₂O concentrations at any part of the globe in a variety of interesting formats including spherical levels extending from the earth’s surface into the stratosphere, a flat map Cartesian view with strict longitudinal and latitudinal planes, and simply x-y plots. The data viewed could be chosen via explicit selection or relative position using sliding bars and dials.

4. Penn State University

Reference:


5. Workshop on Debugging and Performance Tuning for Parallel Computing Systems

Reference:

VII. Looking Ahead

The Science and High Performance Computing

A more direct method of transforming the UKMO and NASA data to spectral form is being developed that will not require linear interpolation processes to "move" data from one grid system to a different one for spectral transformation. Although the interpolation process that has been used to date is not thought to contribute in any important way to the introduction of any spurious high frequency waves to the data, in view of the now-known existence of such waves in the wind data base, it is thought that the elimination of any potential high frequency noise that may be introduced numerically in preparation for transformation be undertaken. The distribution of energy as a function of spatial resolution for the transformed assimilated data base will then be compared with observational data in order to delineate the frequencies that contain spurious values.

A major upgrade of the parallel model that is currently under way involves the simultaneous integration with a number of atmospheric species and the inclusion of the necessarily complex chemical packages that will be required. For this purpose, we propose to make use of a substantially modified version of a large atmospheric chemical model obtained from the "Laboratoire de Physique et Chimie de l'Environnement", CNRS, Orleans, France. This model is to be included as a separate module linked and interacting with the current parallel transport model and should thus permit state-of-the-art simulations of stratospheric mixes of important atmospheric constituents.

Minor changes to the parallel model that are planned for the next few months include the installation of new fourth-order numerical scheme for the spectral vertical diffusion calculations and the introduction of wind data at the lowest model levels to better simulate the effects of the Earth's surface boundary layer.

The infrastructure grant concerns tool development and distributions, especially focussing on steering and its use for scientific processors, with extensions of these tools to address entire distributed laboratories.

The Visualization

We are extending Glyphmaker in ways to increase its power in the analysis of atmospheric simulations that will grow significantly in size and complexity as the parallel approaches are scaled up. It will be necessary to manage levels of detail in the visualizations so that we can retain highly interactive exploratory analysis as the data grows. This will require both automatic and user-directed methods, since the user will not know at the outset what the data contains but will want to direct and refine the visualization process. We are working on general methods for detail management that are based on an understanding of the nature of physical data and that include both approaches for 3 and 4D (including time) pattern recognition and for feature recognition and extraction. These approaches will allow a natural organization of the data for further study including higher level visualization (e.g., surface and volumes) of general unstructured or scattered data. These approaches will also permit us to represent the data with visual abstractions at multiple levels of complexity. We will work closely with application scientists so that the visual abstraction process matches the physical abstraction process that they use to simplify and then understand their data.

We plan to extend the visual representations and interactions for steering. One extension will allow the user to specify distribution functions with a few parameters so that, for
example, more physically accurate concentration profiles can be inserted into the
simulation. Thus the user can easily specify how model changes are distributed within the
extent of the steering object. Also, we are incorporating the ability to acquire steering
specifications from the visualization output. For example, if an isosurface specification
produces a surface in the visualization, we will be able to use the surface as a spatial
parameter for steering. We plan to write a paper shortly on our present and some of our
new steering capabilities.

In order to achieve our ultimate goal of real-time exploratory visualization, steering and
control of simulations, regardless of the size of data output, we must investigate
alternatives to our present visualization approach. Among other things, this means looking
at tools other than SGI Iris Explorer and Inventor. The reason for this is that we must have
fast rendering of thousands of potentially independent objects; neither Explorer or Inventor
are optimized for this case. We are considering, for example, the use of the CAVE libraries
from NCSA. These are built for scientific visualization in immersive virtual environments.
They thus are built for real-time use, have been employed on big data, and have some tools
for exploratory navigation built in. By integrating the CAVE libraries with Open Inventor,
we can retain several of our interaction and direct manipulation tools. As an alternative, we
are also considering building our own renderer using OpenGL. This will give us optimal
efficiency and control over visualization capabilities. However, we will have to rebuild
most of our interaction capabilities and some of our visualization techniques.

Whichever path we take for our rendering tools, we will move them from GL to OpenGL.
This coupled with use of libraries like Open Inventor will make available a large number of
platforms for use by our system.

*Collaborative Steering*

We plan to incorporate support for collaborative work in the monitoring/steering
infrastructure beyond the simple example of replicating the pixels of a visualization on
several workstations’ screens. Support will be needed to allow the collaborators to have
different views of a single visualization (or possibly different visualizations of the same
data) and to coordinate the steering interactions and feedback among the views. For
rendering the visualizations we use an object-oriented graphics library which allows one to
arrange objects into a tree structure to describe a scene. This library includes several
objects which respond to user input (mouse, keyboard, etc.) which we use for steering.
To support collaboration we add a mechanism to this library which maintains consistent
copies of the scene tree structure on two or more machines.

**VIII. Bibliography**

Lou, Guang Ping, Fred Alyea, and Derek Cunnold, “N\textsubscript{2}O transport dynamics and its
climatology”, in preparation, 1996.

Kindler, Thomas Paul. “The Development of Supercomputing Tools In a Global Chemistry
Transport Model (CTM) and Its Application to Selected Problems in Global Atmospheric

Yves Jean, Thomas Kindler, William Ribarsky, Weiming Gu, Gregory Eisenhauer,
Karsten Schwan, and Fred Alyea, “An Integrated Approach for Steering, Visualization,
Visualization ’95*, pp. 383-387.
In this research we have constructed a tightly coupled set of methods for monitoring, steering, and applying visual analysis to large scale simulations. This work shows how a collaborative, interdisciplinary process that teams application and computer scientists can result in a powerful integrated approach. The integrated design allows great flexibility in the development and use of analysis tools. This work also shows that visual analysis is a necessary component for full understanding of spatially complex, time-dependent atmospheric processes.


In this paper we present a general data organization for exploratory visualization/analysis. It supports mappings between (user-defined) visual representations and the original data. We present tools that take advantage of this organization and allow selection of data using spatial or other constraints. These selected data can be reclassified for binding to new visual representations (or hidden from view). The selection and binding is performed in an intuitive manner by direct manipulation so that even users who are not graphics experts can do it. The users then have easy-to-use tools to calculate new quantities from selected variables and thus to formulate new data structures. It is important to put these intuitive and easy-to-use tools directly in the hands of users who, though not graphics experts, are experts in their own fields and thus are the best ones to control the visualization/analysis process.


Ph.D., Jeffrey Vetter, “Interactive Steering of Large-Scale Parallel Applications” (in progress).


Ph.D., Vernard Martin, “Performance of Heterogeneous Parallel Applications” (in progress).
Annual Report:

Parallelization and Visual Analysis of Multidimensional Fields: Application to Ozone Production, Destruction, and Transport in Three Dimensions

Grant No. NAGW-3886

Karsten Schwan

December, 1997

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

List of investigators:
Karsten Schwan, College of Computing
Fred Alyea, Earth and Atmospheric Sciences
M. William Ribarsky, College of Computing
Mary Trauner, Information Technology
Jeremy Heiner, PhD student, College of Computing
Milind Nanal, PhD student, College of Computing
Beth Schroeder, PhD student, College of Computing
Jeff Waldrop, MS student, College of Computing
Ray Wang, postdoc, Earth and Atmospheric Sciences
Song Zou, PhD student, College of Computing

List of associated investigators:
Greg Eisenhauer, Research Scientist II, College of Computing
Vernard Martin, PhD student, College of Computing
Jeffrey Vetter, PhD student, College of Computing
Derek Cunnold, Senior Research Scientist, Earth and Atmospheric Sciences

Undergraduate Students:
Onome Okuma, CS, Robert Sumner, CS
Davis King, CS, Roman Khramez, CS
1 Introduction

2 Scientific Application and Analysis

2.1 Research Contributions to Atmospheric Sciences

As part of this research, we have constructed a parallel, three-dimensional, spectral dynamical/chemical transport model. This model's ability to run on most modern parallel machines and also across distributed compute engines has led to several advances in atmospheric science research, in part due to the improved computational performance offered by this model compared to its previous sequential version and in part due to the online monitoring, steering, and visualization capabilities included with its parallel realization. Last, most recently, the transport model has been enlarged to simultaneously incorporate a number of additional atmospheric chemical constituents.

During the past 12 months, R. Wang and Derek Cunnold were the major contributors to this effort, following the work of T. Kindler and Derek Cunnold who, along with F. N. Alyea and G. P. Lou, developed the original dynamical/chemical model. Due to their efforts, for the first time, it is possible for the full capabilities of the real-time visual rendering of the model variables to be utilized to study the interactions of different chemical constituents in the atmosphere and to simulate both local and global changes in one or more of the constituents which might be due to bomb bursts, volcanoes or other fairly rapid processes. This is accomplished through the "program steering" features built into the visual package. Recent research results attained with this model and the chemical constituents concern insights concerning atmospheric transport reported in [2] and comparative evaluations of NASA vs. UKMO observational data sets [1]. During the past year, the bulk of our effort, has involved the parallelization and substantial modification of a complex chemical model obtained from M. Pirre of the Laboratoire de Physique et Chimie de l'Environnement, CNRS, Orleans, France [3] and integration of this code into the transport package. This model incorporates 31 chemical compounds involving 60 gas phase reactions and 15 photochemical reactions, all of which are evaluated at each time step on a global basis, either on the model's three-dimension grid mesh or in the spectral domain.

2.2 Papers


3 High Performance Computing

3.1 Research Contributions

The atmospheric modeling application and its usage by earth and atmospheric scientists here at Georgia Tech has given rise to exciting new research streams in high performance research: (1) a focus on complete experimentation environments or laboratories rather than the broader HPC community's previous focus on individual, large scale parallel and distributed applications, and (2) a focus on the online interaction with computational tools in these environments. (1) has led to the creation of a broader, multi-year and multi-faculty research efforts, called the 'Distributed Laboratories' effort. This effort's highly visible research has involved the creation of a National Advisory Board to set its direction (comprised of both industry and
academic researchers), it has led to the creation of a state of the art computing facility at Georgia Tech (with funding from the National Science Foundation and from industry), it has resulted in strong multi-faculty research efforts addressing topics ranging from high performance distributed object/agent applications to topics addressing high performance communications. Most recently, it has resulted in the inclusion of Georgia Tech in the NCSA center's renewal effort as a member of Team C as well as in Intel's high performance computing initiative (involving the donation of approx. $4,000,000 of Intel equipment over three years). (2) has led to a strong research stream in program steering, including research on novel visual interfaces for program steering and including research on the middleware infrastructure required for online monitoring and steering. All of these efforts are ongoing and are currently funded by the National Science Foundation, DARPA, other branches of the DoD, and by industry.

Research efforts being undertaken at Georgia Tech that address the topic of distributed laboratories consist of:

- Steering and monitoring tools and infrastructure used in the online observation and manipulation of two scientific computations developed jointly with end users.
- Middleware to transport the on-line monitoring and steering events. Can be changed dynamically to adjust event streams to current system loads and monitoring/steering needs.
- Visualization support to permit the definition of appropriate visual abstractions and their efficient representation on 2D and 3D graphical displays.
- Collaboration infrastructure and abstractions.

3.2 Papers


Greg Eisenhauer, Beth Schroeder, Karsten Schwan, Vernard Martin, and Jeff Vetter, DataExchange: High Performance Communications in Distributed Laboratories, 9th Int'l Conference on Parallel and Distributed Computing and Systems, October, 1997.


Greg Eisenhauer, Weiming Gu, Eileen Kraemer, Karsten Schwan and John Stasko, Online Displays of Parallel Programs: Problems and Solutions, accepted for publication, Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '97), 1997.

Greg Eisenhauer, Beth Schroeder and Karsten Schwan, From Interactive High Performance Programs to Distributed Laboratories: A Research Agenda, Proceedings SPDP'96 Workshop on Program Visualization and Instrumentation, October 1996.


4 Visualization

4.1 Research Contributions

We have developed a heterogeneous environment for visual steering of computer simulations. Visual steering implies two-way communication through 2D and 3D graphical data representations to bring about
user involvement with the calculations as they occur. One must be able to insert on-the-fly parameter changes, even over 4D (3D + time) regions of the simulation and see results dynamically updated. The environment differs from previous steering systems in that it combines (i) a high level set of tools for instrumenting simulations; (ii) tools for efficient data exchange; and (iii) a visual steering interface for direct interaction with visualizations of the data. The instrumenting tools can be inserted by users into their codes without having to know about networking protocols or machine-dependent binary formats. The visual interface allows visualization of a variety of data from different applications since it employs general graphical objects that users can map to variables of their choice. The steering tools are integrated with the middleware that controls and monitors the distributed simulation and with the visualization/analysis tools so that one can apply steering commands simply by knowing the I/O formats of the instrumentation and without knowledge of the middleware structure. Employing the instrumentation and I/O libraries, users can reconfigure their application codes to show new variables or accept different steering controls. This information is transmitted automatically to the visual steering environment. Since the steering tools appear in the data visualization space, users can steer based on the detailed 3D structure and time-dependent behavior of the data. The visualization/analysis tools can bring out correlations between variables or can focus on selected regions of the data; these analyses can be used to refine the steering or more closely investigate the results of the steering input. Finally, the same visualizations can function both as data representations and as steering inputs. We demonstrate this with isosurfaces. These may define regions of physical interest but irregular geometry where spatial distributions of selected variables can be inserted.

4.2 Papers


References


Final Report:

Parallelization and Visual Analysis of Multidimensional Fields: Application to Ozone Production, Destruction, and Transport in Three Dimensions

Grant No. NAGW-3886

Karsten Schwan

October, 1997

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

List of investigators:
Karsten Schwan, College of Computing
Fred Alyea, Earth and Atmospheric Sciences
M. William Ribarsky, College of Computing
Mary Trauner, Information Technology
Jeremy Heiner, PhD student, College of Computing
Milind Nanal, PhD student, College of Computing
Beth Schroeder, PhD student, College of Computing
Jeff Waldrop, MS student, College of Computing
Ray Wang, postdoc, Earth and Atmospheric Sciences
Song Zou, PhD student, College of Computing

List of associated investigators:
Greg Eisenhauer, Research Scientist II, College of Computing
Vernard Martin, PhD student, College of Computing
Jeffrey Vetter, PhD student, College of Computing
Derek Cunnold, Senior Research Scientist, Earth and Atmospheric Sciences

Undergraduate Students:
Onome Okuma, CS, Robert Sumner, CS
Davis King, CS, Roman Khramez, CS
1 Introduction

This final report has four sections. We first describe the actual scientific results attained by our research team, followed by a description of the high performance computing research enhancing those results and prompted by the scientific tasks being undertaken. Next, we describe our research in data and program visualization motivated by the scientific research and also enabling it. Last, we comment on the indirect effects this research effort has had on our work, in terms of follow up or additional funding, student training, etc.

2 Scientific Application and Analysis

2.1 Research Contributions to Atmospheric Sciences

As part of this research, we have constructed a parallel, three-dimensional, spectral dynamical/chemical transport model. This model's ability to run on most modern parallel machines and also across distributed compute engines has led to several advances in atmospheric science research. These advances are due in part to the improved computational performance offered by this model compared to its previous sequential version and in part to the online monitoring, steering, and visualization capabilities included with its parallel realization. Last, most recently, the transport model has been enlarged to simultaneously incorporate a number of additional atmospheric chemical constituents.

During the past 12 months, R. Wang and Derek Cunnold were the major contributors to this effort, following the previous work of T. Kindler, Derek Cunnold and Fred Alyea who, along with G. P. Lou, developed the original dynamical/chemical model and its sequential implementation. Due to their efforts, for the first time it is possible for the full capabilities of the real-time visual rendering of the model variables to be utilized to study the interactions of different chemical constituents in the atmosphere and to simulate both local and global changes in one or more of the constituents. Such changes might be due to bomb bursts, volcanos or other fairly rapid processes. This is accomplished through the "program steering" features built into the visual package.

Initially, to test the model calculations, two constituents which do not chemically interact were selected, and their time dependent global solutions were compared with single constituent model runs. The single constituent version of the model is a scientifically somewhat limited but computationally very efficient dynamical/chemical transport model and has been used for investigations into the stratospheric properties of N2O, Carbon 14 and associated dynamical flow fields [8, 6, 5]. Interesting research results attained with this model and the chemical constituents concern insights concerning atmospheric transport reported in [8] and comparative evaluations of NASA vs. UKMO observational data sets [7].

During the past year, our efforts (which continue today), involve the parallelization and substantial modification of a complex chemical model obtained from M. Pirre of the Laboratoire de Physique et Chimie de l'Environnement, CNRS, Orleans, France [10], and the integration of this code with the parallel/dynamical transport model. The resulting, complex dynamical/chemical model of the earth's atmosphere incorporates 31 chemical compounds involving 60 gas phase reactions and 15 photochemical reactions, all of which are evaluated at each time step on a global basis, either on the model's three-dimension grid mesh or in the spectral domain. Preliminary computations with this new tool have been made for model evaluation purposes. Due to the complexity of the physical and chemical problems, it is expected that more such evaluations will be required.

The new 3-D chemistry transport model is designed for realistic stratospheric ozone simulations but problems dealing with other constituents, methane for example, can also be studied. At present, we are conducting a case study of the ozone Quasi-Biennial Oscillation (QBO) during both the easterly and westerly phases. As part of this work, a relationship between the ozone and methane in the stratosphere that may be looked at with the 3-D chemistry transport model is the link between the observed asymmetric ozone depletion in the upper stratosphere and hemispheric differences in methane distributions.
2.2 Papers


3 High Performance Computing

Current communication tools and libraries for high performance computing are designed for platforms and applications that exhibit relatively stable computational and communication characteristics. In contrast, additional complexities exist in terms of dynamic behaviors for (1) mixed environments in which high performance applications interact with multiple end users, visualizations, storage engines, and I/O engines - termed 'distributed laboratories' in our research - and (2) high performance collaborative computing applications in general. The atmospheric modeling application and its use by earth and atmospheric scientists here at Georgia Tech have given rise to exciting new research streams in high performance research: (1) a focus on complete experimentation environments, or laboratories, rather than the broader HPC community's previous focus on individual, large scale parallel and distributed applications, and (2) a focus on the online interaction with computational tools in these environments. The first has led to the creation of a broader, multi-year and multi-faculty research effort, called Distributed Laboratories. This effort's highly visible research has involved the creation of a National Advisory Board to set its direction (comprised of both industry and academic researchers), it has led to the creation of a state of the art computing facility at Georgia Tech (with funding from the National Science Foundation and from industry), and it has resulted in strong multi-faculty research efforts addressing topics ranging from high performance distributed object/agent applications to high performance communications. Most recently, it has resulted in the inclusion of Georgia Tech in the NCSA center's renewal effort as a member of Team C as well as in Intel's high performance computing initiative (involving the donation of approximately $4,000,000 of Intel equipment over three years). The second research stream has led to strong research in program steering, including research on novel visual interfaces for program steering, the use of immersive interfaces for steering complex scientific applications, and research on the middleware infrastructure required for online monitoring and steering. All of these efforts are ongoing and are currently funded by the National Science Foundation, DARPA, other branches of the DoD, and by industry support.

In the remainder of this section, we focus on specific software artifacts and research papers attributed to this NASA-funded effort.

3.1 Research Contributions

Distributed laboratories are environments where scientists and engineers working in geographically separated locations share access to interactive visualization tools and large-scale simulation computations, share information generated by such instruments, and collaborate across time and space to evaluate and discuss their results. The intent of our research is to permit scientists, engineers, and managers at geographically distinct locations (including individuals telecommuting from home) to combine their expertise in solving shared problems by allowing them to simultaneously view, interact with, and steer sophisticated computation instruments executing on high performance distributed platforms. Research efforts being undertaken at Georgia Tech that address the topic of distributed laboratories are:
• Development of steering and monitoring tools and infrastructure used in the online observation and manipulation of scientific computations developed jointly with end users.

• Creation of middleware to transport the on-line monitoring and steering events, and exploration of its runtime adaptation to adjust event streams to current system loads and monitoring/steering needs.

• Understanding and constructing visualization support that permits the definition of appropriate visual abstractions and their efficient representation on 2D and 3D graphical displays.

• Creating abstractions and infrastructure that jointly enable multiple users to collaborate across distributed underlying machines and via the same or different computational tools.

The recent outcomes of the first two items, the steering and monitoring tools and infrastructure and the middleware, are summarized next. Visualization and collaboration work is described in Section 4.

Program Monitoring and Steering

Monitoring. Initially, our research focussed on the efficient, online monitoring of threads-based parallel programs, resulting in insights concerning the definition of alternative monitoring constructs embedded in application code, the construction of effective online monitoring support infrastructures co-resident with applications on shared memory parallel machines, and the creation of monitoring abstractions that give rise to flexible methods for online monitoring. Recent results include the evaluation of the performance effects of using alternative monitoring abstractions (attained in part with additional NASA student support and with support from the DOE via an extended student internship at Los Alamos Laboratories) [14]. Results also include the definition of quality of service constraints for online monitoring and the application of these constraints to distributed computational instruments operating in conjunction with the atmospheric model [9]. Next steps will include the creation of a real-time monitoring infrastructure so that specific monitoring rates may be guaranteed across distributed and parallel systems alike. This is particularly important when monitoring is combined with online program steering[14] or with online program adaptation[12].

Steering. We focus on the use of steering for program exploration rather than data exploration. Namely, we assume that scientists wish to use steering to explore different alternatives when running their large-scale simulations. Sample exploratory actions we have studied include alternative settings to selected simulation parameters, the use of alternative mathematical methods for computing certain simulation characteristics, and the online comparison of observational data with simulation output to drive the simulation and to detect model inaccuracies. Our approach to such steering actions is distinct from past work on program adaptation in its emphasis on user-driven steering actions. Namely, by providing users with rich graphical interfaces with which they may study simulation and observational data and the comparison of both, we can then enhance these same interfaces with graphical (and textual) constructs through which steering may be performed. Such 'steering by data manipulation' provides a natural way with which steering and data analysis/evaluation actions may be integrated. Our next steps in this work focus on the access to and use of very large data sets and complex computations, where a single graphical action by an end user may trigger significant changes in computations or may result in accesses to large amounts of data. Such user-triggered actions must imply changes outside the graphical environment itself or they will remain limited in scope and size to the single machine on which the user interface resides. The configurable middleware work described in the next section addresses this topic. First, however, we describe the monitoring and steering infrastructure(s) developed as part of our research.

The Falcon [4] Monitoring and Steering System is the basis on which all of our group's online program monitoring and steering is performed. Namely, Falcon provides the basic infrastructure for the online capture, buffering, and transport of events from a parallel or distributed application program and 'into' it from monitoring and steering servers associated with it. Prior to program execution, a program is instrumented
with application-specific constructs for monitoring or steering, typically by the end user. At runtime, Falcon's threaded servers associated with the program (and located in the program's address space) capture, buffer, and forward monitoring events and also inject appropriate steering events into the application. In earlier work, Falcon's utility was evaluated on shared memory platforms with performance monitoring tasks, using a parallel molecular dynamics application. In our current work, Falcon's server threads or processes monitor and steer the atmospheric modeling code developed in this project. It is also the basis on which runtime program adaptation for embedded applications is being explored [12].

Using Falcon as a basis, two different prototypes of steering toolkits have been developed and evaluated. Both prototypes share the same monitoring infrastructure, but differ in the steering and monitoring abstractions they implement. More importantly, they each explore different research goals.

The Progress Steering Toolkit [13] has been created to understand in detail the runtime overheads of steering and monitoring with alternative functionality and implementations ascribed to each and on shared memory computing platforms. When using Progress end users instrument their applications with library calls and then steer parallel applications with the Progress runtime system. Progress provides monitoring abstractions like sensors and probes and steering abstractions called actuators. Once created, the instances of such abstractions associated with the application being steered are known to and manipulated by Progress' two runtime components: (1) a steering server executing in the same address space as the target program and capable of inspecting and manipulating program state, and (2) a potentially remote client providing command and graphical interfaces. The application is steered via explicit user actions at the remote client, via algorithms executed at the client or server, or both. Also present in Progress is a 'steering language' with which sets of steering actions may be stated and even optimized with respect to the perturbation associated with monitoring and steering or with respect to the latencies of such actions.

The Eagle toolkit is exploring a more 'natural' model of steering than the earlier models offered by Falcon and Progress. Specifically, Eagle offers true object abstractions for monitoring and steering that may be created dynamically, in servers or clients, and associated with program instrumentation at runtime. The intent is to provide within the monitoring and steering infrastructure 'mirrors' of application-level objects so that it appears to developers that they are actually adjusting application objects when they are extracting state from or manipulating the target application. The technical contributions of Eagle include the formulation of its object model, the implementation of its event infrastructure implementing mirror objects and their efficient association with target applications, and the runtime optimization of events and event transport to deal with heterogeneous and dynamic target execution platforms.

Middleware for Interactive High Performance Applications.

Current middleware for high performance applications tends to focus on the efficient representation and execution of the application itself, not on the execution of this application in the context of additional tools it uses or of interfaces with which it is used. To address these shortcomings, we have developed the DataExchange library [2], a communication infrastructure supporting high performance interactive and collaborative applications. DataExchange addresses the interactive and dynamic nature of high performance applications by providing facilities for naming and locating data sources, for dynamically connecting to those data sources and for automatic and configurable redirection of data flow. Further, to support flexible data processing, analysis and reduction, DataExchange also offers an active-message-style message processing facility that serves to refine and extend DataExchange's data flow management functionality by making content-based message forwarding decisions. Recent enhancements to DataExchange include C++ and Java interfaces to the library, support for connectionless protocols (e.g., UDP), and implementation of a thread-safe model.

High performance interactive and collaborative applications are clearly heterogeneous and may be more strongly described as diverse. This means that, in addition to operating in an environment where machine-
level representations of elementary types may differ, the various components of the application may be
developed in a much less tightly coupled manner than a traditional parallel or distributed program. One
implication of this diversity is that it is not generally practical for all of the components operating in
the distributed laboratory to agree at compile-time on the formats of all data records to be exchanged
during execution. To meet this need and others, the DataExchange library is based on Portable Binary I/O
(PBIO) [1], a binary I/O package which provides a variety of mechanisms for handling data in a diverse
environment. In addition to features such as named data types and the ability to perform type conversions
between matching fields of different basic types, PBIO supports more elaborate and flexible typematching
than either PVM or MPI. Recently, PBIO has been extended with dynamic code generation for decoding
binary files at the receiver. Dynamic code generation is implemented using the Vcode package [3] developed
at MIT.

Our next steps in this research are building on the high performance 'interactivity infrastructure' pro-
vided by DataExchange to provide interfaces that are compliant with emerging industry standards and to
enable additional functionality not otherwise accessible to the high performance computing community. In
particular, layered above DataExchange, we are completing the implementation of a high performance ob-
ject substrate offering IDL interfaces so that CORBA-compliant objects as well as Java objects may interact
with this infrastructure. Technically interesting about our approach is our ability to associate attributes
with such IDL descriptions that permit the online configuration of objects. In addition, we are constructing
an event infrastructure that supports the efficient transport and analysis of monitoring and steering events
across distributed systems, including the specification of quality of service constraints for such events.

3.2 Software Artifacts

Falcon. Falcon is a system for application level on-line monitoring and steering large-scale parallel pro-
grams. In addition, it provides a graphical interface for visualizing thread-level information extracted through
the monitoring mechanism, and a reorder filter that imposes a partial order on a stream of thread-level events.

Progress. Progress [13] is a toolkit for developing steerable applications. The toolkit provides sensors,
probes, actuators, function hooks, complex actions, and synchronization points. Progress' server uses
Pthreads or a Mach-compatible Cthreads libraries in its runtime support. It also offers language constructs
for formulating and optimizing sets of steering actions.

Eagle. Eagle offers notions of mirror objects and event channels for remote association of monitoring
and steering abstractions with a target application. Event channels offer runtime configuration for high
performance, for heterogeneous and dynamically changing configurations of target applications.

DataExchange. The DataExchange library is a communication infrastructure supporting high perfor-
mance interactive and collaborative applications.

PBIO. PBIO is a binary I/O package that supports named data types, the ability to perform type con-
versions between matching fields of different basic types, and flexible type matching such that receivers are
not required to specify every field in the types to be exchanged.

3.3 Published Research

Greg Eisenhauer, Weiming Gu, Eileen Kraemer, Karsten Schwan and John Stasko, Online Displays of
Parallel Programs: Problems and Solutions, accepted for publication in Conference on Parallel and Dis-
tributed Processing Techniques and Applications (PDPTA '97), 1997.

Greg Eisenhauer, Beth Schroeder, and Karsten Schwan, DataExchange: High Performance Communications
Greg Eisenhauer, Beth Schroeder, Karsten Schwan, Vernard Martin, and Jeff Vetter, DataExchange: High Performance Communications in Distributed Laboratories, 9th Int'l Conference on Parallel and Distributed Computing and Systems, October, 1997.


Greg Eisenhauer, Weiming Gu, Eileen Kraemer, Karsten Schwan and John Stasko, Online Displays of Parallel Programs: Problems and Solutions, Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA ’97), 1997.

Greg Eisenhauer, Beth Schroeder and Karsten Schwan, From Interactive High Performance Programs to Distributed Laboratories: A Research Agenda, Proceedings SPDP’96 Workshop on Program Visualization and Instrumentation, October 1996.


4 Visualization

4.1 Research Contributions

Visual Steering and Analysis of Simulations

We have developed a heterogeneous environment for visual steering of computer simulations [16]. Visual steering implies two-way communication through 2D and 3D graphical data representations to bring about user involvement with the calculations as they occur. One must be able to insert on-the-fly parameter changes, even over 4D (3D + time) regions of the simulation and see results dynamically updated. The environment differs from previous steering systems in that it combines (i) a high level set of tools for instrumenting simulations; (ii) tools for efficient data exchange; and (iii) a visual steering interface for direct interaction with visualizations of the data. The instrumenting tools can be inserted by users into their codes without having to know about networking protocols or machine-dependent binary formats. The visual interface allows visualization of a variety of data from different applications since it employs general graphical objects that users can map to variables of their choice. The steering tools are integrated with the middleware that controls and monitors the distributed simulation and with the visualization/analysis tools so that one can apply steering commands simply by knowing the I/O formats of the instrumentation and without knowledge of the middleware structure.

Employing the instrumentation and I/O libraries, users can reconfigure their application codes to show new variables or accept different steering controls. This information is transmitted automatically to the visual steering environment. Since the steering tools appear in the data visualization space, users can steer based on the detailed 3D structure and time-dependent behavior of the data. The visualization/analysis tools can bring out correlations between variables or can focus on selected regions of the data; these analyses can
be used to refine the steering or more closely investigate the results of the steering input. Finally, the same visualizations can function both as data representations and as steering inputs. We demonstrate this with isosurfaces. These may define regions of physical interest but irregular geometry where spatial distributions of selected variables can be inserted.

Recent work includes support for time-dependent steering. Here time is treated on exactly the same basis as the spatial dimensions so there is a 4D environment, three shown spatially and one through animation. The steering interface is built upon a flexible visualization/analysis system. This permits the immediate display of time-dependent results from the dynamic simulations and refined interaction with the results to bring out the character and correlations of multivariate data. The user can then launch new simulations at any stage in this exploration using the visualization to define and focus the simulation parameters, region of interest, and time frame.

Collaboration via Visual Interfaces

We have developed a visual steering approach that supports collaboration between distributed users, allows asynchronous collaboration, supports project histories, and supports different roles [15]. Collaboration is achieved through sharing and direct manipulation of 3D interfaces for steering and analysis. With the collaboration tools a user may steer a simulation and share both the steering input and visual output with one or more geographically distributed colleagues.

Our approach, which employs Open Inventor for interaction and rendering, relies on the directed acyclic graph (DAG) abstraction employed by many modern user interface libraries including Open Inventor. Collaboration is built into the Open Inventor graphics library by utilizing the DAG abstraction. Since the data visualizations are contained in one or a few connected Inventor nodes, collaboration is achieved through maintaining consistency between shared nodes in a relevant portion of the DAG structure. This consistency is achieved through the introduction of a sharer node into the DAG where the sharer node is responsible for maintaining consistency between shared child nodes, including the nodes used for data visualizations.

We have developed three roles for use in the collaborative tools. In the first role, a “teaching” user offers to “share” nodes that simply enable other users to follow what the teacher is doing, perhaps at different levels of detail. In the second role, two or more users can be aware of each others’ actions by seeing where other users are actively exploring data or making changes but not being coupled to those explorations or changes. A third, more complex scenario is illustrated with the atmospheric model. Here users not only share views of each other’s actions but also pass control of steering objects between each other and see the results of their steering inputs.

4.2 Software Artifacts

GlyphMaker/Explorer based visualization interface. The original visual analysis tool used Glyphmaker [11] built on top of the SGI Explorer environment to help the user explore relations between spatially complex, time dependent data.

Open Inventor based visualization interface. Described in the preceding section, the Open Inventor based analysis tool allows asynchronous collaboration and 4D steering in addition to visually displaying complex, 3D atmospheric modeling data.

4.3 Published Research


Funding and Indirect Effects

This NASA-sponsored effort has been one of the most interesting tasks undertaken by our research group during the last five years. Specifically, due to its emphasis on cooperation between end users and computer scientists, the Computer Science team was able to use project insights to develop an entirely new research direction in the area of high performance computing. This direction concerns the interaction with high performance applications rather than their efficient execution, as addressed by most major national efforts (e.g., the metacomputing efforts, MPI, the parallel scalable I/O initiative, etc.). This enabled our team to generate additional funds and to undertake related research efforts that have already had significant additional impact.

Efforts leveraging this NASA grant have resulted in both an excellent equipment infrastructure now available to our research group and in significant additional research funding. It has also resulted in the exposure of many U.S. graduate and undergraduate students to the scientific applications and high performance computing technologies of interest to NASA (more than 50 PhD students, 20 MSc. students, and 20 undergraduates, including more than 10 women and minority students – detailed data available upon request). For brevity, we do not list these students individually. We also do not report the indirect results in terms of technical output as papers, software products, and technical reports.

New funding includes:


References


