DEVELOPMENT AND PERFORMANCE OF A MASSIVELY PARALLEL REGIONAL SPECTRAL MODEL

Yifeng Cui
San Diego Supercomputer Center
University of California, San Diego
9500 Gilman Drive, MC0505
La Jolla, CA 92093-0505, U.S.A
E-mail: yfcui@sdsc.edu

Hann-Ming Henry Juang
National Center for
Environment Prediction
World Weather Bureau,
5200 Auth Road
Camp Springs, MD 20746, U.S.A

Giridhar Chukkapalli
San Diego Supercomputer Center
University of California, San Diego
9500 Gilman Drive, MC0505
La Jolla, CA 92093-0505, U.S.A

Masao Kanamitsu
Scripps Institution of Oceanography
University of California, San Diego
9500 Gilman Drive, MC0224
La Jolla, 92093-0224, U.S.A

KEYWORDS
Regional Spectral Model; RSM; domain decomposition; parallelization; message passing.

ABSTRACT

The Regional Spectral Model (RSM) is a nested primitive equation spectral model used by U.S. operational centers and international research communities to perform weather forecasts and climate prediction on a regional scale. In this paper, we present the development of an efficient parallel RSM with a message passing paradigm. Our model employs robust and efficient 1-D and 2-D decomposition strategies and incorporates promising parallel algorithms to deal with complicated perturbation architecture and ensure portability using hybrid MPI and openMP. We also achieve bit reproducibility when our parallel RSM is compared to the sequential code. Performance tests were performed on an IBM SP, Compaq, NEC SX-6 and the Earth Simulator and our results show good scalability at over a thousand processors.

INTRODUCTION

The Regional Spectral Model (RSM) is a regional scale climate model developed by the National Oceanic and Atmospheric Administration's National Center for Environment Prediction (NCEP) for the study of regional scale climate and conduct seasonal climate predictions (Juang and Kanamitsu 1994; Juang et al. 1997). The model is nested within NCEP's global spectral model (GSM), and the global model provides the RSM with its required initial and boundary conditions. The RSM requires large compute resources to provide high-resolution scale climate predictions focused on limited areas. For the past decade, RSM has run on high-end workstations, on serial or shared memory parallel sytems, and scalar or vector processor systems. However, the need to run larger and longer scenarios at high resolutions with more sophisticated physics exceeds the capabilities of the original single-node computer architectures. Increased availability of distributed-memory parallel architectures provides the motivation for this work, initiated in 2002, to implement a message-passing version of the algorithm. The current parallel version, completed in beginning 2003, supports the message-passing interface for implementation on scalable, distributed-memory computers.

The RSM is a nested primitive equation spectral model on a stereographic projection and uses sine-cosine series as horizontal basis functions, consisting of a low-resolution global spectral model and a high-resolution regional spectral model. Spectral models have proven to be effective in achieving high-order accuracy when compared with grid-point models (Juang and Kanamitsu 1994). The sequential implementation of RSM uses nesting technique to interpolate coarse-grid global data to fine-grid local data. The perturbation technique is used to force the solution on the regional scale simulations to the global scale coarse-grid solution. The complex parallel implementation of the RSM is designed to be highly flexible, so that the code can be run on a range of architectures, including distributed shared memory as well as distributed vector processors.

Spectral transform methods have important computational advantages (Bourke 1972), but are in many respects the most difficult to parallelize efficiently because of their highly nonlocal communication patterns (Foster et al. 1995). In this paper, we provide a comprehensive description of the design of the parallel RSM, and an evaluation of its performance on IBM

Power3, Power4, NEC SX-6 as well as on Earth Simulator platforms. We particularly emphasize the flexible 1-D and 2-D data decomposition, and offer a strategy to maintain code structure while improving performance. Throughout our discussion the Regional Spectral Model will be referred to as RSM and the Global Spectral Model as GSM.

REGIONAL SPECTRAL MODEL

The RSM is introduced to predict deviations from the global model, and to improve the forecast of poorly represented global-scale waves in the regional domain (Juang and Kanamitsu 1994). The RSM's prognostic variables are expressed as perturbations from its global counterparts. The RSM uses the same primitive hydrostatic system of virtual temperature, humidity, surface pressure and mass continuity prognostic equations on terrain following sigma coordinates as global spectral model GSM does. The philosophy behind this design is to use the regional model to generate regional scale features forced by the large scale. Thus, the large scale in the regional domain needs to be preserved. Consistency between the two models insures that the change of the large-scale motion in the regional domain is minimized.

The numerical method is described as follows. The difference between the regional model and the global model fields are decomposed into sine and cosine functions (as opposed to spherical harmonic functions in the global model). The sine and cosine functions conveniently satisfy the zero and symmetric lateral boundary conditions appropriate for the differences. The space derivatives are computed as a sum of the derivatives computed from global spherical coefficients and from the sine and cosine functions of the regional model.

The RSM has a high-order accuracy of spectral computations, and a time-dependent perturbation method, which distinguishes it from other regional spectral models (Juang et al. 1997). The spectral transformation of the RSM is a two-dimensional cosine series for perturbation of pressure, divergence, temperature, and mixing ratio, and a two-dimensional sine series for perturbation of vorticity. In the horizontal, the regional model uses double sine-cosine series with wall boundary conditions as base functions, while the global model uses spherical harmonics as base functions. In the vertical, the regional model uses exactly the same finite-difference formulation as in the global model.

One-dimensional Fast Fourier Transform is used in x-direction while simple fourier summation is performed in y-direction, intentionally avoiding the use of the two-dimensional Fast Fourier Transform. This approach has the advantage of reducing the memory requirement and is the best fit for distributed machine adaptation. Also, this method is analogous to the global spherical transform, and thus the program structure of the two models is very similar.

The RSM program system is managed by Concurrent Versions System (CVS) and controlled by the configure and Makefile system. The modeling system is composed of three components: libraries, source code, and run scripts. The *library* contains model libraries, utilities, and climatological/constant fields, machine dependent and resolution independent sources. The library needs only to be made once. The *source code* is used to create executables, defines model resolution and options, creates model resolution dependent constants, and compiles source codes and creates run executables. The *run scripts* are for running the model and producing the outputs.

The model output files include two restart files: one contains fields on sigma surfaces, and the other contains fields on ground surface including diagnostic files such as surface fluxes and precipitation.

IMPLEMENTATION OF RSM ON MESSAGE PASSING

The parallel RSM maintains the implementation structure used in the GSM, but also makes use of promising parallel algorithms. This concept was designed by Juang in 2002. A future paper will discuss the concept in more details.

The parallel implementation builds on the existing shared-memory model. The basic strategy of the implementation of RSM is to make a minimum of changes to the 500,000 lines sequential code while a scalable and portable parallel version of RSM can obtain bit reproducible results. The coding design emphasizes the flexibility and readability. For flexibility, the parallel RSM is designed to run in hybrid mode, so that the code can be used for a range of architecture including inner loop vectorization, outer loop multi-threads (openMP) and multi-nodes (MPI). For readability and easy maintenance, the MPI implementation is designed to be as consistent as possible with the parent Global Spectral Model.

To achieve these goals, the single program multiple data (SPMD) programming paradigm is used, such that each processor performs all compositions for only one subdomain. The model data structure is shrunk in the north/south dimension, using only as much memory as needed on each processor. The Fortran 90 data structure is implemented to simplify the data sharing among all routines through few passing arguments. In order to manage the necessary code change, the C preprocessor, such as #define and #ifdef etc, are used to handle dependencies associated with shared- vs. distributed memory coding constructs without degrading the performance. For load balancing, local symmetric distribution of the grids is used and the computation in grids is partitioned so that each processor has approximately the same amount of work to do in each phase of the computation.

The perturbation method requires large memory and significant computation time to deal with base field over the entire regional domain. The parallel RSM employs a number of Fortran 90 features, such as dynamic memory and pointers, to achieve a more flexible and run-time configuration model suitable for distributed memory parallel computers. In particular, a dynamic memory management scheme is employed for all large model arrays to take full advantage of the large aggregate memory on distributed-memory platforms. An example of the coding design is:

```
#ifndef MP
    allocate (syn(igrd12_,levs_))
       … …
     deallocate (syn)
#else
     if(km.eq.1) then
       allocate (synpk1(igrd12p_,jgrd12_))
          … …
       deallocate (syn1)
     else if(km.eq.levs_) then
       allocate (synpk2(igrd12p_,levsp_,jgrd12_))
       … …
       deallocate (synpk2)
     endif
#endif
```

where the C preprocessor is set for sequential or parallel (one or multiple vertical levels) implementation. Individual parallel tasks are assigned a range of model latitudes for which they are responsible.

In parallel RSM, both 1-D and 2-D decomposition are built for flexibility, which enables runs on any number of processors. 1-D can be used on any number of processors, but only up to the number of the smallest dimension among all directions and all space. The 2-D decomposition data transposition strategy utilizes a 2-D model data structures, meaning a single dimension of data structure in memory for each processor. This method has been applied successfully in several parallel atmospheric models (Foster and Worley 1997; Barros et al. 1995; Skalin and Bjorge 1997), including the Global Spectral Model. 2-D can be used up to the number of product of two smallest dimensions in all directions and all spaces, except with any prime number of processors (Juang and Kanamitsu 2001). In the current design, the total numbers of working processors NODES equals Nrow times Ncol where Nrow and Ncol are any integers in X and Y directions. If NODES are prime integer, we set Ncol=1 and Nrow=NODES. In this case, 2-D decomposition becomes 1-D.

For a spectral model that uses transform method as in RSM, dependent variables and their tendencies take the form of spectral coefficients and grid point values at different stages of the computations. Each stage is characterized by full dimension in one of the three-dimensions. For example, when the Fourier transform is performed in an east-west direction, the full array in the east-west direction needs to be available on each processor, but the size in the north-south and vertical directions can be arbitrary. On the other hand, when the Legendre transform is performed, each processor needs the full array in north-south direction, but the array in east-west and vertical directions can be arbitrary. Thus, the parallelization of the spectral model requires rearrangement of arrays from one configuration to the other (for example, full array in x to full array in y). This rearrangement, also called the transpose method, requires communication between processors and is programmed using MPI routines. The transpose method is used to minimize the code modification - the original serial is left unchanged, the transpose routines are inserted between the transforms in east-west direction and in the north-south direction.

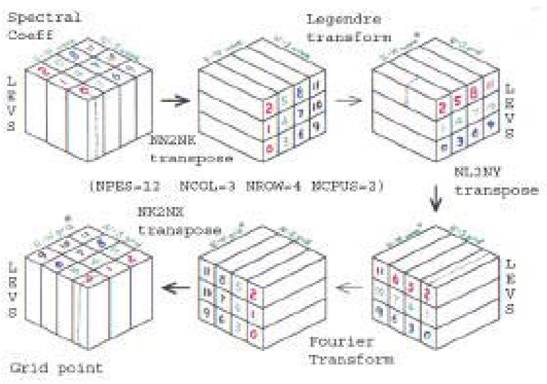

Figure 1: Parallel RSM Data Transposition Strategy, concepted by H.-M. H. Juang.

The entire transposition process is illustrated in Fig. 1. The general sequences of the forecast model after data are read in and transformed to spectral space include: transform from spectral to grid-point space, compute all non-linear model dynamics and model physics, transform back from grid-point to spectral space, and perform linear computation such as semi-implicit and time filer in spectral space. It requires only one transpose from the Legendre transform array configuration to the Fourier transform configuration for 1-D decomposition. All processors communicate with each other. In contrast to this, 2D decomposition requires three transposes in forward direction, namely in advance of Legendre transform that has a north-south dependency, in advance of Fourier transform that has east-west dependency and in advance of nonlinear computation that has vertical dependency. In this case, communications only take place within a subgroup of processors. After the dynamics and physical computation with vertical dependency are complete, the data is transformed back with three more transposes in the reverse way.

The parallel code includes both the data model and the message passing model. The input/output (I/O) is the part of the code where the SPMD paradigm is not employed. Also, the global to regional spherical transform is not parallelized in the current effort. This is primarily because of very limited execution time compared with the regional forecast computation, as was indicated in the profiler performance analysis. The global coding is generally executed independently on each node.

Bit reproducibility was a goal of this parallel effort. This implies that on the testing computer, the parallel code be run on a variable number of processors and the sequential code should give identical results. To debug the massage-passing version of RSM, we used a parallel debugging tool called Totalview, developed at Etnus, to compare serial and parallel version outputand ensure bit-by-bit reproducibility. This produces a high level of confidence at each coding step. The final version of the parallel RSM produces results identical to the original sequential code on the same machine, independent of the number of processors on which it ran.

PARALLEL PERFORMANCE

The parallel RSM is portable, and only minimal modifications are needed to run the code on different platforms. The code has already been run on an IBM SP, Compaq Alpha, NEC SX-6 and the Earth Simulator. The performance numbers listed below are collected from an IBM SP and SX-6. The testing horizontal resolutions are at T62L28r12885 and at T62L28r512335 (global model resolution of triangular truncation 62 with 28 levels). T62 has sufficient resolution to give good large-scale predictions. Our test IBM SP platforms were the Power3 Blue Horizon and Power4 DataStar, both located at San Diego Supercomputer Center. The Power4 DataStar is a 7.9 teraflops IBM SP system which consists of 176 8-way 1.5Ghz p655 nodes and 7 32-way 1.8 Ghz p690 nodes, with total shared memory of 3.2 terabytes. The Power3 Blue Horizon is a 1.7 teraflop system with 144 8way SMP nodes. Each SMP node has 4 gigabyte of memory shared among its eight 375 Mhz Power3 processors. The vector machine NEC-6 is a single cabinet, 8 cpu node with 64 GB of symmetric shared memory. Each cpu is a single-chip 8-way vector processor. The vector units operate at 500 Mhz. Peak performance is 8 GFLOPS/processor with 8 pipes given. The 8 vector register per CPU are 256 elements long and CPU-memory bandwidth is 32GB/sec/CPU (Baring 2003).

We performed our experiments in two stages. In the first stage, single node tests were performed for code optimization. In the second stage, we tested and analyzed scalability on equivalent platforms.

Optimization

On IBM SPs, we use xprofiler and the IBM SP hardware counter utilities HPM (Hardware Performance Monitor). to analyze the single-node performance and final options are tuned as:

FORT_FLAGS="-O3 -qfixed -qrealsize=8 -qnosave -qmaxmem=-1 -bmaxdata:0x80000000 -bmaxstack:0x10000000"

LOAD_FLAGS="-O3 -qfixed -qrealsize=8 -qnosave -qmaxmem=-1 -bmaxdata:0x80000000 -bmaxstack:0x10000000"

The total performance of the IBM SP3/SP4 is about 109/531 Mflop/s on a single processor at 128x85x28 resolution, resulting in a 7%/8% of peak performance.

On ARSC NEC-6, we set up some environments such as I/O variable F_FILEINF, profiling variable F_FTRACE and performance data variable F_PROGINF to analyze the code performance. Some exploration of the compiler options was made to achieve optimal performance. The compiler options are tuned as:

FORT_FLAGS="-Chopt -ftrace -f3 -float0 -ew -llapack_64 -lblas_64 -lfft_64 -I./MODS"

LOAD_FLAGS="-Chopt -ftrace -f3 -float0 -ew -llapack_64 -lblas_64 -lfft_64 -I./MODS"

For a few subroutines, they abort under the highest optimization (–Chopt) but not under a lower optimization (–Cvopt). Some modest modifications such as inlining are sufficient to achieve 35% of peak performance on average (2806Gflops/p), and 80% of peak performance in the best case for Legendre transform on the resolution at 720x715x28. The vector ratio achieves over 96%.

Scalability

The wall clock time of the parallel code runs on different number of processors are shown in Figure 2 for IBM SP3/SP4. Here, a time step of 30 seconds is used and the simulation period is 6h. The solid lines represent the measurement on Power3, the dashed lines on Power4; the hollow points reflect the 128x85x28 resolution, the solid points the 512x335x28 resolution. The figure shows that the low-resolution model performance saturates quickly as the number of processors increase. For the high-resolution model, it scales to 1024 processors and the model efficiency does not diminish quickly with an increase in the number of processors. The smoothness of the curve is affected at certain points by a mismatch of the domain decomposition with the number of mesh points (Wehner et al. 1995), as well as a transition from using small number of nodes to larger number of nodes. The speedup and efficiency on various processor configurations are summarized in Figure 3. The fine-resolution runs (512x335) take significant longer time on the dynamics and physics computation. At 1024 processors, the efficiency is reduced to 39%. There are several factors contributing to the falloff in performance. In addition to the remaining serial codes for the treatment of lateral RSM boundaries, we use the 'master-worker' strategy - the master processor distributes the work to the available processors and is responsible for reading and writing the history and checkpoint data on an hourly-averaged basis. As a result, I/O may account for 20-30% of the total elapsed time. Also, with the increased number of processors, the message communication cost is increased, and the load imbalance becomes noticeable, which will be discussed later.

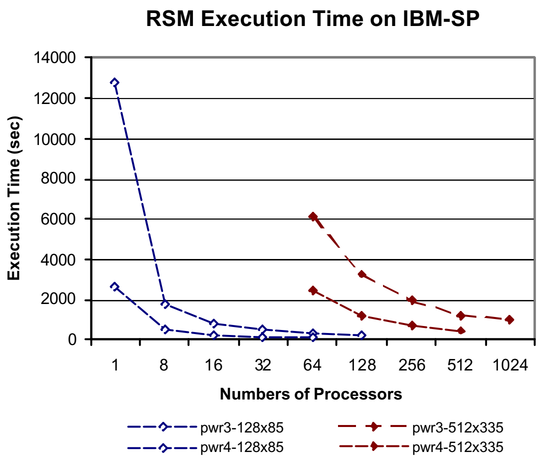

Figure 2: RSM Execution Time on IBM SPs

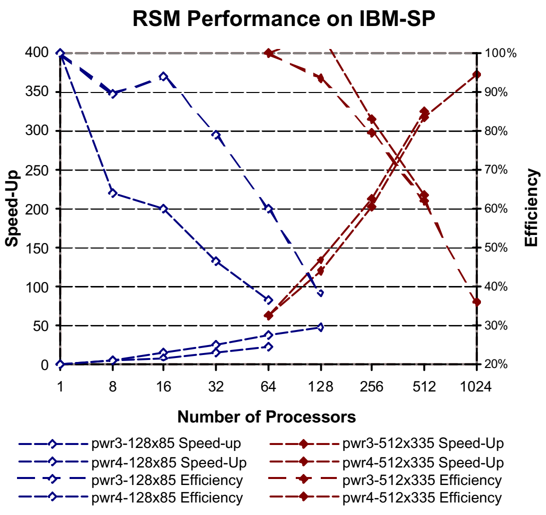

Figure 3: RSM Speedup and Efficiency on IBM SPs

The performance measured with 64-processors at 512x335x28 resolution is 137/357 Mflop/s on IBM SP3/SP4. Performance gradually degrades from this level to 60 Mflop/s on 1024 processors, which is fairly good when compared with other applications. The per-processor performance achieved on microprocessor-based parallel computers is often disappointing: a small fraction of peak. The reasons for this are not yet fully understood, but seem to be a combination of code structure (data structures and loop structures designed for vector machines often do not perform well on RISC microprocessors) and high memory bandwidth requirements (Drake and Foster 1995). Another reason for this is the parallel partitioning strategy designed in climate modeling. The spectral transform method imposes the most severe constraint on selecting a parallel decomposition strategy (Hack et al. 1995). The measurements above are based on system-independent libraries including Fast Fourier Transform (FFT) for portability consideration. We recently ported the code to use vendor-supplied subroutine libraries ESSL FFT, which increased performance by about 15%.

Communication Overhead

The parallel model has inter-processor communication overhead not present in the sequential model. The communication overhead in the parallel version of the RSM at 512x335x28 was approximately 34% on the 64 processor run, but increased to 52% on the 128 processor run. As mentioned above, the spectral model has the highly nonlocal communication patterns. The results indicate that the communication cost is the most responsible for sublinear performance scaling of the code on IBM SP system.

Load Imbalance

With an increasing number of processors the load imbalance becomes noticeable. The load imbalance accounted for about 12% of the total cost on the 16-

processor runHere load imbalance is calculated as follows:

$$L = \frac{T_{\max} - T_{avg}}{T_{\max}}$$

where Tmax and Tavg are the maximum and average execution times of MPI tests respectively. The load balance of 1.0 means all processors take exactly the same amount of time. Unlike the global spectral model that uses triangular truncation, the regional spectral model doesn't have a markedly uneven subdomain distribution in the 2-D decomposition. The load imbalance is mainly due to the different tasks of the processors such as the calculation in the physical parameterization (e.g. shortwave radiative transfer) or I/O.

Machine Comparisons

Our tests show that the one-dimensional decomposition is more efficient than the two-decomposition on SX-6 vector processors due to increased vector lengths. This is consistent with previous work on the Fujitsu VPP5000 (Juang and Kanamitsu 1997). However, this does not apply for cache-based IBM-SP. On the IBM SP, the 2-D decomposition is significantly better than 1-D decomposition. The tests on the Power3 show that on a small number of processors (8 to 16), the run based on 2-D decomposition saves approximately 20% of the execution time; on a large number of processors (such as 128), it saves approximately 40% of execution time.

Table 1: Performance Comparison on SPs and SX-6

	Power3	Power4	SX-6
1 cpu			
Execution time (sec)	902	193	94
Mflips/Mflops	132	593	1002
Sustained Performance*	8.8%	8.7%	12.5%
4 cpus			
Execution time (sec)	252	84	57
Mflips/Mflops	136	345	708
Sustained Performance*	9.1%	5.1%	8.9%
Efficiency	90%	57%	41%

* Percent of theoretical Peak Performance

A rough comparison of the performance on different machines is summarized in Table 1. A more detailed comparison, including theEarth Simulator, will be addressed later in another paper. Here, we test on the small resolution at 128x85x28 and set a time step of 360 seconds. The Power4 is more than 4 times faster than the Power3, but it sustains a much lower fraction of peak due to its relatively poor ratio of memory bandwidth to peak performance. Results demonstrate that the SX-6 achieves high sustained performance and significantly outperforms the superscalar designs of the Power3 and Power4. In addition to its vector architecture and the shared-memory structure, the SX-6 has high memory bandwidth and low memory latency. The SX-6 vector unit lacks data cache. Instead of relying on data locality to reduce memory overhead, memory latencies are masked by overlapping pipelined vector operations with memory fetches (Oliker et al. 2003). Limited scaling on the SX-6 is mainly due to the reduced length of big vector loops as the number of processors increases. More work is being done to improve its scalability and efficiency on vector architecture.

CONCLUDING REMARKS

The production parallel Regional Spectral Model is designed to execute on massively parallel computer systems. The model employs a flexible 1-D and 2-D algorithmic decomposition, which has been implemented for both shared-memory and distributed-memory parallel architectures. With the parallel version of RSM, overall computational performance is quite respectable for distributed-memory implementation on IBM SP machine over number of nodes from a few to 1024 processors. Since the completion of the parallel implementation, the parallel RSM has been widely adopted by the international research community. In particular, the National Center for Environment Prediction (NCEP) has incorporated the parallel implementation into their operational RSM version for daily operational weather forecast and climate prediction.

Although the parallel RSM code scales up to 1024 processors with fine-resolution, the parallel speedup with RSM begins to show a significant degradation as the number of processors employed begins to approach the maximum number of available parallel processes, as seen with other climate models. The communication overhead and load imbalance are among the reasons of this degradation. A significant weakness in the current version of the parallel RSM is the treatment of I/O. Further implementation of parallel I/O techniques is needed to achieve sufficient I/O bandwidth to the parallel file systems. In addition, the model initialization step involving global to regional spectral transform has been done sequentially due to the small fraction of time used by this step.

During the course of this work, the RSM was ported to the Earth Simulator (ES) to prepare for a 50-year downscaling of atmospheric reanalysis over the continental US. While RSM performance is still being optimized on the Earth Simulator, we have already observed 1.6 Gflops preprocessor performance at the preliminary test of 10-km resolution at 1024x651x28 on ES, which is 20% of the peak performance. The ES is a NEC supercomputer that implements high-speed vector processors in full scale and consists of 640 nodes connected by a 640x640 single-stage crossbar switch. The experience on ES will be very important in enabling meaningful comparisons between the capabilities of the Earth Simulator and IBM SP, and can guide future system upgrades to support RSM and similar applications including distributed simulations.

ACKNOWLEDGEMENTS

This work was supported by the National Partnership for Advanced Computational Infrastructure (NPACI) at the San Diego Supercomputer Center, NOAA Climate Dynamics and Experimental Prediction program, California State, and Climate Modeling Center at National Center for Environmental Prediction. We used the IBM SP3 Blue Horizon and SP4 Datastar systems at the San Diego Supercomputer Center, the NEC SX-6 system at the Alaska Arctic Region Supercomputing Center, and the NEC Earth Simulator of Japan.

REFERENCES

Baring, T.L.. 2003. "SX-6 Compare and Contrast". *Techn. Report.* Arctive Regional Supercomputer Center, University of Alaska Fairbanks (May).

Barros, S.R.M.; D. Dent; L Isaksen; G. Robinson; G. Mozdynski, and F. Wollenweber. 1995. "The IFS model: A parallel production weather code". *Parallel Computing,* No.21 (Oct), 1621-1638.

Bourke, W.. 1972. " An efficient, one-level, primitive-quation spectral model". *Monthly Weather Review*, No.102, 687-701.

Drake, J. and I. Foster. 1995. "Introduction to the special issue on parallel computing in climate and weather modeling". *Parallel Computing,* No.21 (Oct), 1539-1544.

Foster, I.; B. Toonen and P.H. Worley. 1995. "Performance of Parallel Computers for Spectral Atmospheric Models". Tech. Report ORNL/TM-12986, Oak Ridge National Laboratory. Oak Ridge, Tenn., (Apr).

Foster, I. and P.H. Worley. 1997. "Parallel algorithms for the spectral transform method". *SIAM J. Sci. Comput,* No.18(2), 806-837.

Hack, J.J.; J.M. Rosinski; D.L. Williamson; B.A. Boville and J.E. Truesdale. 1995. "Computational design of the NCAR community climate model". *Parallel Computing*, No.21 (Oct), 1545-1569.

Juang, H.-M. H. and M. Kanamitsu. 1994. "The NMC Nested Regional Spectral Model". *Monthly Weather Review*, No.122, 3-26.

Juang, H.-M. H.; S.-Y. Hong and M. Kanamitsu. 1997. "The NCEP Regional Spectral Model: An Update". *Bulletin of the American Meteorology Society,* No.78, 2125-2143.

Juang, H.-M. and M. Kanamitsu. 2001. "The Computational Performance of the NCEP Seaonal Forecast Model on Fujitsu VPP5000". *Ninth ECMWF Workshop on the use of high performance computing in meteorology* (Nov 13-17). World Scientific under the title: Developments in teracomputing. Ed. by W . Zwieflhofer and N Kreitz. Singapore, ISBN 981-02-4761-3.

Oliker L.; A. Canning; J. Carter; J. Shalf and D. Skinner. "Evaluation of Cache-based Superscalar and Cacheless Vector Architectures for Scientific Computations". *Supercomputing 2003* (Phoenix, Arizona, Nov 15-21).

Skalin, R. and D. Bjorge. 1997. "Implementation and performance of a parallel version of the HIRLAM limited area atmospheric model". *Parallel Computing,* No.23 (Dec), 2161-2172.

Wehner, M.F.; A.A. Mirin; P.G. Eltgroth; W.P.Dannevik; C.R.Mechoso; J.D. Farrara and J.A. Spahr. 1995. "Performance of a distributed memory finite difference atmospheric general circulation model". *Parallel Computing*, No.21 (Oct), 1655-1675.

AUTHOR BIOGRAPHIES

CUI, YIFENG received his Ph.D. in Hydrology from University of Freiburg. He also holds a bachelor degree in Meteorology from Nanjing Institute of Meteorology, and a master degree in Hydroclimatology from Hohai University in 1987, China. He worked for a couple of years for Environment Canada before moving in 2000 to the University of California, San Diego, where he now works on parallel performance with the Scientific Computing Department at the San Diego Supercomputer Center.. His e-mail address is: `yfcui@sdsc.edu.`

HANN-MING HENRY JUANG is a research meteorologist at the National Center for Climate Prediction, NOAA. His e-mail address is: `henry.juang@noaa.gov.`

GIRIDHAR CHUKKAPALLI received his Ph.D. in Mechanical Engineering from University of Toronto, Canada. He has an extensive background in CFD algorithms (finite element/ finite difference, spectral schemes). He is currently a senior computational scientist with the Scientific Computing Department at the San Diego Supercomputer Center, and has extensive experience with both shared memory and message passing programming. His e-mail address is: `giri@sdsc.edu.`

MASAO KANAMITSU is a research meteorologist at Scripps Institution of Oceanography, University of California, San Diego. His e-mail address is `mkanamitsu@ucsd.edu.`