Performance Tuning and Evaluation of a Parallel Community Climate Model *

John B. Drake† Steve Hammond‡ Rodney James§ Patrick H. Worley¶

Abstract

The Parallel Community Climate Model (PCCM) is a message-passing parallelization of version 2.1 of the Community Climate Model (CCM) developed by researchers at Argonne and Oak Ridge National Laboratories and at the National Center for Atmospheric Research in the early to mid 1990s. In preparation for use in the Department of Energy's Parallel Climate Model (PCM), PCCM has recently been updated with new physics routines from version 3.2 of the CCM, improvements to the parallel implementation, and ports to the SGI/Cray Research T3E and Origin 2000. We describe our experience in porting and tuning PCCM on these new platforms, evaluating the performance of different parallel algorithm options and comparing performance between the T3E and Origin 2000.

1 Introduction

The Community Climate Model (CCM) is an atmospheric general circulation model developed at the National Center for Atmospheric Research (NCAR). Versions of it are used as the atmospheric component of both the Department of Energy's Parallel Climate Model (PCM) and NCAR's Climate System Model (CSM). These are both coupled climate models composed of ocean, atmosphere, land and sea ice component models. The current distributions of CCM, versions 3.2 and higher, can be run on high end workstations, on serial or shared memory parallel vector processor systems, and on parallel systems supporting the MPI message-passing interface. The MPI parallel implementation is based on a one-dimensional decomposition of the computational domain, and is limited to 64 processes for the current target problem resolution of T42L18, which corresponds to a 128×64 computational grid covering the surface of the sphere and 18 vertical levels.

PCCM2.1 is a parallel implementation of version 2.1 of the CCM developed in the early to mid 1990s for the Department of Energy CHAMMP [3] program by a collaboration of researchers from Argonne and Oak Ridge National Laboratories and the National Center for Atmospheric Research. It uses a two-dimensional domain decomposition approach that allows up to 1024 processes to be used for T42L18. PCCM2.1 was originally targeted for the Intel Paragon with 1024 processors and the IBM SP2 with 128 processors. But the code was written to be easily ported to other multiprocessors that support message-passing paradigms, or to run on machines distributed across a network with PVM [7].

PCCM2.1 was not adopted as the production CCM by NCAR because CCM had reached version 3.0 by the time PCCM2.1 was validated. There were also perceived difficulties in maintaining efficiency on parallel vector processors when incorporating the code modifications for the two-dimensional decomposition. However, the computational

* The research performed by J. Drake and P. Worley was supported by the Atmospheric and Climate Research Division of the Office of Science, U.S. Department of Energy under Contract No. DE-AC05-96OR22464. The research performed by S. Hammond and R. James was sponsored in part by the Department of Energy Climate Change and Prediction Program and the National Science Foundation. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

† Oak Ridge National Laboratory, P.O. Box 2008, Bldg. 6012, Oak Ridge, TN 37831-6367 (drakejb@ornl.gov)

‡ National Center for Atmospheric Research, P.O. Box 3000,Boulder, CO 80307-3000 (hammond@ncar.ucar.edu)

§ National Center for Atmospheric Research, P.O. Box 3000,Boulder, CO 80307-3000 (rodney@ucar.edu)

¶ Oak Ridge National Laboratory, P.O. Box 2008, Bldg. 6012, Oak Ridge, TN 37831-6367 (worleyph@ornl.gov)

requirements of the new generation of climate models, as constrained by the parallel platforms currently available to U.S. climatologists, may require the use of large numbers of processors in order to achieve the goals of the U.S. Global Change program [10]. To assess the potential for performance gains from exploiting additional parallelism in the atmospheric model, PCCM has been updated to use the same physics routines and numerical methods as the current version of CCM. As the current version of CCM is also a parallel code, PCCM is no longer a useful name for the two-dimensional parallelization. We will henceforth refer to the new version of PCCM as CCM/MP-2D.

In this paper, we examine the advantages of using a two-dimensional decomposition over a one-dimensional decomposition with CCM/MP-2D on the T3E and the Origin 2000. We also compare the T3E and Origin as platforms for running the code. Fairness is an important issue in these types of performance studies, and we carefully tuned CCM/MP-2D on the T3E and the Origin 2000 before making comparisons. As part of our presentation, we discuss how sensitive performance is to the choice of parallel algorithm. We also look at scalability and instrumentation data to identify where and why performance is lost for each problem size and on each platform.

The next section describes the numerical and parallel algorithms used by CCM/MP-2D. Section 3 describes the target platforms. Section 4 describes the methodology used to port and tune the codes. Section 5 describes CCM/MP-2D performance in detail. Section 7 contains our conclusions.

2 CCM/MP-2D 3.2

CCM3 is the latest version of the community climate model that has been developed at NCAR and provided to atmospheric scientists for over a decade [2, 8, 9]. The vertical and temporal aspects of the model are represented by finite difference approximations. The spherical harmonic transform (spectral transform) method is employed to compute the dry dynamics [8, 11]. This method computes the spherical harmonic function coefficient representation of the atmospheric state variables by first transforming them from the physical domain (longitude-latitude-vertical) to the Fourier domain (wavenumber-latitude-vertical) using fast Fourier transforms (FFTs) in the longitude direction. Then the state variables are transformed from the Fourier to the spectral domains (spectral coefficients - vertical) using Legendre transforms (LT) in the latitude direction. The set of spectral coefficients is typically truncated in some fashion to avoid aliasing. Horizontal derivatives and linear terms involving these variables are calculated and combined to form the dynamical right hand sides in spectral space. These spectral space computations are independent between spectral coordinates. The results are transformed back into the physical domain where they are used to update the model variables.

The calculation of nonlinear terms in the equations of motion are carried out on a grid in the physical domain. A tensor product grid is used, where the latitude coordinates are chosen to be Gauss quadrature points, as used to approximate the Lengendre transform, and the longitude points are equispaced. The "physics" computations involve only the vertical column above each longitude-latitude grid point and are thus numerically independent of each other in the horizontal direction. Trace gases, including water vapor, are transported by the wind fields using a shape perserving semi-Lagrangian scheme [12] on the physical grid.

For spectral global circulation models such as CCM3 it is canonical to denote the resolution by the truncation wave number and the number of vertical levels in the model discretization. For example, a spectral model that uses a 128 longitude by 64 latitude grid and 18 vertical levels and a triangular truncation of the spherical harmonic coefficients is called a T42L18 model. The "T" indicates triangular truncation, 42 indicates the maximum Fourier wave number, and the "L" denotes the number of vertical levels. Resolution T170L18 has 512 longitude by 256 latitude grid and 18 levels.

In the two-dimensional domain decompostion used by CCM/MP-2D, the longitude and latitude dimensions are decomposed, and the resulting blocks combined to define a decomposition into longitude-latitude patches, leaving the vertical dimension undecomposed. Two patches are assigned to each processor, one from the northern hemisphere and its reflection across the equator in the southern hemisphere. This allows symmetry to be exploited in the Legendre transform. This assignment naturally defines a virtual two-dimensional processor grid, with rows representing common latitude assignments and columns representing common longitude assignments.

Given this decomposition, the physics computations are independent between processors, and no interprocessor communication is required. However, much of the physics is related to solar radiation and there is a significant load imbalance between night and day grid points. To alleviate this, each processor swaps half of its grid points with the processor in the same row holding grid points that are 180 degrees away, swapping them back when the physics computations are complete.

The semi-Lagrangian algorithm also uses the physical grid. For each grid point, a trajectory is calculated back in time, to determine what grid cell to use to interpolate the current values. This calculation is independent between grid points, but the data needed to calculate the trajectories and to interpolate the fields may not be local to the processor holding the grid point. The current parallel algorithm fills halo regions of sufficient thickness around each patch that, once filled, all needed information is local to each processor. Typically, this only requires communication with nearest neighbors in the logical processor grid. However, near the poles the halo region for a patch must include the entire polar cap. This requires communication between all processors assigned patches near the pole, resulting in a load imbalance in the cost of filling the halo regions between the polar and equatorial processors.

Two different approaches are supported in CCM/MP-2D for computing the FFTs used in the spectral transform method: distributed and transpose. The distributed algorithm computes the FFT using the given domain decomposition, communicating between processors in the same row to share data and intermediate results. The transpose algorithm "rotates" the domain decomposition within a processor row, undecomposing the longitude coordinate, and decomposing over the vertical levels and the different fields. Using this scheme, each processor has a set of independent FFTs to calculate. When the transforms are complete, the rotation is reversed, undecomposing the vertical levels and the fields, and decomposing over the wavenumber coordinate.

The Legendre transform used in the spectral transform is approximated by Gauss quadrature for each spectral coefficient. Each processor computes its contributions to these integrals, and a collective summation of the contributions over each column of processors is used to complete the computation. The parallel summation algorithm used in the Legendre transform replicates the spectral coefficients assigned to a given column of processors over all processors in the column. This redundancy results in duplicate work in spectral space, but allows the inverse Legendre transform to be computed without further interprocessor communication. Given the relatively small amount of time spent in spectral space computation, this is often a cost-effective tradeoff.

For more details on the parallel algorithms used in CCM/MP-2D see [4, 6]. Parallel algorithm improvements introduced in CCM/MP-2D not described in these references include support for vendor-supplied FFT routines and MPI collective communication rotuines for the transpose and collective sum operations.

3 Platforms

CCM/MP-2D 3.2 has been ported to a number of different platforms, using a number of different message-passing libraries. In this study we focus on performance using the T3E-900 at the National Energy Research Scientific Computing Center (NERSC) and a SGI Origin 2000 at Los Alamos National Laboratory (LANL), each using the vendor-supplied MPI communication libraries. At the time of these experiments, these two systems were the most likely candidates for use in production runs of the coupled climate models by the Department of Energy.

The SGI Origin 2000, henceforth referred to as the "Origin", is a distributed shared memory (DSM) parallel system made up of "nodes" consisting of two processors that share a common memory. Nodes are interconnected via a high-performance, highly-connected but nonuniform access network. Thus, although all memory is globally accessible, access time varies with the network distance between the memory and accessing processor. The Origin supports traditional shared memory programming models, but current experience indicates that the "programming discipline" natural to message-passing is important for performance, and message-passing is a reasonable approach to using the machine. For these experiments we used version 6.5 of the IRIX operating system, MPI from version 1.3.0.0 of the SGI Message Passing Toolkit, and the MipsPro 7.20 Fortran compiler with compilier options -O2 -64 -r10000. We also used the FFT routines from the SCSL math library. The particular Origin used in these experiments is one of the machines in the Advanced Computing Laboratory at LANL that is dedicated to climate research. It has 128 250-MHz MIPS R10000 processors. CCM/MP-2D performance data were collected in March and April of 1999.

The T3E-900, henceforth referred to as the "T3E", is a second-generation distributed memory parallel system designed by Cray Research. Each node consists of a single processor/memory pair interconnected via a high-performance, three-dimensional bidirectional torus network. Hardware support exists for accessing remote memory directly, but experience has shown that message-passing is still the best programming paradigm to use if high performance is required. For these experiments we used version 2.0.4.46 of the UNICOS/mk operating system, MPI from version 1.3.0.0 of the Message Passing Toolkit, and the Cray CF90 version 3.2.0.0 Fortran compiler with compilier option -Oscalar3. We also used the FFT routines from the LIBSCI math library. The NERSC T3E has 644 450-MHZ DEC Alpha EV5 RISC processors. CCM/MP-2D performance data were collected in February and March of 1999.

To provide additional context to the Origin and T3E data, some measurements are also presented for the following platforms:

- IBM SP3 at Oak Ridge National Laboratory (ORNL). This system has 62 POWER3 2-way SMP nodes (200

MHz POWER3 with 4MB L2 cache, equivalent to an RS/6000 Model 260). Data was collected in May and June of 1999 using the ESSL math library and -O3 optimization.

- SGI Origin 2000 using 195 MHz processors. This is the same Origin system at LANL described above prior to being upgraded to 250 MHz processors. Data was collected in May, 1998 using the SCSL math library and -O2 optimization.
- HP/Convex SPP-2000 at the National Center for Supercomputer Applications. This system has 64 180-MHz HP PA-RISC 8000 processors, organzied as 4 16-processor Hypernodes. Data was collected in May 1998 using the VECLIB math library and +02 optimization.
- 266 MHz Pentium II workstation at ORNL. Data was collected in May 1998 using the Portland group compilers and -O3 optimization.

.

4 Methodology for Porting and Tuning

CCM/MP-2D has numerous options that can be set to tune performance on a parallel platform. At the most primitive level is the choice of communication protocol, for example which of the many MPI point-to-point communication routines to use, and whether to try to overlap communication with computation or hide latency. Different choices may be appropriate for different phases of the computation, for example the best protocol for the physics load balancing algorithm may be different from the best protocol for the parallel semi-Lagrangian algorithm. At the next level is the choice of parallel algorithm to use to implement the transpose, equivalent to the MPI_ALLTOALLV command, and the collective summation, equivalent to the MPI_ALLREDUCE command. These MPI collective communication commands are also supported options, but they are not always the best choice. At a higher level is the choice of distributed versus transpose parallel FFT algorithm.

At the highest level is the aspect ratio of the logical processor grid and the mapping of the logical processor grid to the real machine. For example, 64 processors can be configured as a 64x1, 32x2, 16x4, etc. logical processor grid, where the first number denotes the number of processors assigned to compute the parallel FFT, and decompose the longitude direction, while the second denotes the number of processors assigned to compute the parallel Legendre transform. Different choices also imply different shaped domain decomposition patches, which will affect the efficiency of the parallel semi-Lagrangian algorithm.

CCM/MP-2D is a large code with a relatively expensive initialization phase, requiring the input of large static datasets. In a production run, the initialization cost is unimportant, as the code will run for days or weeks. However, in a short tuning evaluation run, the initialization phase dominates the runtime, limiting the number of tests that can be made with the full code. To aid in tuning the performance of CCM/MP-2D, we used two kernel codes: COMMTEST and PSTSWM.

COMMTEST tests the performance of exchanging data between two or more processors. For these experiments we looked at the "peak achievable" rate for swapping data between two processors, using both unidirectional and bidirectional protocols. The distinguishing feature of this test code is that it uses the same communication primitive wrappers used in CCM/MP-2D, so that the results are relevant to what would be seen in the production code. All available message-passing protocols for exchanging data between two processes were examined, for a large range of message sizes.

PSTSWM is a parallel spectral transform shallow water model [5, 14] that is an accurate representation of the parallel algorithms used for the dry dynamics in CCM/MP-2D. In particular, the parallel algorithms were designed and evaluated in PSTSWM first, then ported to CCM/MP-2D. A series of test suites have been developed for PSTSWM that look at all possible communication protocols for each of the parallel algorithms used in the spectral transform method. From this data, we identified a small number of parallel algorithms and implementations to examine in the context of CCM/MP-2D. We do not currently have a kernel code for the parallel semi-Lagrangian or physics load-balancing algorithms, however some of the PSTSWM options and the COMMTEST results are relevant to these, and provide data sufficient to make intelligent decisions.

Note that COMMTEST and PSTSWM cannot determine how the different parallel algorithm options interact in the full code, nor show the effect of the different aspect ratios. Using the full code we tested all possible aspect ratios for each of the interesting parallel algorithms and for each total number of processors. We also looked at two problem sizes: T42L18 and T170L18.

5 CCM/MP-2D Performance

In this section we use the data from the kernel codes to determine performance characteristics of the target platforms. We also identify a subset of good parallel algorithm options, and discuss how sensitive performance is to the choices. We then present detailed data on the performance of CCM/MP-2D.

5.1 Single processor performance

CCM/MP-2D has good load balance for a wide range of processor counts. The primary factors of performance and scalability on a given platform are the floating point computation and interprocessor communication rates. Unfortunately, it is difficult to use CCM/MP-2D to determine the relevant computation rates. The computation rate is typically dependent on the problem granularity, and the rate for a single processor run is unlikely to accurately reflect the performance that would be achieved on a given node of a parallel run. CCM/MP-2D has only a small number of supported problem resolutions, limiting the investigation of relevant granularities. For a rough comparison of computational rates of the T3E and the Origin we measured the serial performance of the PSTSWM kernel code for a series of problem sizes (T42, T85, T170) and for a number of different vertical resolutions (L1, L2, L3, L16). PSTSWM computation rates have proven to be good predicters of CCM/MP-2D computation rates for code outside of the column physics. In a more definitive benchmarking exercise, we would also need a serial column physics kernel.

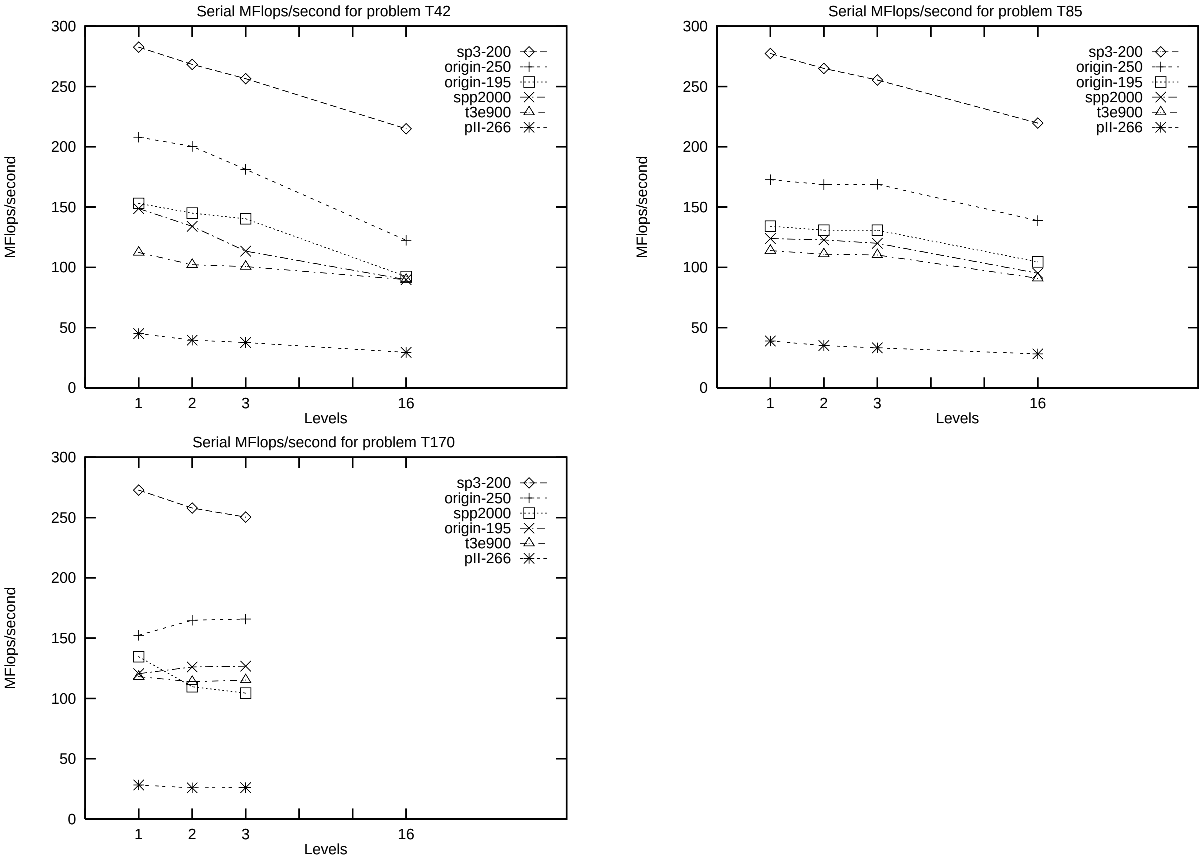


Figure 1: Serial PSTSWM MFlop/second rates.

A graphical comparison of PSTSWM serial performance is given in Fig. 1, where Origin and T3E results are denoted by `origin-250` and `t3e900`, respectively. The metric is Mflops/second, where the MFlop count was measured on a Cray C90, so the results are best interpreted as graphs of normalized inverse time. The results use math libraries where available and the best identified compilers and compiler options that also work with CCM/MP-2D.

The results that are most interesting for this study are

- Cache effects can be seen in the Origin data, with the smallest granularity case (T42L1) having significantly better performance ($>$ 200 MFlops/sec) than the larger granularity cases ($<$ 150 MFlops/sec). Thus there is a potential for improved computational rates as the number of processors increases.
- The T3E shows less sensitivity to problem granularity, at least over this range of problem sizes. The T3E computation rate is also between half and two thirds that of the Origin.

For comparison, the single processor MFlop rate for CCM/MP-2D when simulating two days at a resolution of T42L18 is 94 MFlop/sec on the Origin and 122 MFlop/sec on the SP3. The floating point operation count was calculated on yet a different machine, so these rates are not directly comparable to the PSTSWM results. However, the primary difference in observed performance between the PSTSWM and CCM/MP-2D runs is the effect of the many exponential, logarithm, and trigonometric function calls used in the column physics calculations. Note that this problem size, which is the smallest problem size for which we currently have datasets, is too large to run on a single processor of the T3E at NERSC.

5.2 Peak communication performance

The results in Fig. 2 describe the maximum bandwidth that was observed when exchanging data between two processors using the COMMTEST kernel and MPI, as well as an estimate of the latency associated with the protocol that is used to achieve this bandwidth. Both unidirectional and bidirectional bandwidth results are given, for messages of size 8 Kbytes, 128 KBytes, and 2 MBytes. This is a “not to be exceeded” performance figure in that multiple logically simultaneous exchanges may contend for shared resources. However, CCM/MP-2D is only loosely synchronous, and the progress of the processors will naturally drift to avoid contention for bandwidth. Thus the performance observed in these experiments may be approachable. We find these tests very useful in identifying differences in communication protocols, but rely more on the PSTSWM experiments described in §5.3 to evaluate communication patterns that arise in CCM/MP-2D. Note that both the Origin and the T3E bandwidth rates increase if the SHMEM communication library is used instead of MPI [13]. However, the current version of CCM/MP-2D does not use SHMEM.

The results of importance here show the relative advantage of the T3E over the Origin in achieveable bandwidth. The SPP2000 results show how good MPI performance can be in a shared memory architecture, but these measurements were within a hypernode and do not reflect the bandwidth achieveable between hypernodes on the SPP2000 architecture. Note that the low latency that can be achieved on the T3E [1] is not as evident when using the vendor-supplied MPI.

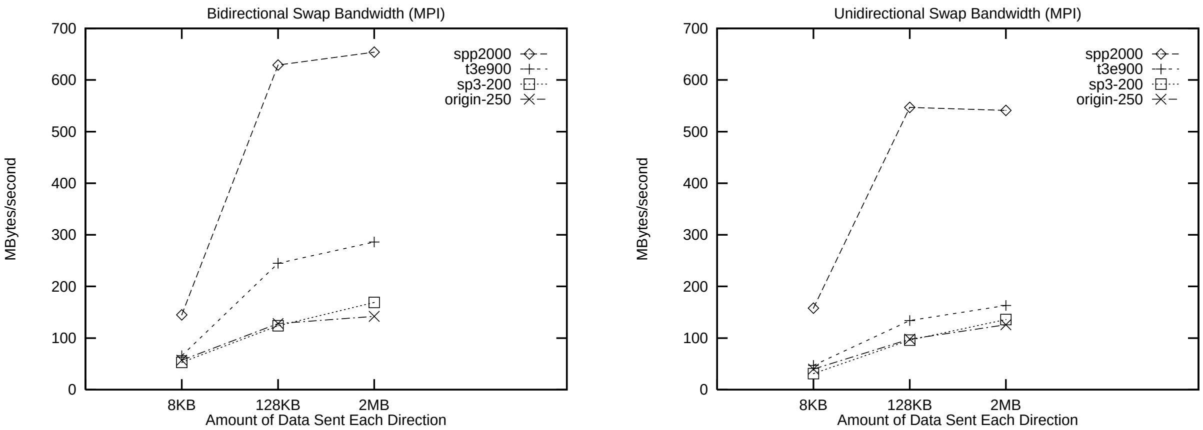
5.3 Parallel algorithm tuning I

The first step in tuning CCM/MP-2D performance is to eliminate the uninteresting tuning options. We did this in two stages. First, a large number of experiments are run using PSTSWM with one-dimensional domain decompositions, decomposing solely in longitude (Px1) or in latitude (1xP) in the physical domain. This isolates the individual parallel algorithms for the Legendre and Fourier transforms. The experiments looked at all supported communication protocols for a number of problem sizes and several numbers of processors, to identify a set of communication protocols that are good for both large and small granularity cases. The results for certain of these algorithms can also be used to infer good/bad protocols for the parallel semi-Lagrangian transport and column physics load balancing algorithms.

Three transpose-based parallel FFT algorithms were examined, each of which employs a data tranpose algorithm that is functionally equivalent to MPI_ALLTOALLV:

- `srtrans`: sends $P-1$ messages using a send-receive protocol to transpose across P processors;
- `swtrans`: sends $P-1$ messages using a swap protocol to transpose across P processors;
- `logtrans`: sends $O(\log(P))$ messages using a swap protocol to transpose across P processors,

where a swap protocol exchanges data between two processors during each interprocessor communication request, while a send-receive protocol sends to one processor and receives from another (potentially different) processor. `srtrans`, `swtrans`, and `logtrans` all use different orderings of interprocessor communication between processors, and `logtrans` sends more data than the other two.



Platform	bidirectional bandwidth (peak MByte/sec)	estimated latency (usec/msg)	unidirectional bandwidth (peak MByte/sec)	estimated latency (usec/msg)
2MByte Message				
Origin	134	38	106	23
T3E	289	68	163	30
128KByte Message				
Origin	130	29	93	17
T3E	245	24	134	20
8KByte Message				
Origin	46	18	36	12
T3E	66	25	47	21

Figure 2: Interprocessor Communication Rates.

Three distributed Legendre transform algorithms were examined, each of which employs a distributed vector sum algorithm that is functionally equivalent to MPI_ALLREDUCE:

- `exchsum`: an exchange-based algorithm that sends $\log_2(P)$ messages using a swap protocol to sum across P processors;
- `halfsum`: a recursive halving-based algorithm that sends $2\log_2(P)$ messages using a swap protocol to sum across P processors;
- `ringsum`: a ring-based algorithm that sends $P-1$ messages using a send-receive protocol to sum across P processors.

`exchsum` sends more data than the other two.

Finally, one distributed FFT was examined:

- `dfft`: sends $O(\log(P))$ messages using a swap protocol to calculate FFT distributed across P processors.

The data generated from these experiments can be viewed at

`http://www.csm.ornl.gov/~worley/studies/protocol.html`

The results relevant to this study are as follows.

Origin.

- The choice of communication protocol is important for optimizing performance.

- For large granularity experiments, the best communication protocol for `dfft` uses a nonblocking send (MPI_ISEND), blocking receive (MPI_RECV) protocol that exploits bidirectional communication and overlaps communication with computation. For small granularity experiments, the protocol using MPI_SENDRECV (and not overlapping communication and computation) is best.
- The best communication protocol for `ringsum` and for the transpose algorithms is either a nonblocking send, blocking recv protocol that exploits bidirectional communication, but does not attempt to hide latency, or the protocol using MPI_SENDRECV.
- The best communication protocol for `halfsum` is either the nonblocking send, blocking receive protocol that is optimal for `ringsum` or a protocol that uses explicit handshaking messages and "ready" sends (MPI_IRECV / handshaking logic / MPI_RSEND) to more precisely control how the messages are exchanged.

T3E.

- The choice of communication protocol is (again) important for optimizing performance.
- `ringsum`, `srtrans` and `swtrans` have the same optimal protocols: the MPI_SENDRECV-based protocol or a protocol that uses buffered sends (MPI_BSEND) to send all of the outgoing data first, followed by a sequence of blocking receive calls to receive the incoming data.
- Either the MPI_SENDRECV or a buffered send, blocking receive protocol is optimal for `dfft`.
- The hypercube-based algorithms `exchsum`, `halfsum`, and `logtrans` have the same optimal protocols. For small granularity cases either the MPI_SENDRECV or a buffered send, blocking receive protocol is optimal. For large granularity cases a nonblocking receive (MPI_IRECV), blocking send (MPI_SEND) protocol that attempts to hide latency is optimal.

5.4 Parallel algorithm tuning II

The second stage in eliminating uninteresting tuning options is to identify the best transpose FFT algorithms and the best distributed Legendre algorithms, using the optimal communication protocols for each algorithm, and also comparing against implementations using the MPI collective communication routines MPI_ALLTOALLV and MPI_ALLREDUCE and against a "generic" algorithm, representing a conservative choice of parallel algorithm and communication protocol that should work on most parallel systems. Figure 3 describes the results of this comparison in terms of the percentage degradation in performance from not using the optimal algorithm.[1] The problem sizes were chosen so as to give the correct granularity for larger problem sizes when used with a two-dimensional domain decomposition. For example, a transpose FFT using 8 processors to solve a problem size of T21L8 has the same granularity as a transpose FFT and a distributed Legendre transform on a 8×8 processor grid when solving a problem of size T42L16.

As can be seen from this data, the variation in timings due to the choice of parallel algorithm is quite high. On the T3E, the MPI_ALLTOALLV-based transpose FFT is consistently better than the alternatives. In contrast, on the Origin `swtrans` is the best single choice. For the distributed Legendre transform, `halfsum` is optimal or near-optimal on both the T3E and the Origin, and MPI_ALLREDUCE should never be used.

These experiments are not ideal for evaluating the different parallel algorithms, as they do not address the process placement and resource contention that occurs in a two-dimensional decomposition. To validate these results, we also ran experiments with a two-dimensional decomposition, using a fixed parallel algorithm in the coordinate direction not being investigated. These experiments have their own, but different, evaluation deficiencies, but agreement between the one and two-dimensional decomposition experiments is a reasonable indicator that the evaluation results are a good basis for choosing parallel algorithms to use with CCM/MP-2D. For the Origin and the T3E, both types of experiments had similar results.

We can also use these experiments to compare performance between the different platforms, as shown in Fig. 4. Here the best measured time for each platform (`time`) is normalized by the minimum time over all platforms (`min`) for the given experiment. This comparison represents an evaluation of the interprocessor communication performance that is more relevant to CCM/MP-2D than the COMMTEST experiments, as it uses communication patterns more like those that occur in CCM/MP-2D. Note that the timings are for the full PSTSWM code, so the results are also influenced by the computation performance. However, the parallel algorithms compared in Fig. 3 differ only in the communication costs, and the high variation in these results indicates that the performance of PSTSWM is strongly

[1] measured time for a given algorithm (`time`) minus the minimum time over all algorithms (`min`) divided by the minimum time

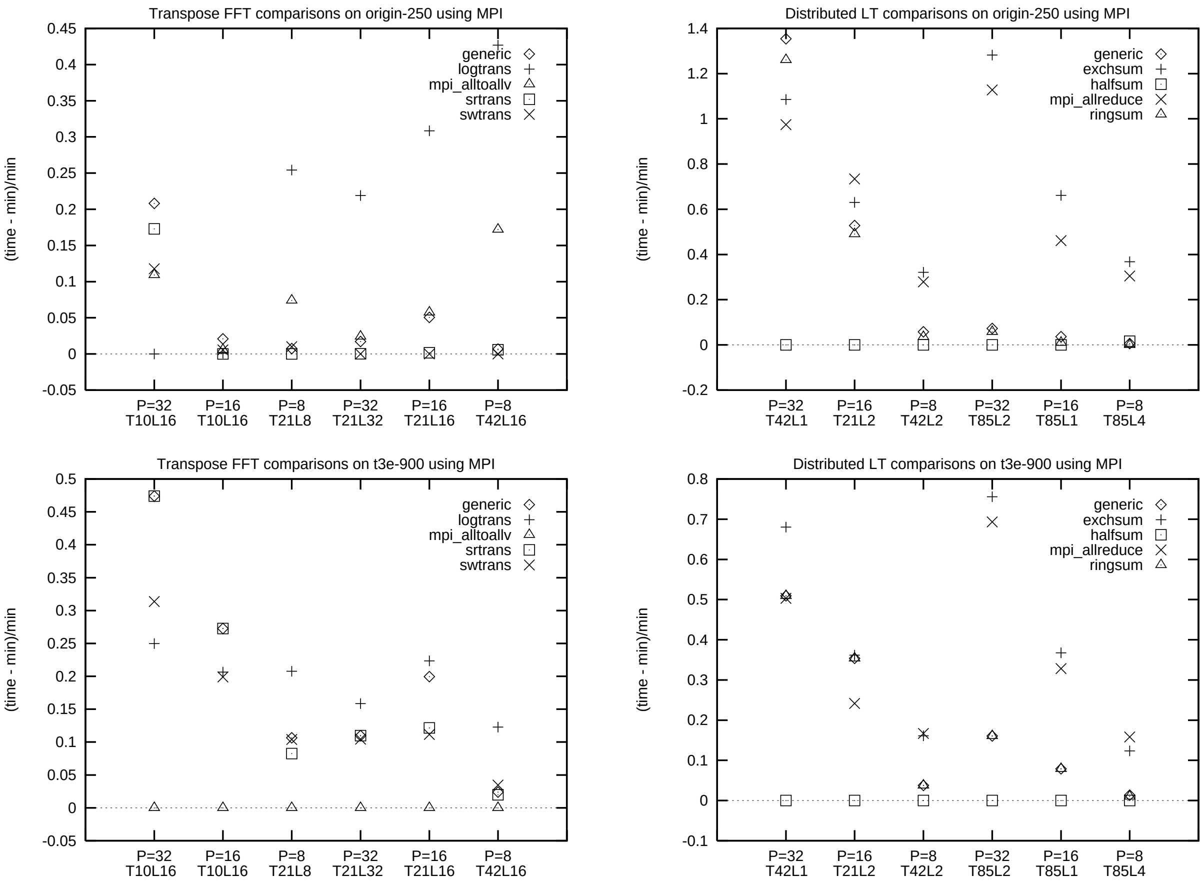


Figure 3: Parallel Algorithm Comparisons.

dependent on the communication performance for these problem sizes, numbers of processors, and platforms. The lines connecting the data in Fig. 4 are used to make it easier to read and interpret the results. There is no meaning associated with the values on the lines between data points. Note that we do not have SPP2000 results for the 32 processor experiments.

The platform comparison data is somewhat difficult to interpret, however it is clear that the Origin performs better than the T3E on all but the smallest granularity cases, when communication cost most strongly dominates the execution time. However, a fast(er) processor is not everything, as the Origin has similar or better performance than the SP3. The platform differences are strongly dependent on the parallel algorithm used in the comparison, as they make different demands on the interconnect.

5.5 CCM/MP-2D algorithm tuning.

Using the PSTSWM experiments, we identified parallel algorithms to examine in CCM/MP-2D. This included using communication protocol sensitivity data to choose communication protocols for updating the halo regions in the parallel semi-Lagrangian algorithm and for swapping vertical columns in the physics load-balancing algorithm.

Origin. Fourteen different algorithms were examined, seven distributed FFT/distributed LT algorithms: **d0**-**d6**, and seven transpose FFT/distributed LT algorithms: **t0**-**t6**. **d0** and **t0** use MPI_SENDRECV-based implementations of the "best" algorithms: `swtrans` for the transpose FFT and `halfsum` for the distributed Legendre transform. **d1**-**d3** and **t1**-**t3** use increasingly exotic implementations, attempting to exploit communication/computation overlap

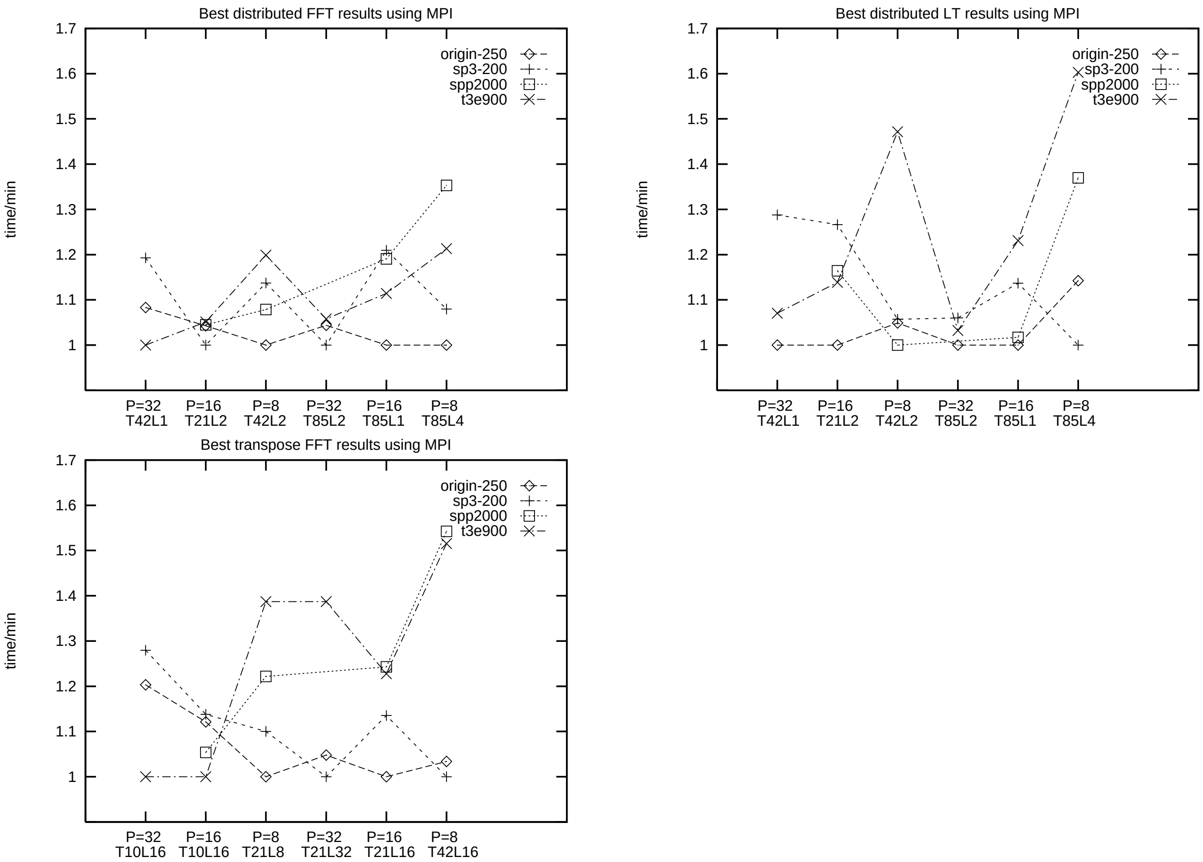


Figure 4: Best Relative PSTSWM performance for one-dimensional decomposition experiments.

or controlling the messaging more precisely with ready send protocols. **d4**-**d6** and **t4**-**t6** are identical to **d0**-**d2** and **t0**-**t2** except that `ringsum` is used instead of `halfsum` for the distributed Legendre transform.

T3E. Five different algorithms were examined, two distributed FFT/distributed LT algorithms: **d0**-**d1**, and three transpose FFT/distributed LT algorithms: **t0**-**t2**. **t0** and **t2** use MPI_ALLTOALLV to implement the transpose FFT algorithm, while **t1** uses `swtrans`. **d0**, **t0**, and **t2** use MPI_SENDRECV-based implementations, while **d1** and **t1** use implementations that attempt to overlap communication with computation.

Experiments. To choose the best parallel algorithm for CCM/MP-2D for each processor count, we examined each parallel algorithm for all aspect ratios and two different process mappings: row-major, assigning each logical processor row to consecutive processors, and column-major, assigning each logical processor column to consecutive processors. Experiments were run using two different problem sizes: T42L18 and T170L18. For T42L18, we measured the time to calculate 9 time steps, which includes 3 timesteps that calculate solar radiation (and use load balancing). For T170L18, we measured the time to calculate 14 time steps, which includes 1 timestep that calculates solar radiation. A more expensive column physics calculation, of absorption and emissivity, does not first occur until a later time step and so is not included in the tests, but this should not effect the parallel algorithm comparisons.

Note that for the 128 processor experiments on the Origin, the application will share processors with system processes. This does not cause significant performance degradation as long as process migration is disabled. This was verified by also running 112 processor experiments, which exhibit scaling behavior similar to that of the 128 processor experiments. However, if process migration is enabled (which is typically the default) then when a system process requires a processor the application processes may start migrating, which destroys data locality and degrades performance.

The results are presented in Tab. 1. The minimum 1D time refers to the minimum timing using a latitude-only domain decomposition. Note that with T42L18, we can use a maximum of 32 processors in the latitude direction with CCM/MP-2D. For T170L18, the maximum is 128 processors. Also note that the choice of parallel FFT algorithm is not important in 1xP decompositions.

Processors	Minimum Time (secs.)	Optimal Aspect Ratio	Optimal Algorithm	Minimum 1D Time	(1D-Min)/Min
T42L18 tuning results on the Origin using MPI					
16	6.0	1x16	t2	6.0	.00
32	3.3	2x16	d0	3.6	.09
64	2.1	4x16	d2	-	-
128	1.4	8x16	d3	-	-
T170L18 tuning results on the Origin using MPI					
64	48.3	4x16	d6	65.5	.36
128	31.5	8x16	t6	61.7	.96
T42L18 tuning results on the T3E using MPI					
16	13.8	1x16	t2	13.8	.00
32	7.1	1x32	t0	7.1	.00
64	4.2	2x32	t0	-	-
128	2.4	4x32	t0	-	-
256	1.5	8x32	t0	-	-
512	1.2	16x32	d0	-	-
T170L18 tuning results on the T3E using MPI					
64	96.1	2x32	t0	97.8	.02
128	48.5	4x32	t0	60.4	.25
256	25.8	8x32	t0	-	-
512	14.8	16x32	t1	-	-

Table 1: CCM/MP-2D parallel algorithm tuning results.

From these data we see that the optimal aspect ratio for CCM/MP-2D is of the form 1xP until the number of latitude processors reaches a maximum number, 16 for the Origin and 32 for the T3E, after which this stays fixed and further processor increments are applied to the longitude direction. This is independent of whether the problem size is T42L18 or T170L18. Once this latitude processor maximum is reached, the performance of the one-dimensional decomposition falls off rapidly, at least for T170L18. For T42L18, the algorithmic limit is reached soon after the "performance" limit.

On the Origin, the distributed FFT is optimal for most cases, while on the T3E the transpose FFT (using MPI_ALLTOALLV) is optimal except for the 512 processor cases. In a rough sense, the T3E performance is half that of the Origin, requiring twice as many processors to achieve similar timings. However, the T3E performance scales reasonably well to higher numbers of processors, while initial experiments indicate that running CCM/MP-2D across multiple 128-processor Origin 2000 machines does not scale well currently.

5.6 CCM/MP-2D benchmarking measurements.

The final step in our evaluation is to benchmark the performance of CCM/MP-2D on the Origin and the T3E using the parallel algorithms and aspect ratios identified previously. This represents our best efforts in "fair benchmarking", short of rewriting the code, which is not feasible. Large scientific simulation codes like CCM3 outlive most computing platforms, and are also constantly being updated. A significant rewrite in order to increase performance on any one given platform is rarely justified. Thus the tuning options currently built into CCM/MP-2D are extremely useful and important.

For T42L18, we measured the performance for a 10 day simulation, reporting the average time per simulated day. For T170L18, we measured the performance for a 2 day simulation. These results are graphed in Fig. 5. Note that these experiments do not include significant I/O. I/O is strongly dependent on the type of experiment being run, and would have made it more difficult to compare the parallel algorithms. Thus, these experiments represent the maximum performance that can be achieved, and a production run may experience lower performance.

To measure scalability, Fig. 5 also shows the effective MFlops per second per processor for a given total number of processors. For a perfectly scalable algorithm, the curve would be flat. Note that the floating operation counts for T42L18 and T170L18 were determined on different machines and in different ways, so are not directly comparable. However, they are internally consistent, allowing a convenient way to compare both the raw performance and the difference in scalability between the Origin and the T3E.

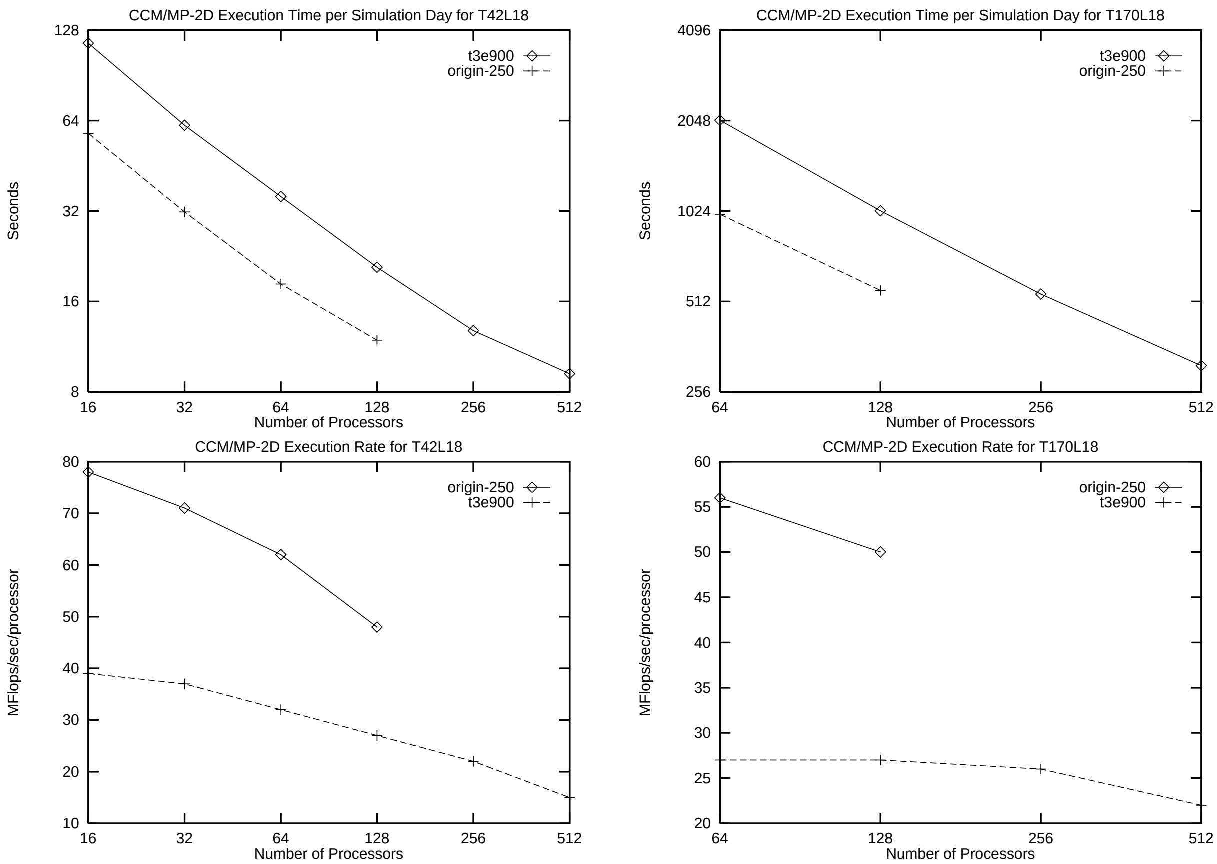


Figure 5: CCM/MP-2D benchmarking experiments.

From these results it is clear that reasonable performance can be obtained for large processor configurations if care is taken in tuning the parallel algorithms. While performance does drop off for large numbers of processors for the T42L18 problem size, it is still useful to use these configurations for the cases when runtime is more important than efficiency. As in the earlier comparisons, the T3E requires approximately twice as many processors as the Origin to achieve the same performance.

Table 2 describes the source of performance degradation for these runs. CCM/MP-2D was rerun with performance instrumentation enabled. Timings were compared with the uninstrumented experiments to verify that the instrumentation was not perturbing the performance significantly. The instrumentation data was used to estimate the time spent in five different categories:

Serial Work: time spent in computation found in serial implementation;

Comm: time spent in interprocessor communication;

Imbal: time spent idle due to load imbalance;

Copy: time spent copying data in support of interprocessor communication;

Dupl: time spent in redundant spectral calculations.

where the time is summed over all processes. *Serial Work* should be constant as a function of the number of processors. If it increases as the number of processors increases, then this indicates that the computational rate is decreasing. If *Serial Work* decreases, then the rate is increasing. We used the *Serial Work* estimate for the experiment using the smallest number of processors to define a baseline, and refer to the relative change from this baseline as the *Rate*.

Processors	Percentage Efficiency	Percentage Comm	Imbal	Copy	Dupl	Rate
T42L18 performance analysis on the Origin using MPI						
16	83.1	10.7	1.5	1.1	3.5	-
32	75.9	15.4	3.7	2.5	3.3	-0.7
64	65.6	22.1	5.0	3.5	2.9	0.9
128	50.9	32.5	6.3	2.3	2.2	5.9
T170L18 performance analysis on the Origin using MPI						
64	70.4	16.7	6.2	3.3	3.4	-
128	63.0	23.2	7.2	4.2	3.0	-0.6
T42L18 performance analysis on the T3E using MPI						
16	86.8	6.5	2.4	0.9	3.4	-
32	81.4	8.4	2.2	1.0	6.6	0.4
64	70.3	11.1	4.2	4.9	5.7	3.9
128	60.5	15.1	4.5	5.8	4.8	9.3
256	49.3	17.5	6.1	7.1	4.1	15.9
512	34.3	23.5	6.1	7.2	2.9	26.0
T170L18 performance analysis on the T3E using MPI						
64	72.1	6.8	11.0	3.2	6.9	-
128	72.1	7.7	11.3	3.5	7.2	-1.9
256	68.3	8.9	10.6	4.0	6.6	1.5
512	59.1	9.7	10.0	4.2	5.8	11.3

Table 2: CCM/MP-2D performance degradation analysis.

The interpretation of these results is aided by an understanding of the relationship between communication overhead and the number of processors for CCM/MP-2D. A rough rule of thumb is that communication costs increase or decrease slower than computation costs as the number of processors increase. However, any increase in the communication overhead is moderate for the optimal algorithms, and the communication cost functions are relatively well behaved. Thus, from the empirical results, it is clear that the major cause of performance degradation for the T42L18 problem is the small problem size. On the Origin, this is primarily reflected in the rapidly increasing percentage of time spent in interprocessor communication and the associated data copying. On the T3E, communication costs also increase, although not as quickly. However, the computational rate also degrades dramatically as the amount of work per processor decreases. Most data are used a very small number of times before being flushed from the cache.

For the larger T170L18 problem, communication costs are also the dominant cause of performance degradation on the Origin. Here the cause is the high bandwidth requirements associated with the large problem size (for all numbers of processors). On the T3E, communication costs are comparable to the idle time caused by load imbalance between the equitorial and polar regions in the semi-Lagrangian transport. The T3E also suffers from computation rate degradation for the largest number of processors. However, this is not significant until then.

6 Conclusions

From the empirical experiments, it is clear that a one-dimensional domain decomposition as implemented in CCM/MP-2D is the best choice when using a small number of processors. However, a two-dimensional decomposition allows larger numbers of processors to be exploited efficiently, and is likely to become a requirement for future versions of the CCM. Whether an explicit message-passing version is required to exploit the additional parallelism is not clear as yet.

The empirical experiments also reinforce our opinion that it is important to tune the parallel algorithms on each platform. For this to be feasible the tuning options must be built into the production code, and parallel algorithm kernel codes must be available for tuning.

Finally, the empirical results indicate that a 128 processor Origin 2000 achieves performance similar to that of a 256 processor T3E-900. However, larger numbers of processors can be used efficiently on the T3E, especially for large problem sizes.

References

[1] A. Anderson, J. Brooks, C. Grassl, and S. Scott, *Performance of the CRAY T3E Multiprocessor*, in Proceedings of SC97, Los Alamitos, CA, 1997, IEEE Computer Society Press.

[2] L. J. Bath, J. Rosinski, and J. Olson, *Users' guide to NCAR CCM2*, NCAR Tech. Note NCAR/TN–379+IA, National Center for Atmospheric Research, Boulder, Colo., 1992.

[3] Department of Energy, *Building an advanced climate model: Progress plan for the CHAMMP climate modeling program*, DOE Tech. Report DOE/ER–0479T, U.S. Department of Energy, Washington, D.C., December 1990.

[4] J. B. Drake, I. T. Foster, J. G. Michalakes, B. Toonen, and P. H. Worley, *Design and performance of a scalable parallel community climate model*, Parallel Computing, 21 (1995), pp. 1571–1591.

[5] I. T. Foster, B. Toonen, and P. H. Worley, *Performance of parallel computers for spectral atmospheric models*, J. Atm. Oceanic Tech, 13 (1996), pp. 1031–1045.

[6] I. T. Foster and P. H. Worley, *Parallel algorithms for the spectral transform method*, SIAM J. Sci. Comput., 18 (1997), pp. 806–837.

[7] G. A. Geist, A. L. Beguelin, J. J. Dongarra, W. Jiang, R. J. Manchek, and V. S. Sunderam, *PVM: Parallel Virtual Machine - A Users Guide and Tutorial for Network Parallel Computing*, MIT Press, Boston, 1994.

[8] J. J. Hack, B. A. Boville, B. P. Briegleb, J. T. Kiehl, P. J. Rasch, and D. L. Williamson, *Description of the NCAR Community Climate Model (CCM2)*, NCAR Tech. Note NCAR/TN–382+STR, National Center for Atmospheric Research, Boulder, Colo., 1992.

[9] J. T. Kiehl, J. J. Hack, G. Bonan, B. A. Boville, D. L. Williamson, and P. J. Rasch, *The National Center for Atmospheric Research Community Climate Model: CCM3*, J. Climate, 11 (1998), pp. 1131–1149.

[10] National Research Council, *Capacity of U.S. Climate Modeling to Support Climate Change Assessment Activities*, National Academy Press, Washington, D.C., 1998.

[11] W. Washington and C. Parkinson, *An Introduction to Three-Dimensional Climate Modeling*, University Science Books, Mill Valley, CA, 1986.

[12] D. L. WILLIAMSON AND P. J. RASCH, *Two-dimensional semi-Lagrangian transport with shape-preserving interpolation*, Mon. Wea. Rev., 117 (1989), pp. 102–129.

[13] P. H. WORLEY, *Impact of Communication Protocol on Performance*, in Proceedings of the Second International Workshop on Software Engineering and Code Design in Parallel Meteorological and Oceanographic Applications, June 1998, pp. 277–288.

[14] P. H. WORLEY AND B. TOONEN, *A users' guide to PSTSWM*, Tech. Report ORNL/TM–12779, Oak Ridge National Laboratory, Oak Ridge, TN, July 1995.