Performance Evaluation of the IBM SP and the Compaq AlphaServer SC [*]

Patrick H. Worley[†]
Computer Science and Mathematics Division
Oak Ridge National Laboratory[‡]
P.O. Box 2008, Bldg. 6012
Oak Ridge, TN 37831-6367
worleyph@ornl.gov

ABSTRACT
Oak Ridge National Laboratory (ORNL) has recently installed both a Compaq AlphaServer SC and an IBM SP, each with 4-way SMP nodes, allowing a direct comparison of the two architectures. In this paper, we describe our initial evaluation. The evaluation looks at both kernel and application performance for a spectral atmospheric general circulation model, an important application for the ORNL systems.

1. INTRODUCTION
A common architectural design for supercomputers from US vendors is a cluster of modestly parallel shared memory (SMP) nodes. Two of the more popular examples of this design are the IBM SP and the Compaq AlphaServer SC. The SP has been selected in recent procurements at the National Center for Atmospheric Research (NCAR), the National Centers for Environmental Prediction, the National Energy Research Scientific Computing Center (NERSC), the North Carolina Supercomputer Center, Oak Ridge National Laboratory (ORNL), and the San Diego Supercomputer Center. The recently announced AlphaServer SC is being installed at NCAR, Lawrence Livermore National Laboratory, ORNL, and a number of commercial sites.

By June 2000, Oak Ridge National Laboratory will have 256+ processor installations of both systems, each using 4-way SMP nodes. In preparation for their arrival, we have been porting and tuning codes on the current ORNL SP and AlphaServer SC, which have 62 Winterhawk II 4-way SMP nodes and 16 ES40 4-way SMP nodes, respectively. In this paper, we describe our current evaluation results.

The first step in our evaluation has been the porting and tuning of a large spectral atmospheric general circulation code, CCM/MP-2D. This code is a highly parallel implementation of the NCAR Community Climate Model CCM3. CCM/MP-2D was developed by researchers at Argonne National Laboratory, Oak Ridge National Laboratory and the National Center for Atmospheric Research.

CCM/MP-2D has many tuning parameters, and a hierarchy of tests was employed to establish reasonable values for these parameters. While motivated by the tuning process, the results of the low level communication and computation tests and the parallel algorithm comparisons have wider application, representing a rigorous evaluation of many aspects of system performance.

2. BACKGROUND
2.1 Platforms
The current IBM SP at ORNL, henceforth referred to as "SP-375", is a cluster architecture utilizing Winterhawk II SMP nodes. Each node has four 375-Mhz POWER3+ processors, each with an 8MB L2 cache, all sharing a 1.6GB/sec bus to memory. Internode communication is routed through a multistage interconnect with 300 MB/sec bidirectional and 150 MB/sec unidirectional peak point-to-point bandwidths. For our experiments, the system was running AIX version 4.3.3 and PSSP version 3.1.1.

The Compaq AlphaServer SC, henceforth referred to as "AlphaSC-500", is a cluster architecture utilizing ES40 SMP nodes. Each node in the system has four 500-Mhz Alpha 21264 (EV6) processors, each with a 4MB L2 cache, all shar-

[*]Research sponsored by the Atmospheric and Climate Research Division and the Office of Mathematical, Information, and Computational Sciences, U.S. Department of Energy under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.

[‡]Oak Ridge National Laboratory is managed by UT-Battelle, LLC for the United States Department of Energy under Contract No. DE-AC05-00OR22725

International Conference on Supercomputing May 8-11, 2000, Santa Fe, New Mexico USA

ing two 2.6GB/sec buses to memory. Internode communication is routed through a "fat tree" interconnect with 400 MB/sec bidirectional and 200 MB/sec unidirectional peak point-to-point bandwidths. For our experiments, the system was running Digital UNIX V5.0 and RMS version 2.36.

The planned upgrade to the ORNL SP involves adding more nodes, but using the same interconnect technology and SMP nodes. The upgrade to the AlphaServer SC system will also use the same interconnect technology, but with 667-MHz Alpha 21264 (EV67) processors, each with an 8MB L2 cache, in a 4-way SMP node. The memory system in the new AlphaSC node will be a slight modification of that used in the current system.

2.2 CCM/MP-2D

The Community Climate Model (CCM) is an atmospheric general circulation model developed at NCAR [1, 6, 7]. Versions of it are used as the atmospheric component of both the Department of Energy's Parallel Climate Model and NCAR's Climate System Model.

The vertical and temporal aspects of CCM are represented by finite difference approximations. The spectral transform method is employed to compute the dry dynamics [6, 8]. 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. Horizontal derivatives and linear terms involving these variables are calculated and combined to form the dynamical right hand sides in spectral space. 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. These "physics" computations involve only the vertical column above each longitude-latitude grid point and are thus 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 [9] on the physical grid.

CCM/MP-2D is a message-passing parallel implementation of the CCM that uses a two-dimensional domain decomposition [2]. The longitude and latitude dimensions are decomposed, and the resulting blocks are combined to define a decomposition into longitude-latitude patches, leaving the vertical dimension undecomposed. This decomposition naturally defines a virtual two-dimensional processor grid, with rows representing common latitude assignments and columns representing common longitude assignments.

Given this decomposition and the independence between vertical columns, 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 these are filled, all needed information is local to each processor.

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 [2, 5].

3. METHODOLOGY

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 and hide latency. 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. 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 and evaluating the performance of CCM/MP-2D, we used three kernel codes: COMMTEST, CRM, and PSTSWM.

COMMTEST measures the performance of exchanging data between two or more processors. For these experiments we looked at the "peak achievable" rate for exchanging data between two processors. The distinguishing feature of this test code is that it uses the same communication primitive wrappers used in CCM/MP-2D, so 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.

CRM is a standalone version of the radiation model used in the CCM[7]. For these experiments we looked at the computational rate achieved in computing a set of independent columns. A variety of compiler options and math libraries were examined, but only the best performing choices are presented here.

PSTSWM is a parallel spectral transform shallow water model [4, 10] that is an accurate representation of the parallel algorithms used for the dry dynamics in 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 these 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, CRM, and PSTSWM cannot determine how the different parallel algorithm options interact in the full code, nor do they 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, corresponding to 18 vertical levels and the horizontal grid resolutions described in Tab. 1.

T42L18 is the problem size used in current climate simulations in PCM and CSM. T170L18 represents a problem size of interest for future simulations.

	physical domain		spectral domain
	longitude	latitude	spectral coefficients
T42	128	64	946
T170	512	256	14706

Table 1: Horizontal Grid Resolutions

4. EVALUATION

4.1 Computation

CCM/MP-2D has good load balance for a wide range of processor counts. The determining factors for 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. For a rough comparison of computational rates on the SP-375 and the AlphaSC-500 we ran experiments using PSTSWM and CRM.

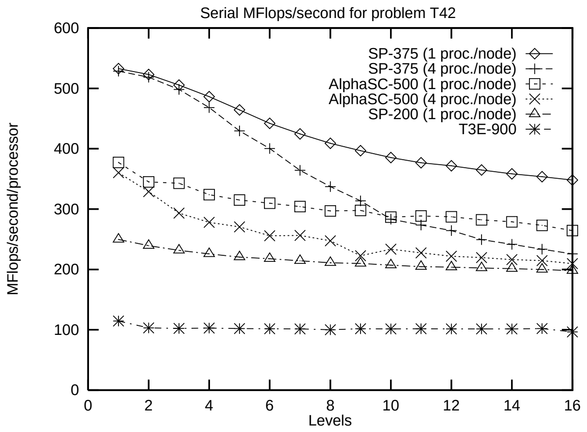
4.1.1 PSTSWM

For the first experiment, we measured the serial performance of the PSTSWM kernel code. PSTSWM computation rates have proven to be good predictors of CCM/MP-2D computation rates for code outside of the column physics. We examined performance for the two horizontal resolutions (T42, T170) and with a varying number of vertical levels. This reflects a range of problem granularities that would arise when using the transpose algorithm, which decomposes over the vertical dimension.

We also ran this experiment in two different ways. First, a single processor in the 4-way SMP node was used, to evaluate peak processor performance. Second, all processors in a node were used, solving identical problems simultaneously, to determine the achieveable node performance.

For the most part, the spectral transform algorithm acts on individual horizontal layers (longitude-latitude). However, in PSTSWM and CCM, the vertical index is the second index (after longitude) and all layers are transformed as a block. Thus, increasing the number of vertical levels changes the cache and memory access patterns. Also, the hardware performance monitor on a C90 indicated that the ratio of floating-point operations to memory references is less than 1.3 for these problem sizes. Thus, for large numbers of vertical levels, serial PSTSWM performance is very sensitive to the performance of the memory subsystem. This is especially true when all processors in a node are computing.

A graphical comparison of PSTSWM serial performance is given in Fig. 1. The metric is MFlops/sec/processor, where the MFlop counts were measured on a Cray C90 (using the harware performance monitor), and validated on an SGI Origin 2000 using the `-ideal` option of the SpeedShop performance analysis tool to count basic blocks in the executable. The timings use math libraries and the best identified compilers and compiler options that also work with CCM/MP-2D. Results for an IBM SP with Winterhawk I 2-way SMP nodes (200MHz POWER3 processors each with



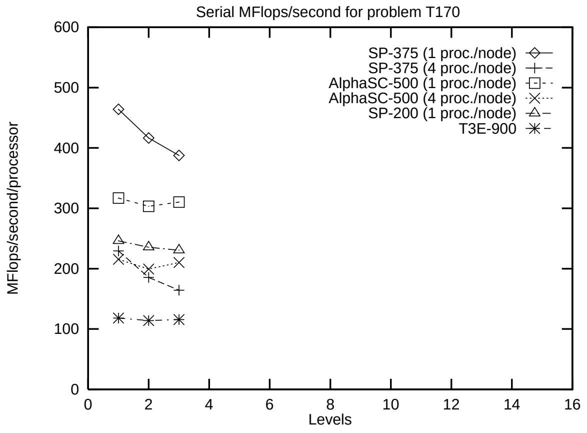


Figure 1: Serial PSTSWM MFlop/sec rates.

a 4MB L2 cache) and a T3E-900 (with 450MHz Alpha EV5 processors each with a 96KB L2 cache) are included to provide additional context. These are denoted by `SP-200` and `T3E-900`, respectively.

The results that are most interesting for this study are as follows:

- The performance of the SP-375 is excellent when the problem fits in cache, exceeding the AlphaSC-500 performance by 40% and the performance of the SP-200 by 100%. Even when the problem size grows too large for the cache, the SP-375 performance remains at least 25% and 80% better than the AlphaSC-500 and SP-200, respectively.
- The EV6 processor, with its 4MB L2 cache, and the ES40 memory subsystem used in the AlphaSC-500 combine to triple the performance achieved by the EV5 processor used in the T3E with only a small increase in clock speed. The AlphaSC-500 also outperforms the SP-200, which has the same size L2 cache, by approximately 50%.
- Contention for the memory bus when all processors in the node are computing can severely degrade the per processor performance on both the SP-375 and the AlphaSC-500. The SP-375 is more sensitive however, and in the worst case (T170, 3 vertical levels), the SP-375 performance falls below that of the AlphaSC-500.

4.1.2 CRM

For the second experiment, we measured the serial performance of the CRM kernel code, which should be an accurate estimate of the performance in the most time consuming part of the column physics. In the CCM, the column physics routines also work with arrays whose first two indices are longitude and vertical level, respectively. In the block decomposition of the physical domain described earlier, the longitude dimension may be decomposed over the processors, but the vertical dimension is not. To examine the impact of the domain decomposition on the computation rate, we modified CRM to allow us to vary the length of the longitude dimension.

In these experiments, we varied the longitude dimension from 1 to 512 (the maximum longitude dimension for T170) for columns with 18 vertical levels. We also added an outer loop to keep the total number of vertical columns computed fixed at 512. This avoids problems with timing runs of vastly different duration. It also represents the situation in which the total number of processors is fixed and we are looking at the effect of varying the aspect ratio of the logical processor grid. Finally, we also ran the experiment using both one processor per node and 4 processors per node, as in the PSTSWM experiments.

Unlike PSTSWM, the column physics routines are much more computation-intensive, both in the use of the data and in the operations performed. For example, over 99% of the floating-point operations in PSTSWM are floating-point multiply or add. In contrast, over 6% of the floating-point operations in CRM are `sqrt` and 3% are divide or reciprocal.

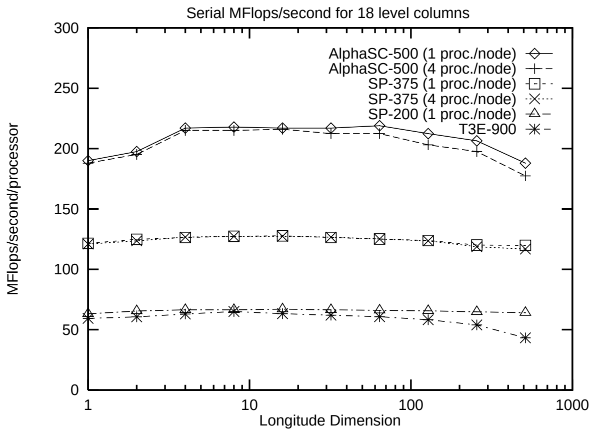


Figure 2: Serial CRM MFlop/sec rates.

A graphical comparison of CRM serial performance is given in Fig. 2. The metric is MFlops/sec/processor, where the MFlop counts were measured on an SGI Origin 2000 using the `-ideal` option of SpeedShop. Note that we did not weight these counts. For example, `sqrt` is counted as a single floating-point operation. In consequence, these results are best interpreted as graphs of normalized inverse time.

As before, the timings use math libraries and the best identified compilers and compiler options that also work with CCM/MP-2D.

From these data we draw the following conclusions:

- Performance of the radiation calculations is not very sensitive to the domain decomposition.
- Contention for the memory bus is not a performance issue for the radiation calculations.
- The AlphaSC-500 processor is 50% faster than the SP-375 for these calculations, and over 3 times faster than the SP-200 and T3E-900.
- The ratio of the SP-375 and SP-200 performance is essentially the same as the ratio of their respective clocks.

In summary, the SP-375 achieves better performance than the AlphaSC-500 on the serial PSTSWM kernel, while the AlphaSC-500 is the better performer on the CRM kernel code. Because of this lack of agreement between the two evaluations, we are unable to predict whether the SP-375 or AlphaSC-500 is likely to be faster for the CCM. It will depend on the mix of the "PSTSWM-like" and "CRM-like" code in the application, which will itself depend on the problem size. But these experiments do show relative weaknesses and strengths of the two different processor and node architectures.

4.2 Communication

For tuning, COMMTEST is used to identify performance differences in the many MPI communication protocols that can be used to exchange data. Here, we use COMMTEST to determine the maximum bidirectional exchange rate, looking at differences between the two platforms.

The communication rate is measured as the sum of the number of bytes sent and received by a single processor divided by the largest elapsed time for any of the participating processors, where MPI_BARRIER is used to synchronize the processors before beginning a measurement. Care is taken to invalidate the data cache between measurements, as well as other "devices" necessary to collect meaningful measurements.

Measuring bandwidth on clusters of shared memory nodes is complicated, as there are many different measures to consider. For the SP-375 and AlphaSC-500, we measured bandwidths for five different cases:

0-1: Exchanging data between processors 0 and 1, on the same node.

0-1,2-3: Simultaneous exchanges between processors 0-1 and 2-3, i.e., each processor in a node exchanging data with another processor in the smae node.

0-4: Exchanging data between processor 0 on one node and processor 4 on a different node.

0-1-2-3-4-5-6-7-0: Simultaneous send/receive between neighbors in a logical processor ring covering two nodes.

0-4,1-5,2-6,3-7: Simultaneous exchanges between processors 0-4, 1-5, 2-6, and 3-7, i.e., each processor on one node exchanging data with the corresponding processor on the other node.

In all cases, bandwidth was measured for the slowest single exchange. Therefore, bandwidth contention in the second, fourth and fifth cases appears as decreased bandwidth measurements. The IBM User Space protocol was used by MPI on the SP. On both the AlphaSC-500 and SP-375, the MPI library uses shared memory for communication within a node. The external network is accessed only for communication between nodes.

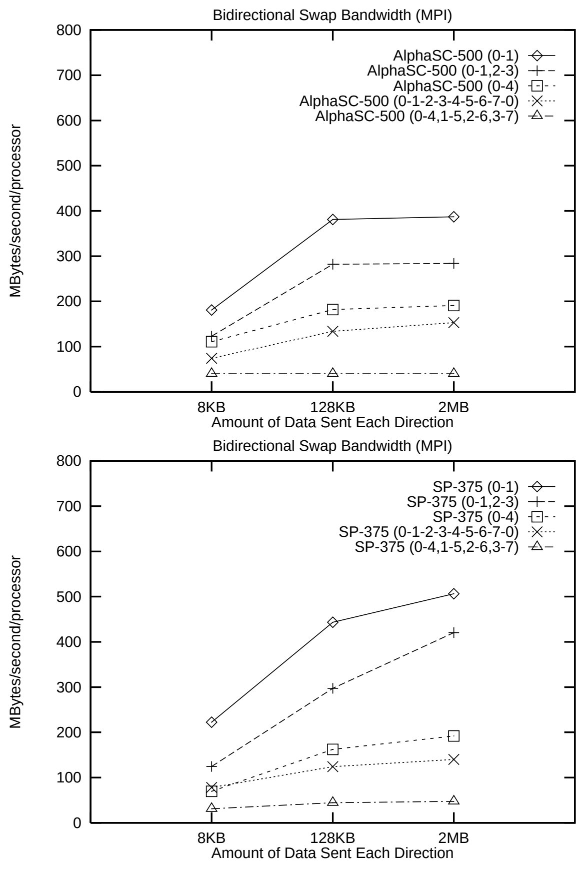


Figure 3: Interprocessor Communication Rates.

The results in Fig. 3 describe the maximum bandwidth that was observed when exchanging data of sizes 8 KBytes, 128 KBytes, and 2 MBytes using the vendor-supplied MPI. For these simple metrics, the peak SP-375 performance within a node (503 MBytes/sec) is significantly better than that of the AlphaSC-500 (387 MBytes/sec). Between nodes, the peak performance is nearly identical (192 MBytes/sec vs. 189 MBytes/sec). This latter result is somewhat misleading. The current network interconnect card on the AlphaSC-500 does not support bidirectional communication efficiently, and the AlphaSC-500 peak unidirectional internode bandwidth is identical to its peak bidirectional bandwidth, and significantly greater than the 140 MBytes/sec peak unidirectional bandwidth we measured on the SP-375. Thus, for parallel algorithms that do not request simultaneous bidirectional communication, the AlphaSC-500 will perform better in internode communication.

Note that both systems are subject to contention for bandwidth in a synchronous algorithm (as represented by experiments **0-1-2-3-4-5-6-7-0** and **0-4,1-5,2-6,3-7**). As CCM/MP-2D is only loosely synchronous, the progress of the processors will naturally drift to avoid some of this contention for bandwidth.

The AlphaSC-500 has the advantage with regards to latency, as shown in the following table, especially between nodes. Here the latency is defined to be half the time required to send and receive back an 8 byte message.

	Intranode (**0-1**)	Internode (**0-4**)
AlphaSC-500	6	6
SP-375	9	21

Table 2: Time required to send an 8 byte message (microseconds).

These results lead us to expect the communication performance of the AlphaSC-500 to be somewhat better for latency-sensitive parallel algorithms, and the SP-375 to be better for bandwidth-sensitive parallel algorithms with significant intranode communication.

4.3 PSTSWM Parallel Algorithms

The first step in tuning the CCM/MP-2D parallel performance was to eliminate the uninteresting tuning options. We did this in two stages. First, a large number of experiments were 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. We then repeated the experiments using two dimensional decompositions (PxQ), but fixing the tuning options for either the parallel Legendre or Fourier transform parallel algorithm, and varying the options for the other. In these experiments, we used all processors in the node; i.e., 8 processor experiments utilized 2 nodes.

We examined three transpose-based parallel FFT algorithms, each of which employs a data tranpose algorithm that is functionally equivalent to MPI_ALLTOALLV, three distributed Legendre transform algorithms, each of which employs a distributed vector sum algorithm that is functionally equivalent to MPI_ALLREDUCE, and one distributed FFT algorithm. 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.

Space limitations prevent us from describing the details of this tuning step. However, the methodology used is described in more detail in [3], and the data generated from these experiments can be viewed at:

`http://www.epm.ornl.gov/~worley`

The second step in eliminating uninteresting tuning options was to identify the best transpose FFT algorithms and the best distributed Legendre algorithms. We used the optimal communication protocols for each algorithm, and also compared against implementations using the MPI collective communication routines MPI_ALLTOALLV and MPI_ALLREDUCE. 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. Again, we refer the reader to the above URL for details on this study. We briefly discuss the performance of the MPI collective communication routines below.

On the SP-375, the MPI_ALLTOALLV-based transpose FFT performed well for the smaller granularity cases, but was not as competitive for the large granularity cases. The performance degradation (for the entire code, as compared to runs using the empirically-determined optimal algorithm) varied from 2% to 20%. In contrast, on the AlphaSC-500, the MPI_ALLTOALLV-based transpose performed reasonably well for large granularity cases and badly for the small granularity cases, with performance varying from 4% to 400% slower than the optimum. Note that the 400% slowdown is repeatable, but is anomalous compared to the other timings.

On the AlphaSC-500, the MPI_ALLREDUCE-based distributed Legendre transform performance varied from 8% to 48% slower than the optimum, doing best for the smallest granularity case. On the SP-375, the performance of the MPI_ALLREDUCE-based distributed Legendre transform was very poor in all cases, ranging from 80% to over 200% slower than the optimum. In summary, the performance of MPI collective communications should be examined carefully on both systems before use in performance-sensitive code.

These experiments also allow us to compare performance between the two platforms, as shown in Fig. 4. Here the minimum 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 computational performance. However, as indicated by the performance variation seen in the evaluation of the MPI collective communication routines described above, the performance of PSTSWM is strongly dependent on the communication performance for these problem sizes, numbers of processors, and platforms.

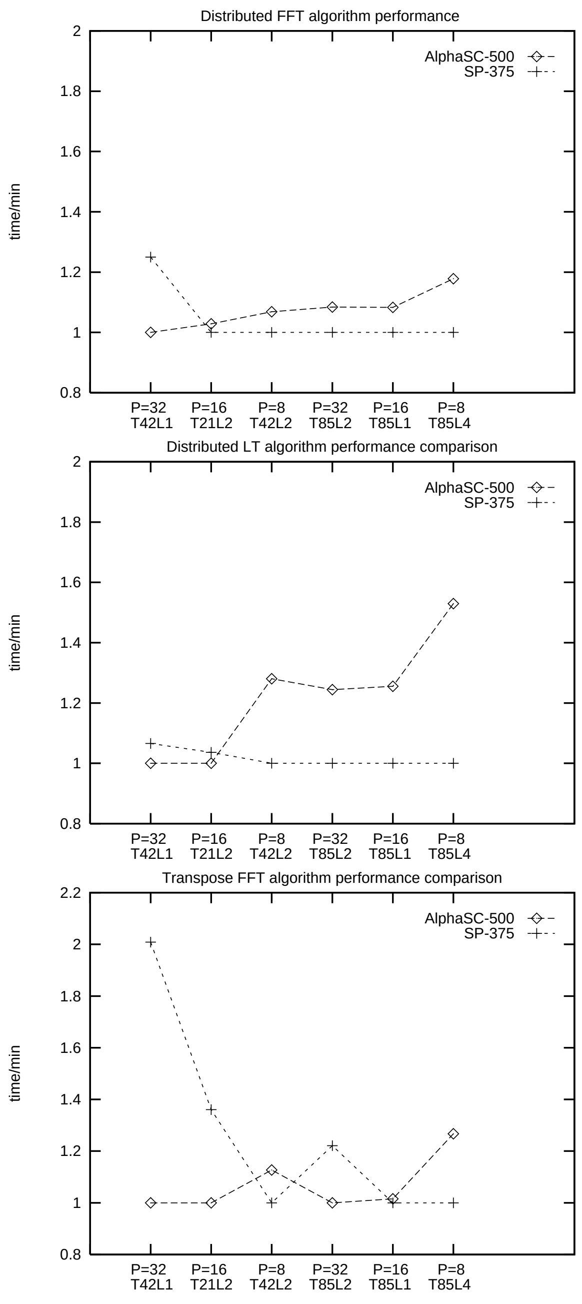


Figure 4: Relative PSTSWM parallel performance for one-dimensional decomposition experiments.

These data indicate that the SP-375 outperforms the AlphaSC-500 when bandwidth is most important (larger problem sizes, smaller numbers of processors, distributed transforms), while the AlphaSC-500 performs best when latency costs are significant (smaller problem sizes, larger numbers of processors, transpose-based transforms). These results agree with the results of the COMMTEST experiments.

4.4 CCM/MP-2D Parallel Algorithms

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. For the AlphaSC-500, seven different algorithms were examined, three distributed FFT/distributed LT algorithms and four transpose FFT/distributed LT algorithms. For the SP-375, ten different algorithms were examined, four distributed FFT/distributed LT algorithms and six transpose FFT/distributed LT algorithms.

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.

Space constrains discussion of these experiments here, but Fig. 5 indicates the type of results. These graphs describe the performance of each of the candidate algorithms for each aspect ratio and processor ordering relative to the minimum timing (`(time-min)/min`). Results are graphed for T42L18 on 32 processors. As can be seen, both the aspect ratio and the choice of parallel algorithm (even among the set of "good" candidate algorithms) can have a large impact on performance, with a 20% variation in performance common for a given aspect ratio, and up to 100% differences in performance across the aspect ratios. Over all, the AlphaSC-500 and SP-375 have remarkably similar performance sensitivities, with an optimal aspect ratio of 2x16 for both. The optimal algorithms are not the same, however. Note that the outlier for the 1x32 aspect ratio on the SP-375 graph corresponds to using the MPI_ALLREDUCE-based algorithm, and is repeatable. Other data generated from these experiments can be viewed at the previously mentioned URL.

4.5 CCM/MP-2D Benchmarks

The final step in the evaluation was to benchmark the performance of CCM/MP-2D on the AlphaSC-500 and the SP-375 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 the CCM 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.

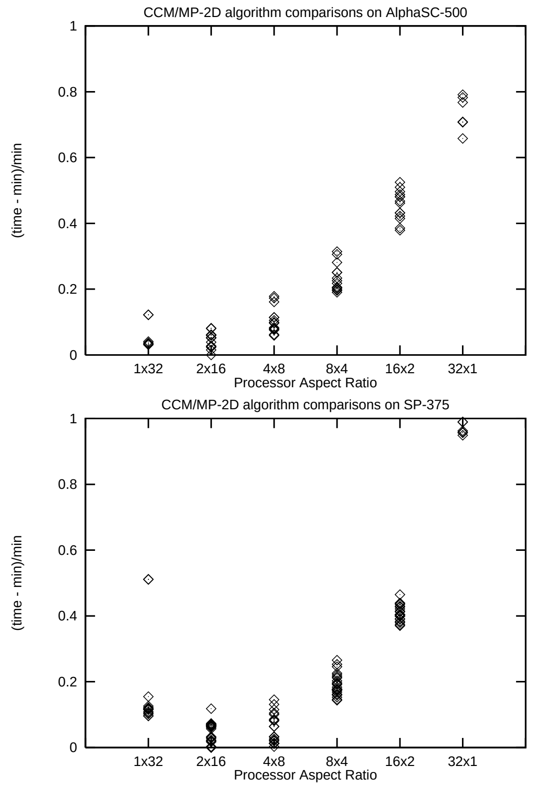


Figure 5: CCM/MP-2D Parallel Algorithm Comparisons for T42L18.

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. 6. Results for a 128 processor SGI Origin 2000 with 250MHz processors, a 644 processor T3E-900, and a 512 processor IBM SP using 2-way Winterhawk I SMP nodes are included to provide additional context. These are denoted by `Origin2000-250`, `T3E-900`, and `SP-200`, respectively. The tuning studies described earlier were also performed on these systems, and the results reflect the optimized performance. In all cases, all processors in an SMP node, up to the maximum number of processors, were used in these experiments. Thus, a 16 processor run would use 8 nodes on the SP-200 and 4 nodes on the SP-375 and AlphaSC-500.

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 other aspects of system performance. Thus, these experiments represent the maximum performance that can be achieved, and a production run with I/O included will experience lower performance.

To measure scalability, Fig. 6 also contains graphs showing the effective MFlops per second per processor for a given total number of processors. For a perfectly scalable algorithm, the curve would be level. The MFlop counts were measured on an SGI Origin 2000 using the `-ideal` option

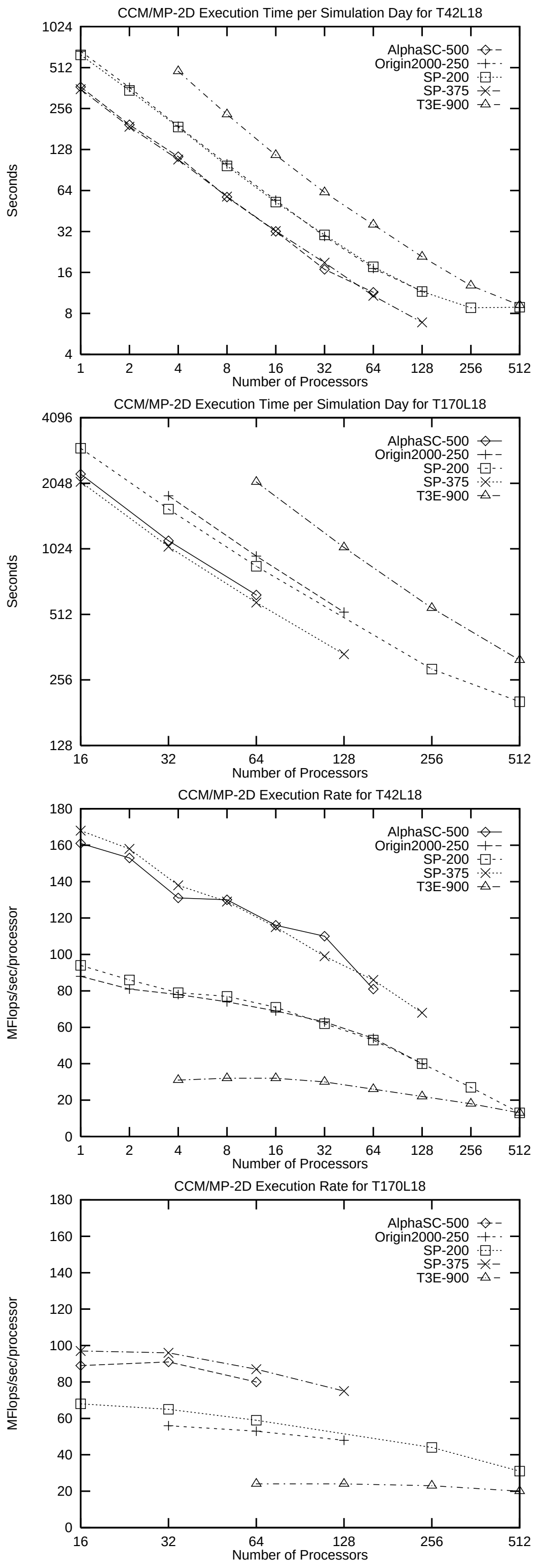


Figure 6: CCM/MP-2D benchmarking results.

of SpeedShop. As with CRM, a significant percentage of the floating-point operations in this count are `sqrt` (8% for T42L18 and 6% for T170L18) and divide or reciprocal (4% for T42L18 and 3.5% for T170L18). This skews the computation rate calculations, making them lower than might be fair for these architectures. However, the count is applied uniformly across all the systems, and is a fair way of comparing performance across the different platforms for this particular study.

In Fig. 6, the performance of the AlphaSC-500 and the SP-375 are essentially identical for problem size T42L18. For T170L18, the SP-375 has slightly better performance, but it shares the same scaling behavior as the AlphaSC-500. So, despite the performance differences in the two systems described earlier, the aggregate performance for the *individually optimized* codes is very similar. This is an important point - each machine has its own performance quirks, and different tuning options were used on each system.

The performance of the AlphaSC-500 and SP-375 is quite good compared to the other systems included in the figures. The need to weight the operation count makes it difficult to evaluate system performance in more general terms, or even to compare between the different problem sizes. Given that T42L18 has an even higher percentage of `sqrt` and divides than T170L18, we would expect it to have a lower computational rate than T170L18. As the opposite is true, we assume that the higher computational rate for T42L18 is due to improved cache utilization as compared to T170L18.

Scalability appears to be quite good when solving problem size T170L18. However, as the T170L18 computational rates start lower than those for T42L18, scalability is also easier to maintain. The scalability is not as good for T42L18. On the SP-200, going from 128 to 256 nodes (256 to 512 processors) produces a slowdown. As the SP-200 shares the same switch and the same node memory subsystem as the SP-375, the implication is that 128 nodes will also be a maximum for this size problem on the SP-375. While the faster processor in the SP-375 lowers some of the local MPI overhead when compared to the SP-200, the SP-375 also has four processors per node contending for access to the node memory and the network, compared to two in the SP-200. The T42L18 scaling implications for the AlphaSC-500 are less clear. However, the AlphaSC-500 performance tracks that of the SP-375 so closely that we expect similar performance limitations for Compaq system.

Figure 7 breaks down these performance results into "total computation time", i.e., the sum over all processors of the time spent doing work found in a serial run, and "parallel overhead time", i.e., the sum over all processors of time spent in interprocessor communication, buffer copying in support of communication, idle time due to load imbalances, and performing redundant work. (The T3E-900 results are omitted, as they make the data for the AlphaSC-500 and SP-375 difficult to read.)

For T42L18, the serial computation and parallel overhead estimates are nearly identical for the AlphaSC-500 and the SP-375. There is some evidence that the parallel overhead for the AlphaSC-500 is growing faster than that for the SP-

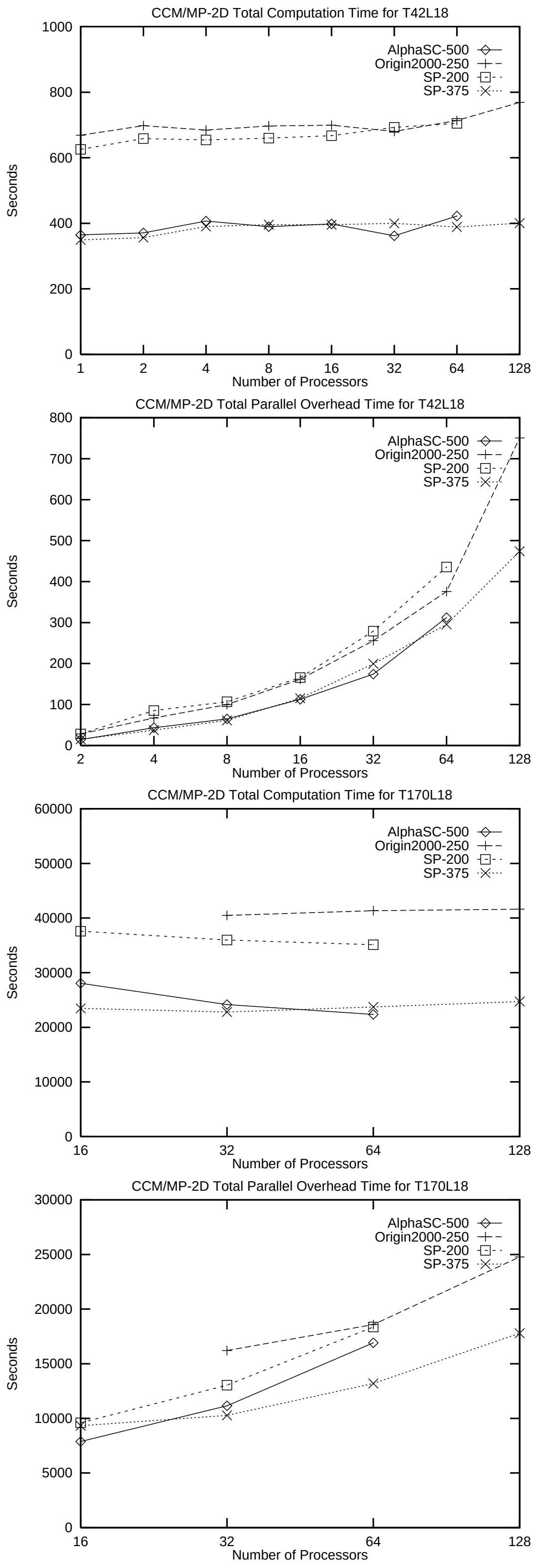


Figure 7: CCM/MP-2D performance analysis.

375 for larger numbers of processors. For T170L18, the AlphaSC-500 parallel overhead estimate is clearly growing much faster than that for the SP-375. But the serial computation time is also decreasing for the AlphaSC-500, indicating an improved computational rate.

5. CONCLUSIONS

In this paper, we studied performance aspects of the new IBM SP and Compaq AlphaServer SC systems installed at ORNL, using kernels and application codes relevant to the spectral atmospheric model CCM. From this initial evaluation, both the SP-375 and AlphaSC-500 provide improved performance over the SGI Origin 2000 and T3E-900 and earlier generations of the IBM SP currently used to run Department of Energy application codes. However, parallel algorithm tuning was important in achieving this performance. In particular, the MPI collective communication-based implementations of the parallel algorithms did not perform well.

Overall system performance of the AlphaServer SC and the SP are remarkably similar for the given code and problem sizes. From the analysis described in Fig. 7, the parallel overhead on the SP is smaller and growing slower than the parallel overhead on the AlphasServer SC. This may result in the SP having superior scaling for larger numbers of processors.

Additional nodes will be added to the ORNL SP system in the near future. This will allow us to further evaluate the scalability of the SP interconnect. The ORNL AlphaServer SC is also being upgraded, to a 256 processor system with faster processors (667MHz) and a larger L2 cache (8MB). Based on the results of this study, we expect the new AlphaServer SC system to outperform the SP-375 for up to 64 or 128 processors on this application code. The scalability of the Compaq system beyond 128 processors is an open question, however.

6. ACKNOWLEDGEMENTS

This research was performed using a 64 processor Compaq AlphaServer SC and two different generations of the IBM SP, a 124 processor Winterhawk I system and a 248 processor Winterhawk II system, all at Oak Ridge National Laboratory. Performance data for the SGI Origin was collected on a 128 processor system at the Advanced Computing Laboratory at Los Alamos National Laboratory. Performance data for the T3E-900 was collected on the 644 processor system at the National Energy Research Scientific Computing Center. The 512 processor IBM SP at NERSC was also used to verify and supplement the Winterhawk I results collected at ORNL.

7. REFERENCES

[1] 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.

[2] 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.

[3] J. B. Drake, S. Hammond, R. James, and P. H. Worley, *Performance Tuning and Evaluation of a Parallel Community Climate Model*, in Proceedings of the ACM/IEEE Conference on High Performance Networking and Computing (SC99), Los Alamitos, CA, November 1999, IEEE Computer Society Press.

[4] 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.

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

[6] 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.

[7] 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.

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

[9] D. L. Williamson and P. J. Rasch, *Two-dimensional semi-Lagrangian transport with shape-preserving interpolation*, Mon. Wea. Rev., 117 (1989), pp. 102–129.

[10] 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.