ALGORITHM COMPARISON AND BENCHMARKING USING A PARALLEL SPECTRAL TRANSFORM SHALLOW WATER MODEL*

Patrick H. Worley

Mathematical Sciences Section
Oak Ridge National Laboratory
P.O. Box 2008, Bldg. 6012
Oak Ridge, TN 37831-6367, U.S.A.

and

Ian T. Foster and Brian Toonen

Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439, U.S.A.

1. Introduction

In recent years, a number of computer vendors have produced supercomputers based on a massively parallel processing (MPP) architecture. These computers have been shown to be competitive in performance with conventional vector supercomputers for some applications. As spectral weather and climate models are heavy users of vector supercomputers, it is interesting to determine how these models perform on MPPs, and which MPPs are best suited to the execution of spectral models.

The benchmarking of MPPs is complicated by the fact that different algorithms may be more efficient on different architectures. Hence, a comprehensive benchmarking effort must answer two related questions: which algorithm is most efficient on each computer and how do the most efficient algorithms compare on different computers. In general, these are difficult questions to answer because of the high cost associated with implementing and evaluating a range of different parallel algorithms on each MPP platform.

---

*This work was supported by the Atmospheric and Climate Research Division, Office of Energy Research, U.S. Department of Energy under Contract DE-AC05-840R21400 with Martin Marietta Energy Systems and under Contract W-31-109-Eng-38 with the University of Chicago.
DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.
In a recent study\(^2\), we developed a testbed code called PSTSWM that incorporated a wide range of parallel spectral transform algorithms. Studies with this testbed confirm that the performance of different algorithms can vary significantly from computer to computer, and that there is no single algorithm that is optimal on all platforms and for all problem sizes. Availability of this testbed makes it feasible to perform a comprehensive and fair benchmarking exercise of MPP platforms for spectral transform codes. We report the preliminary results of this exercise in this paper, presenting benchmark results for the Intel Paragon, the Cray T3D, and the IBM SP1.

2. Method

We first briefly describe the structure of the PSTSWM testbed code, the experiments that were performed during the benchmarking exercise, and the computers on which benchmarks were performed.

2.1 The PSTSWM Testbed

A number of researchers have investigated parallel algorithms for the spectral transform algorithm used in atmospheric circulation models. A variety of different parallel algorithms have been proposed, but until recently no detailed quantitative comparisons of the different approaches had been attempted. (However, some qualitative comparisons have been reported\(^{1,4}\).) To permit a fair comparison of the suitability of the various algorithms, we have incorporated them into a single testbed code called PSTSWM (for parallel spectral transform shallow water model). PSTSWM is a message-passing parallel implementation of the sequential Fortran code STSWM\(^3\). STSWM uses the spectral transform method to solve the nonlinear shallow water equations on a rotating sphere; its data structures and implementation are based directly on equivalent structures and algorithms in CCM2, the Community Climate Model developed at the National Center for Atmospheric Research.

PSTSWM differs from STSWM in one major respect: vertical levels have been added to permit a fair evaluation of parallel algorithms based on matrix transposes. This is necessary because in a one-layer model, a transpose algorithm reduces to a one-dimensional decomposition of each grid and hence can utilize only a small number of processors. The addition of vertical levels also has the advantage of modeling more accurately the granularity of the dynamics computation in atmospheric models. In all other respects we have changed the algorithmic aspects of STSWM as little as possible. In particular, we did not change loop and array index ordering, and the serial performance of the code is consistent with that demonstrated by CCM2.
PSTSWM is structured so that a variety of different algorithms can be selected by runtime parameters\(^2\). Both the fast Fourier transform (FFT) and the Legendre transform (LT) that make up the spectral transform can be computed using several different distributed algorithms (in which data structures are decomposed over processors and communication is performed as required by the FFT or LT) and transpose algorithms (in which data is explicitly reorganized prior to an FFT or LT to avoid a need for communication during the transform). Runtime parameters also select from among a range of variants of each of these major algorithms. All parallel algorithms were carefully implemented, eliminating unnecessary buffer copying and exploiting our knowledge of the context in which they are called. At the present time, this allows us to achieve better performance than can be achieved by calling available vendor-supplied routines. Hence, it provides a fairer test of the parallel algorithms.

2.2 Experimental Technique

We performed experiments for a range of horizontal and vertical resolutions, as summarized in Table 1. (Horizontal resolution is specified in terms of both spectral truncation and physical grid size.) This range of resolutions was considered necessary firstly because there is little agreement on standard resolutions and secondly because the performance of different parallel algorithms can vary significantly depending on the number of vertical levels. The highest vertical resolutions in the T42 and T85 models are intended to be representative of resolutions used in stratospheric models. In the experiments reported in this paper, we restrict horizontal resolutions to be powers of two; in later studies, we will expand the set of experiments to include other common resolutions such as T63 and T213.

All experiments involved a five day simulation using the performance benchmark proposed by Williamson et al.\(^5\): global steady state nonlinear zonal geostrophic flow. The number of timesteps (Table 1) assumes a Courant number of 0.5. In practice, a timestep almost twice as large could be used without losing stability, halving the execution time, but at the cost of some degradation in the solution accuracy for the larger model resolutions. We report raw execution times, Gflops rates, and Mflops achieved per processor. The computation rates are derived from floating point operation counts obtained using the hardware performance monitor on a Cray Y/MP.

Experiments were performed in two stages. In the first stage, a set of tuning experiments were performed to determine the most efficient algorithms and algorithmic variants on each computer and at each resolution. These
Table 1: Problem Sizes Considered in Empirical Studies, Floating Point Operations per Vertical Level, and Number of Timesteps for a 5-Day Simulation

<table>
<thead>
<tr>
<th>Horizontal</th>
<th>Grid</th>
<th>Vertical Levels</th>
<th>Flops/level/step</th>
<th>Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>T42</td>
<td>64 × 128</td>
<td>L16, L18, L44</td>
<td>4129859</td>
<td>222</td>
</tr>
<tr>
<td>T85</td>
<td>128 × 256</td>
<td>L16, L18, L24, L36, L60</td>
<td>24235477</td>
<td>446</td>
</tr>
<tr>
<td>T170</td>
<td>256 × 512</td>
<td>L18, L24, L32, L36, L48</td>
<td>153014243</td>
<td>891</td>
</tr>
</tbody>
</table>

experiments were very detailed, involving 3000-5000 separate measurements on each computer. Based on the results of these experiments, we selected an optimal algorithm for each problem size, number of processors, and parallel platform. In the second stage, we measured execution times on each computer and for each resolution listed in Table 1, using the optimal algorithms identified in the first stage.

2.3 Target Computers

We performed experiments on the three parallel computer systems listed in Table 2. These systems have similar architectures and programming models, but vary considerably in their communication and computational capabilities. Our values for message startup time ($t_s$) and per-byte transfer time ($t_b$) are based on the minimum observed times for swapping data between two processors using the swap routines in PSTSWM, and represent achievable, although not necessarily typical, values. Note that the startup times include the additional subroutine overhead and logic needed to implement the swap semantics used in the code. The linear ($t_s, t_b$) parameterization of communication costs is surprisingly good for the Paragon and T3D when using the optimal communication protocols, but is only a crude approximation for the SP1. The MBytes/second measure is bidirectional bandwidth, and so is approximately twice $1/t_b$. The computational rate (Mflops/sec) is the maximum observed by running PSTSWM on a single node for a variety of problem resolutions, and so is an achieved peak rate rather than a theoretical peak rate.

The Paragon used in these studies has two processors per node. For all measurements, the second processor was used as a message coprocessor, and $P$ in Tab. 2 and the X axis for all figures refer to the number of compute processors (or nodes). The Paragon used the SUNMOS operating system de-
Table 2: Parallel Computers Used in Empirical Studies, Characterized by Operating System Version, Microprocessor, Interconnection Network, Maximum Machine Size Used in Experiments, Message Startup Cost \( (t_a) \), Per-Byte Transfer Cost \( (t_b) \), and Per-Processor MFlops/Sec

<table>
<thead>
<tr>
<th>Name</th>
<th>OS</th>
<th>Processor</th>
<th>Network</th>
<th>( P )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paragon</td>
<td>SUNMOS 1.6.1</td>
<td>i860SP</td>
<td>16 × 64 mesh</td>
<td>1024</td>
</tr>
<tr>
<td>SP1</td>
<td>AIX + MPI-f</td>
<td>Power 1</td>
<td>multistage crossbar</td>
<td>128</td>
</tr>
<tr>
<td>T3D</td>
<td>MAX 1.1.0.5</td>
<td>Alpha</td>
<td>16 × 4 × 4 torus</td>
<td>256</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>( t_a ) (( \mu \text{sec} ))</th>
<th>( t_b ) (( \mu \text{sec} ))</th>
<th>MB/sec</th>
<th>MFlops/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paragon</td>
<td>72</td>
<td>0.007</td>
<td>282</td>
<td>11.6</td>
</tr>
<tr>
<td>SP1</td>
<td>100</td>
<td>0.059</td>
<td>36</td>
<td>25.6</td>
</tr>
<tr>
<td>T3D</td>
<td>18</td>
<td>0.012</td>
<td>168</td>
<td>18.2</td>
</tr>
</tbody>
</table>

Developed at Sandia National Laboratories and the University of New Mexico, which currently provides better communication performance than the standard Intel operating system. The SP1 used in these studies is a nonstandard system incorporating the faster SP2 interconnection network. Interprocessor communication was performed using MPI-f version 1.3.3., an experimental implementation of the MPI message-passing standard developed and made available to us by Hubertus Franke of IBM Yorktown. Interprocessor communication routines for the T3D were implemented using the Shared Memory Access Library, which supports reading and writing remote (nonlocal) memory locations. Experiments were performed using single-precision arithmetic (32-bit floating point values) except on the T3D, where only double precision (64-bit) is supported in Fortran. In future work, we will also evaluate the performance impact of using double precision arithmetic on other machines. On the SP, the computation rate should stay about the same, while communication costs will of course increase; on the Paragon, the computation rate also decreases by approximately 25%.

3. Results

The results of the experiments are summarized in Figures 1–9. These figures show results on the different machines at T42, T85, and T170 resolution, expressed in terms of both execution time for a 5-day forecast, execution rate in Gflops/sec, and megaflops achieved per processor, all as a function
of processor count. Notice the use of a log scale in the X axis. In the figure keys, P represents Paragon, T represents T3D, and S1 represents SP1. On the Paragon, data were obtained on 128, 256, 512, and 1024 processors. On the T3D, data were obtained on 64, 128, and 256 processors. On the SP, data were obtained only on 128 processors. Some problem sizes did not fit on a small number of processors on the T3D, and hence are missing. SP1 data were obtained only at T42 and T85 resolution.

4. Discussion

Figures 1–9 present a large amount of data from which one can derive many interesting conclusions about both the spectral transform method and the performance of the T3D, Paragon, and SP. For example:

- In many cases, performance is significantly better when the number of vertical levels is a power of 2. This phenomenon is to some extent an artifact of the fact that the number of processors and the horizontal grid dimensions are both powers of 2.

- It appears that the T3D is roughly 75 percent faster than the Paragon
Figure 2: Execution times for 5-day forecast at T85 resolution.

Figure 3: Execution times for 5-day forecast at T170 resolution.
Figure 4: Execution rate for 5-day forecast at T42 resolution.

Figure 5: Execution rate for 5-day forecast at T85 resolution.
Figure 6: Execution rate for 5-day forecast at T170 resolution.

Figure 7: Processor performance for 5-day forecast at T42 resolution.
Figure 8: Processor performance for 5-day forecast at T85 resolution.

Figure 9: Processor performance for 5-day forecast at T170 resolution.
with the same number of processors across all problem sizes, with the SP1 performance falling between these two for the 128 processor cases.

- Paragon performance is roughly comparable to T3D performance on one-half as many processors, across a wide range of problem sizes.

The relative performance on the three machines is affected by the fact that the T3D is running with 64 bit precision, while the others are using 32 bit precision. This is a fair comparison for small problem resolutions, where the extra precision is not needed. The higher precision may be needed for larger resolution problems, which will increase the gap between the T3D and the other machines. For example, double precision experiments on the Paragon show a 30-40% increase in execution time over single precision timings across all problem sizes and numbers of processors. Note that we have not attempted to compare the price-performance (performance per unit of capital investment) of the different machines, due to the difficulty of obtaining accurate price data, and its dependence on nontechnical factors. However, this information must clearly be taken into account when interpreting the results of this study.

The absolute performance is affected by the coding style of the program as well as by the parallel algorithms and the communication costs, the latter two over which we have attempted to tune. PSTSWM was designed to model the coding style of PCCM2, the message-passing parallel implementation of CCM2, and is an accurate vehicle for doing so. Some of the computational kernels of PSTSWM (and PCCM2) have been restructured to run more efficiently on the cache-based RISC multiprocessors around which the target MPPs are built, with particular emphasis on accessing memory in a linear fashion. But the ratio of floating point operations to memory accesses is only 1.3 on the T3D, indicating that there is little reuse of data in the cache. Exploiting optimized FFT library routines and more aggressive code restructuring to allow, for example, the use of level 3 BLAS, would further improve the performance of these codes. We are hesitant to make code modifications to PSTSWM that would be difficult to emulate in PCCM2, but one advantage of a code like PSTSWM is that it provides us with a testbed to experiment with various optimization techniques before making changes in the production code.

5. Conclusions

The results of this study indicate that massively parallel computers such as the Paragon, T3D, and SP are indeed capable of multiGigaflips perfor-
mance, even on communication-intensive applications such as the spectral transform method. However, performance depends on a wide variety of factors, including parallel algorithm, communication protocols, and coding style, and even outwardly similar machines are likely to have different optimal tuning parameters. The potential performance improvements from tuning make it worthwhile, especially if the code has been written to allow some of the tuning parameters to be set at runtime. Runtime tuning parameters support both (performance) portability and the empirical determination of the optimal parameters.

In future work, we will run our benchmarks on larger SP and T3D configurations. We will also evaluate the IBM SP2 architecture, which has a processor twice as fast as the SP1 used in these studies, and the Paragon MP architecture, which has two compute processors per node. We also plan to evaluate performance on other problem sizes, in particular T63 and T213. In a related study, we are evaluating the feasibility of calibrating analytic performance models and then using these performance models to predict performance on future architectures.

Acknowledgements

We are grateful to members of the CHAMMP Interagency Organization for Numerical Simulation, a collaboration involving Argonne National Laboratory, the National Center for Atmospheric Research, and Oak Ridge National Laboratory, for sharing codes and results, to Hubertus Franke of IBM for providing his MPI-f library on the SP, to the SUNMOS development team for help in getting PSTSWM running under SUNMOS on the Intel Paragon, and to James Tuccillo of Cray Research for facilitating the Cray T3D experiments.

This research was performed using the Intel Paragon system at Oak Ridge National Laboratory, the Intel Paragon system at Sandia National Laboratories, a Cray T3D at Cray Research, and the IBM SP system at Argonne National Laboratory.

References


