Some Performance Comparisons for an Ocean Model on the SGI Origin 2000 and the HP V-class 2500

Benny N. Cheng
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA, U.S.A.

Abstract ***A state-of-the-art ocean model developed originally at MIT is currently being tested and evaluated for assimilating satellite data here at NASA-JPL. We report on some performance results in running this model on both the SGI Origin 2000 and the HP 2500 V-class parallel supercomputers.***

Keywords: SGI Origin, HP V-class SCA,ocean modeling,MPI, parallel programming

1 Introduction

The MIT Ocean General Circulation Model (OGCM) is a general ocean model based on the incompressible Navier-Stokes equations in fluid dynamics. It is based on a consistent set of hydrostatic, quasi-hydrostatic, and non-hydrostatic equations, with accurate simulations verified in experiments on scales of a few centimeters to tens of kilometers of the ocean. This model differs from other existing models in the following areas: 1) it is discretised using finite volume techniques, permitting a novel treatment of topography, 2) uses height as a vertical coordinate, 3) has a free surface component as oppose to rigid lid model, 4) need not make the hydrostatic approximation, 5) has tangent-linear and adjoint counterparts, 6) was developed and targeted to parallel computers, and finally 7) with minimal modifications, it can be used to study the atmosphere as well. [1]

At the Jet Propulsion Laboratory, we intend to use this model as a platform for the assimilation of the growing remote sensing oceanic data collected from earth orbiting satellites, in particular, the TOPEX/Poseidon satellite,which has been in operation for more than 8 years and still operating, and it's upcoming followon, JASON-1. Techniques for assimilating the data include Kalman filtering and the adjoint, both of which will be used provide near real-time analysis of the state of the world's ocean, either regionally or globally. Further details about this project can be found in [2].

2 Computer Platforms

Our computing resources for running this model include an SGI Origin 2000 with 128 R12000 chips, each operating at 300 Mhz. Reported peak speed is at 76 Gflops, with a total of 64 Gb of memory and 6 Tb of disk storage. The processors are arranged in a hypercube-like topology, with Craylink connections between nodes at a bandwidth of about 600 Mb/s. The MIPS chip is reportedly capable of 2 floating point instructions per cycle, with onchip L1 instruction and data cache size of 32 Kb, plus an external unified secondary L2 cache size of 8 Mb [3]. The installed OS is the 64-bit IRIX version 6.5.

The other platform we tested on is a Hewlett-Packard V2500 SCA (Scalable Computing Architecture) with 64 PA-RISC (Precision Architecture RISC) 8500 chips, each operating at 440 Mhz. This chip is reportedly capable of 4 Flops, hence a peak speed of about 112 Gflops. It also has 1 Mb L1 data cache and 0.5

Mb L1 instruction cache, both residing on the chip [4], but no L2 cache. Total memory capacity is identical to the Origin at 64 Gb, with about 400 Mb of disk space. The processors are connected via a toroid-like interconnect, with SCA Hyperlink connections between nodes at a peak bandwidth of about 3.8 Gb/s [5]. The 64-bit OS used is the HP-UX B.11.0.

Theoretically, the HP V2500 is a faster machine, by a factor of about 1.5. Plus the size of the primary cache is several times that of the MIPS chip. Therefore one would expect that there would be performance improvements running any code on the HP machine compared to the Origin 2000. However, we found this was not the case with our model.

3 Performance Comparisons

The MIT OGCM is a fortran F77 code that can be compiled either as a threaded shared-memory code (not OpenMP) or as a message passing MPI code. Extensive tests done on the Origin with the Multiprocessing (MP) threads show little to be gained by switching to shared-memory code, hence the MPI version of the code was selected. This has a lot to do with memory being unsatisfactorily distributed among the processors, resulting in highly uneven performance, especially with large number of concurrently running users. With the MPI code, the performance numbers are generally more stable, for reasons that are probably related to the way MPI was implemented on the machine.

On both machines, we strive to choose to the best of our knowledge the highest level of optimization allowed by the compilers. On the Origin, the following compiler optimization flags were utilized:

```
-O3 -n32 -mips4
-OPT:Olimit=0:roundoff=3:div_split=ON
```

The `-O3` flag indicates aggressive optimization by the compiler, "generally seeking the highest-quality generated code even if it requires extensive compile time", according to the man pages. `-n32` and `-mips4` generates an n32 object with the full MIPS IV instruction set. For the additional OPT flags, `Olimit=0` means avoiding any automatic cutoff for the size of routines that are optimized. The default for `-O3` is 3000 lines. `roundoff=3` enables all mathematically valid roundoff techniques to be applied, at the cost of losing some precision. `div_split=ON` splits division of x/y as x*(1/y), since multiplication is faster and cheaper than division.

On the V2500, the following optimization flags were applied:

```
+O3 +Ofastaccess
```

`+O3` indicates full optimization across all subprograms within the source file, including subprogram cloning and inlining. `+Ofastaccess` enables fast access to global data.

The code was then ran on 64 processors on both the Origin and the V2500. The results show that the runs on the Origin finish on the average two times faster than on the V2500. The code runs at a speed of about 8 Gflops on the Origin, while clocking in at only 4.2 Gflops on the V2500.

The primary cause of why the code runs slower on a supposedly faster machine wasn't too difficult to locate. Tests with sequential non-MPI code show that V2500 is indeed a faster machine. So we decided to look a little closer at the MPI calls themselves. Utilizing the `mpiview` instrumenting tool on the V2500, Figure 1 shows the total time for all 64 CPUs spent by the code doing each of the MPI calls. It is clear from the figure that a large amount of time was spent doing MPI_Recv calls, about 12910/64 = 201.8 secs execution time on the average. Though nothing like the `mpiview` GUI tool was available on the Origin, a corresponding instrumentation (Figure 2) on the Origin using the Speedshop `ssrun` tool shows that far less time was spent by the Origin on the MPI_Recv calls compared to the V2500, about 6546/64 = 102.3 secs on the average. Percentagewise, about 26% of the wallclock

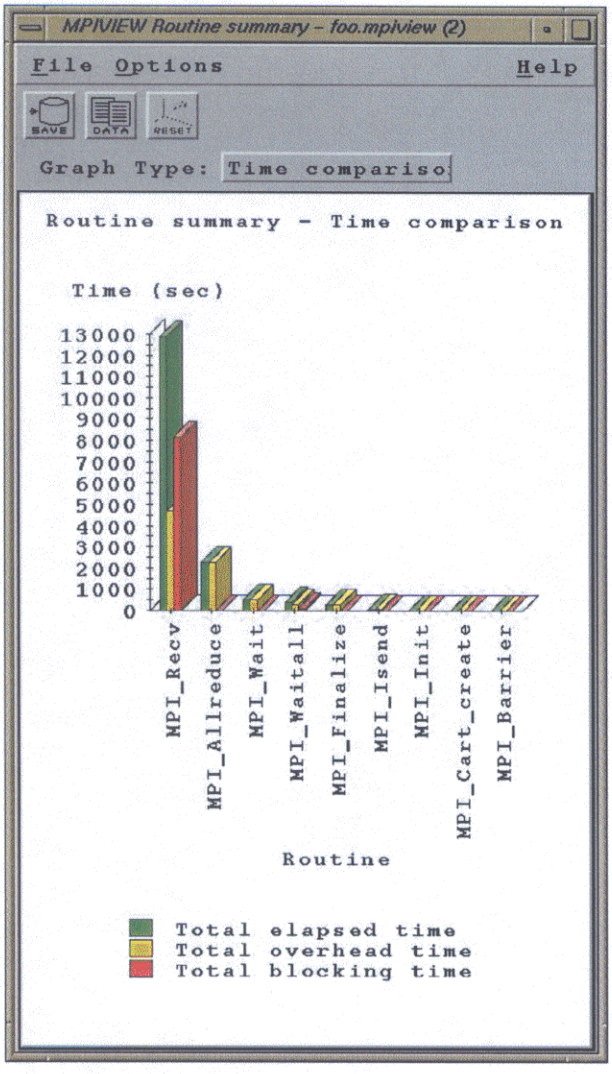

Figure 1: V2500 MPI timings

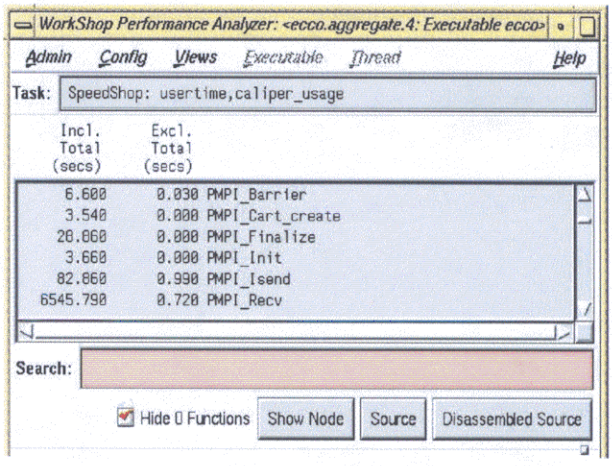

Figure 2: Origin MPI timings

time was spent on MPI_Recv by both V2500 and the Origin 2000.

We are therefore forced to conclude that the observe differences in wall clock time when running the model are mostly due to the influences of the MPI library, which very likely was not optimized on the V2500. There maybe other factors involving hardware, but we believed this to be the major cause.

4 Conclusions

At JPL, we are in constant search for the best platform to run our complex computer models, such as an OGCM. We have on one hand an SGI Origin 2000, which is a symmetric multiprocessing (SMP) system in a hypercube-like interconnection powered by the MIPS chip operating at 300 Mhz. It is scalable, with a cache-coherent non-uniform memory access (ccNUMA) architecture. On the other hand, we currently also have access to an HP V2500 SCA SMP system powered by PA-RISC chips at 440 Mhz. It is also scalable, with ccNUMA architecture in a toroid-like interconnection. The best optimization flags to our knowledge are applied to the compiler when compiling our OGCM code for each of the above machines. Running the same model on both machines, the results were unexpected, with the supposedly faster V2500 machine actually performing at about half the speed of the Origin. Analysis with performance and profiling tools show that the main cause of the timing differences were due to the non-optimal library calls being used by the compiler, in particular the MPI library, as demonstrated by our OGCM code.

Acknowledgement

This work is performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Agency. Reference herein to any

specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology. We highly appreciate the comments and contributions of Mahesh Rajan of HP and the Center for Advance Computing and Research at Caltech in the preparation of this paper.

References

[1] The MIT Climate Modeling Initiative. **http://geoid.mit.edu/climatemodel/oceanmodel.htm**

[2] The ECCO Project. **http://ecco.jpl.nasa.gov**

[3] SGI 2000 Series Applications Programming & Optimization, Parts I and II. *SGI Global Education Courseware*, February 2000.

[4] HP Business and Technical Computing. **http://www.hp.com/computing/framed/technology/micropro/pa-8500/**

[5] The HP SCA Hyperlink. **http://www.cacr.caltech.edu/~mrajan/ppt_pres/v2500_arch/sld009.htm**