Climate Ocean Modeling on a Beowulf Class System

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

Ping Wang
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA, U.S.A.

Yi Chao
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA, U.S.A.

Michael. Bondarenko
User Tech Associates, Inc.
Pasadena, CA, U.S.A.

Abstract ***With the growing power and shrinking cost of personal computers, the availability of fast ethernet interconnections, and public domain software packages, it is now possible to combine them to build desktop parallel computers (named Beowulf or PC clusters) at a fraction of what it would cost to buy systems of comparable power from supercomputer companies. This led us to build and assemble our own system, specifically for climate ocean modeling. In this article, we present our experience with such a system, discuss its network performance, and provide some performance comparison data with both HP SPP2000 and Cray T3E for an ocean model used in present-day oceanographic research.***

Keywords: Beowulf, cluster computing, ocean modeling, parallel programming

1 Introduction

Beowulf class systems consisting of clusters of off-the-shelf PC's are becoming a regular fixtures in research and industrial computing. Traditional supercomputers are refrigerator-size cabinets that contain thousands of microprocessors. These supercomputers are built with specialized components and software that can be operated only by expert technicians and programmers. These machines usually have their own cooling systems, require large amounts of electricity and cost typically more than $1,000,000. The Beowulf approach represents a new business model for acquiring computational capabilities, particularly for small to medium sized applications. It complements rather than competes with the more conventional vendor-centric systems-supplier approach.

At Jet Propulsion Laboratory (JPL), the ocean modeling group recently decided to build its own Beowulf system, the first one to be built inhouse, mainly to run our increasingly complex ocean models. This system consists of 13 Intel Celeron Pentium II PC's running at 300 mhz, interconnected by a 100 mbs fast ethernet network, with a total price of about $26K. The popular Linux was chosen to be the operating system, and being publicly available, enabled much of the needed supporting software to be downloaded from the internet free of charge. The communicating programming model of choice was the message passing interface or MPI, due to its portability. The ocean model to be tested will be based on the Parallel Ocean Program (POP), which has been extensively used at JPL on the Cray T3D/E and HP SPP2000 parallel computers [9]. It is anticipated that we can perform small to medium-size ocean modeling applications on such a Beowulf cluster, which is complemented by the commercial massively parallel computers for large and extremely large sized (killer) applications. The objective of this arcticle is to assess the performance of our network and present some performance comparison data with both HP SPP2000 and Cray T3E using application runs from our ocean model programs.

2 Setting up the Beowulf Cluster

Documentation for setting up a Beowulf cluster are widely available from various sources, and our setup mainly followed that in the "How to Build a Beowulf, A Tutorial" [1] by P.Angelino et al. All machine parts were ordered from local commercial vendors and brought in to JPL for assembly. Most of the parts ordered were identical to that specified in [1], with the exception of ours having faster Pentium chip (PII Celeron), more RAM memory (168mb), larger disks (4.3 Gb), and different network cards (3Com 905B-TX Boomerang) and switch (24 port SuperstackII 3300). We also have the benefit of a newer version of the Linux operating system (Redhat 5.0), which was purchased with the Extreme Linux CDROM from Redhat Systems for less than $30. Installation of the operating system on the main disk went as planned, the machine booted up as expected. The first major problem encountered was the discovery that the 3C905B network card drivers were not available on the CDROM distribution. A quick search through the internet led to Donald Becker's site at Goddard Space Flight Center, who recently made available his latest version (v0.99H) of the network driver that does support this new card. It was now a routine matter to compile this new driver, and to install this as a module to the kernel. A series of tweaks followed in order maximize the network performance, and we found that in order for the card to detect a 100mbs network link, the following must be modified in the 3c59x driver code:

> {"100baseTX", Media_Lnk, 0x02, XCVR_100baseFx, (40*HZ)/10.}

where 14 was changed to 40,which allowed the card a little more time to detect the default link beat. The next major problem was how to clone the first working system to the 12 others, without having to remove the hard drive and do a `dd` each time, as done in [1]. A few hours of research (again through the internet) provided us with the idea of using the Trinux (a diskless version of Linux, residing completely in RAM space) as a starting point for the cloning procedure. The basic step of this process starts with installation of the Trinux on the new machine from a floppy drive, which included the drivers for the SCSI disk and network card. After the partitions were setup properly on the new machine, the network driver is loaded, and a TCP connection is started with the machine to be cloned. The entire filesystem was then copied with the `cpio` command. This method is definitely faster than using `dd` with large disks, as cpio is quicker, and has the major advantage of not having to disconnect the hard drives. With all 13 systems up, connected, and running, the last major task was to setup the automount file sharing system. This proves to be routine, as the documentation provided with the amd software was sufficient.

The Extreme Linux CDROM comes with both LAM and MPICH version of the MPI parallel programming model, and we decided to go with the MPICH version, due to our previous experience with this software. We downloaded a later version (1.2) of MPICH from the Argonne National Labs website, and installed the program. An f90 compiler was also needed for compiling our ocean parallel programs, and the Absoft f90 compiler was our choice.

3 Network Performance

We decided to first examine the network performance of this machine, using a similar tests as done in [2]. The Hyglac machine (JPL's first Beowulf, built at Caltech) network interface cards (NICs) are D-link cards with Tulip chipset, while we mentioned before that ours are 3C905B-TX Boomerangs. Comparing the

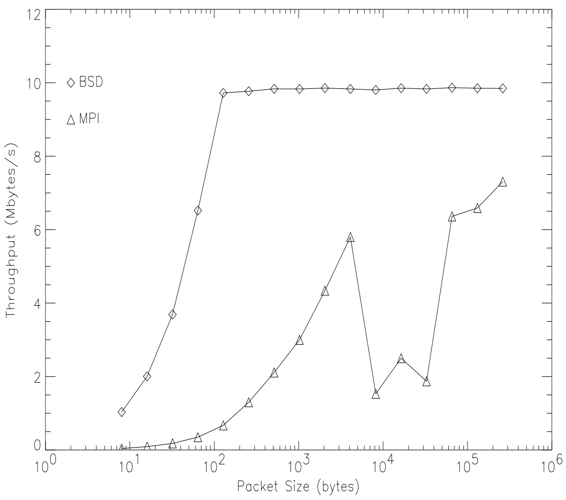

Figure 1: Sockets vs. MPI throughput performance

results with those obtained in [2] with the JPL Hyglac beowulf cluster (Figure 1), the netperf program for BSD socket tests showed that our network has a higher throughput for packet size of less than 512 bytes, and reaching close to its peak rate by the time the packet size hit 128 bytes. The peak rate of 9.87 Mbytes/s however was significantly less than the 11.8 Mbytes/s observed with Hyglac network. For MPI send and receive performance, ours show slightly better rates for up to 4Kbytes packet sizes, and at 64Kbytes and 128Kbytes packets. There is a rather precipituous drop in performance between 8Kbytes and 32Kbytes packet sizes, and then moving back up to a peak rate of 7.31 Mbytes/s for 256Kbytes packet, though the range where this rate drop occurred at a different location for the Hyglac network (between 32Kbytes and 128Kbytes), with maximum recorded rate of 8.3 Mbytes/s. Possible causes of this rate drop are socket buffer size and ethernet segment size [2], though the exact cause has yet to be determined. However, it is clear at this point that it is dependent on the type of NIC used. In summary, our machine will perform better than Hyglac on programs communicating with packets less than 1 Kbytes in size.

4 Description of the Ocean Model

The Ocean General Circulation Model (OGCM) is based on the Parallel Ocean Program (POP) developed at Los Alamos National Laboratory [3]. This ocean model evolved from the Bryan-Cox 3-dimensional primitive equations ocean model [4,5], developed at NOAA Geophysical Fluid Dynamics Laboratory (GFDL), and later known as the Semtner and Chervin model or the Modular Ocean Model (MOM) [6]. Currently, there are hundreds of users within the so-called Bryan-Cox ocean model family, making it the dominant OGCM code in the climate research community. Furthermore, this model has been subjected to a high degree of optimization on parallel machines over the last few years [7][8].

The OGCM solves the 3-dimensional primitive equations with the finite difference technique. The equations are separated into barotropic (the vertical mean) and baroclinic (departures from the vertical mean) components. The baroclinic component is 3-dimensional, and uses explicit leapfrog time stepping. It parallelizes very well on massively parallel computers. The barotropic component is 2-dimensional, and solved implicitly. It differs from the original Bryan-Cox formulation in that it removes the rigid-lid approximation and treats the sea surface height as a prognostic variable (i.e., free-surface). The free-surface model is superior to the rigid-lid model because it provides more accurate solution to the governing equations. More importantly, the free-surface model tremendously reduces the global communication otherwise required by the rigid-lid model.

5 Results on the Beowulf Cluster

At the core of our ocean model is the parallel conjugate gradient solver code used to solve the barotropic component of the model. We are interested in the performance of this solver code ,which comprises much of the computation and communication routines of the model, on different parallel machines. Each of the machine that we compared to runs it's own version of the MPI message passing architecture. The model grid size chosen for testing is a 2 degree x 1 degree global ocean model with 180 x 180 horizontal grid points and 20 vertical levels. which is the largest size that can fit in our Cray T3E memory for a two PE run. The POP 2-dimensional solver code uses a 9 point stencil scheme with diagonal preconditioning. The pperf package,which takes advantage of the special Model Specific Register (MSR) of the Pentium processor, is used to obtain accurate time and floating point operations (FLOP) count for each iteration of the solver. It usually takes several iterations for the solver to complete one timestep of a model run, i.e. for the solution to converge. Performance speed is defined as

$$\text{Speed} = \frac{\text{FLOP per timestep}}{\text{execution time per timestep}}$$

and averaging this over the total number of timesteps. To examine the differences in the flop rate, we also looked at the ratio of computation to communication for the current problem grid size, shown below in column 3. Single node performance results for the solver running on two processors are as follows:

Machine	Speed (Mflop/s)	Ratio
Beowulf	11.9	3.25 : 1
Cray T3E	22.7	3.7 : 1
Exemplar	39.1	0.57 : 1

Table 1: Solver performance

The HP Exemplar is an SPP2000 machine powered by the PA 8000 RISC chip running HPUX, with a peak flop rate of 720 Mflop/s. Its processors are connected via a toroidal interconnect, with a cache-coherent, nonuniform memory-access (ccNUMA) architecture. On the other hand, the T3E uses the DEC Alpha chip and runs on UNICOSMK, with a peak speed of 600 Mflop/s. The Alphas are connected by a high-bandwidth, low-latency bidirectional 3-D torus system interconnect network. The best optimization flags available are applied to the compiler for each of the above machines. Because of the communication overheads, the net flop rate given above is lower than the actual flop rate in accordance with the amount of time spent doing the communications.

Given that the computation to communication ratio is about the same for the Beowulf and the T3E, the Beowulf flop rate is close to what was expected with its peak flop rate of about 300 Mflop/s, half that of the T3E. On the other hand, It is notable that on the Exemplar, the model spents more of its time on communication relative to the other two machines, ,with the faster performance due to the Exemplar's coherent memory caches (at least within a hypernode). This explains why the Exemplar flop rate is so much higher than expected from its peak flop rate, which is only about 2.4 times faster than the Beowulf peak flop rate. We also looked at the solver speedup measurements for the above grid size, shown in Figure 2, using the above two PE run as the baseline.

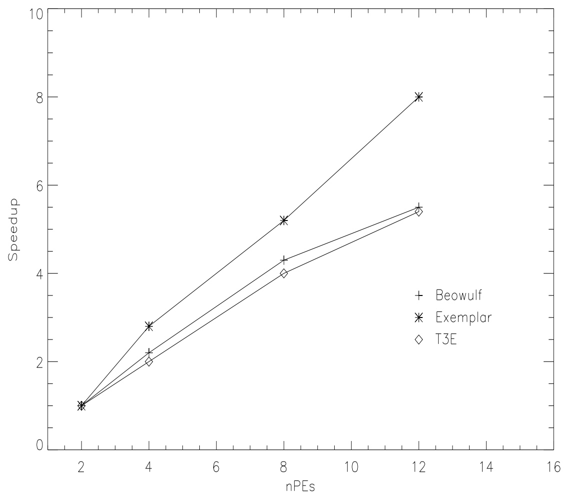

Figure 2: Solver speedup comparison curves

From the figure, we see that the Beowulf speedup is comparable to that of the T3E. The Exemplar exhibits an interesting superlinear speedup with the number of processors, which we attribute to the intrahypernode memory caching in communicating data. Overall, we see that the Beowulf cluster performs favorably in comparison, and cost about 10 times less per node than each of other two machines.

Finally, to convince ourselves and others that an actual model run is feasible, we setup an experiment with the POP model, using realistic topography and forcing the model with real ocean wind, (from the European Centre for Medium Range Weather Forecast (ECMWF)), salinity, and temperature (Levitus) data. The model domain ranges from 100E to 130E and 0 to 30N (closed wall on all four sides), with a resolution of 1/3 degree x 1/3 degree ,and 20 vertical levels. The sea level output at the end of a 120-day run is shown in Figure 3.

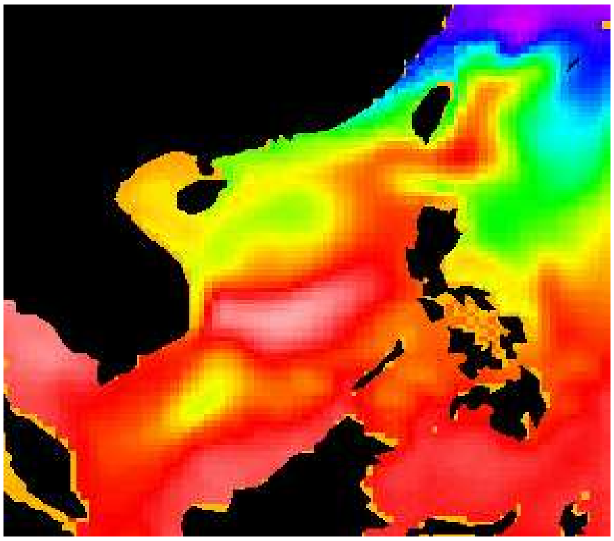

Figure 3: Sea level output at end of 120-day model run, with color scale ranging from purple (lowest) to pink (highest).

6 Conclusions

Beowulf class PC clusters are well suited for ocean modeling applications, especially for small to medium sized problems. With the current trends in PC pricing and CPU performance, the Beowulf computing paradigm seem destined only to grow in suitability. The attractive price-to-performance ratio means such machines are likely to be around for research and many other non time-critical applications. Another major advantage in favor of such a cluster is the ability to use it as a dedicated machine without sharing computing resources with many users, as is currently the case with large expensive machines. Our PC cluster is definitely not the best in quality that can be assembled, but certainly qualifies as a one of the least expensive (in the Los Angeles area), if not the least expensive one in terms of performance. It is not difficult to imagine a passionate ocean modeler having a hard time deciding whether to put a Honda or a Beowulf in his/her own garage.

Acknowledgement

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

References

[1] P. Angelino, C. Chapman, J. Lindheim, J. Salmon, T. Sterling, D. Becker, and D. Ridge. How to build a Beowulf tutorial, Cluster Computing Conference, CCC'97, Atlanta, 1997.

[2] D. S. Katz, T. Cwik, B.H. Kwan, J. Z. Lou, P.L. Springer, T. Sterling, and P. Wang. An Assessment of a Beowulf system for a wide class of analysis and design software. *Adv. in Eng. Software*, 29(3-6):451–461, 1998.

[3] R.D. Smith, J.K. Dukowicz, and R.C. Malone. Parallel Ocean General Circulation Modeling. *Physica*, 60:38–61, 1992.

[4] K. Bryan.Numerical Method for the Study of the World Ocean Circulation. *J. Comp. Phy.*, 4:1687–1712, 1969.

[5] M.D. Cox. Primitive Equation, 3-Dimensional Model of the Ocean. *GFDL Ocean Group Tech. Rep. 1*, GFDL/NOAA, Princeton, NJ, 1984.

[6] R. Pacanowski, R.K. Dixon, and A. Rosati. Modular Ocean Model User's Guide. *GFDL Ocean Group Tech. Rep. 2*, GFDL/NOAA, Princeton, NJ, 1992.

[7] P. Wang, D.S. Katz, and Y. Chao. Optimization of a Parallel OGCM, Proc. of SuperComputing 97, San Jose, November 1997.

[8] Y. Chao, P. Li, P. Wang, D.S. Katz, and B. N. Cheng. Ocean Modeling and Visualization on a Massively Parallel Computer. *Parallel Computing for Industrial and Scientific Applications*, Morgan Kauffman, 1999.

[9] P. Wang, B. N. Cheng, and Y. Chao. Climate Ocean Modeling. *High Performance Cluster Computing*, Prentice Hall, 2000.