LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-JRNL-211096

Parallel Computation of the Regional Ocean Modeling System (ROMS)

P. Wang, Y. T. Song, Y. Chao, H. Zhang

April 6, 2005

International Journal of High Performance Computing Applications

Disclaimer

LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-JRNL-211096

Parallel Computation of the Regional Ocean Modeling System (ROMS)

P. Wang, Y. T. Song, Y. Chao, H. Zhang

April 6, 2005

International Journal of High Performance Computing Applications

Parallel Computation of the Regional Ocean Modeling System (ROMS)

Ping Wang, Y. Tony Song*, Yi Chao*, Hongchun Zhang+
Lawrence Livermore National Laboratory
P.O. Box 808, L-561, Livermore, CA 94551 USA
* Jet Propulsion Laboratory, California Institute of Technology
+ Raytheon ITSS, Pasadena California, USA

Abstract

The Regional Ocean Modeling System (ROMS) is a regional ocean general circulation modeling system solving the free surface, hydrostatic, primitive equations over varying topography. It is free software distributed worldwide for studying both complex coastal ocean problems and the basin-to-global scale ocean circulation. The original ROMS code could only be run on shared-memory systems. With the increasing need to simulate larger model domains with finer resolutions and on a variety of computer platforms, there is a need in the ocean-modeling community to have a ROMS code that can be run on any parallel computer ranging from 10 to hundreds of processors. Recently, we have explored parallelization for ROMS using the MPI programming model. In this paper, an efficient parallelization strategy for such a large-scale scientific software package, based on an existing shared-memory computing model, is presented. In addition, scientific applications and data-performance issues on a couple of SGI systems, including Columbia, the world's third-fastest supercomputer, are discussed.

Keywords: regional Ocean modeling, parallelization, scalability, performance

1 Introduction

Ocean modeling plays an important role in both understanding current climatic conditions and predicting future climate change. In situ oceanographic instruments provide only sparse measurements over the world ocean. Although remote-sensed data from

satellites cover the globe, they only provide information on the ocean surface. Information below the ocean surface has to be obtained from three-dimensional ocean models.

ROMS solves the free-surface, hydrostatic, primitive equations over varying topography by using stretched terrain-following coordinates in the vertical and orthogonal curvilinear coordinates in the horizontal. The objective of this model is to enable the study of complex coastal ocean problems, as well as basin-to-global scale ocean circulation. This is in sharp contrast to more popular ocean models, such as the Modular Ocean Model (MOM) or Parallel Ocean Program (POP), which were primarily designed for basin-scale and global-scale problems.

Recently we fully explored parallelization for the ROMS ocean model with a message-passing interface (MPI) implementation. In this paper, parallelization strategies for such a large-scale scientific software package, based on an existing shared-memory model are investigated with a MPI-programming model so that users can have great flexibility in choosing various parallel computing systems. The model's performance, efficiency, and its applications for realistic ocean modeling are discussed below.

2 ROMS-an Ocean Model Using a Generalized Topography-Following Coordinate System

The shared-memory ROMS [1] developed at the University of California at Los Angeles (UCLA) was based on the serial version of the S-Coordinate Rutgers University Model (SCRUM) [2]. This model solves the three-dimensional, free-surface, primitive equations separately for their external mode, which represents the vertical averaged flow, and the internal mode, which represents deviations from the vertically averaged flow. The external-mode equations are coupled with the internal-mode equations through the nonlinear and pressure-gradient terms [3].

A short time step is used for solving the external mode equations, which satisfies the CFL condition arising from the fast-moving surface gravity waves. In order to avoid the errors associated with the aliasing of frequencies resolved by the external steps (but unresolved by the internal step), the external fields are "time averaged" before they replace those values obtained with a longer internal step. A cosine-shaped time filter, centered at the new time level, is used to average the external fields. In addition, the separated time stepping is constrained to maintain exact volume conservation and constancy preservation properties that are both needed for the tracer equations.

In the horizontal direction, the primitive equations are evaluated using boundary-fitted, orthogonal, curvilinear coordinates on a staggered Arakawa C-grid [4]. Coastal boundaries can also be specified as a finite-discretized grid via land-sea masking.

The model is documented in each file component [5] with irregular coastal geometries and is available to many scientists and researchers for a variety of applications. The model has been shown to be capable of dealing with the irregular coastal geom-

etry, continental shelf/slope topography, and strong atmospheric forcing. It has been successfully tested for many different problems. Due to the many different applications, it is necessary to implement an efficient parallel version of ROMS that can be run on a variety of computing platforms.

3 The Parallelization of ROMS

Currently there are two major parallel computer models — a distributed-memory model and a shared-memory model. Between these two systems, a hybrid model, such as a cluster of SMPs, is also available. Each of these systems require a different programming model. On a distributed-memory computing system, a Message Passing Interface (MPI) software is usually used for intercommunication among different computing processors for applications using domain decomposition techniques, while system directives are used on a shared-memory system to parallelize sequential codes.

These three models each have their own advantages and disadvantages, but they all have to deal with similar issues, such as parallel software portability, software reuse and maintainability, and more importantly, the total time required to transform existing code into code that is executable on advanced parallel systems. The debate about whether the shared-memory or message-passing paradigm is the best is bound to continue for a while. However, many people believe that thread programming allows the general user to gain a reasonable amount of parallelism for a reasonable amount of effort. It is commonly believed that an MPI helps attain better parallel speedups and portability, but it may require more complicated programming by the user. For applications where performance and portability are more important, an MPI model might be a good choice, but for other applications where time is critical, a thread model can be applied because of its simplicity.

3.1 Parallel System and Programming Models

The SGI Origin 2000 at Jet Propulsion Laboratory in Pasadena, CA, was the system available for use during the period when this research was conducted. This system is a scalable, distributed, shared-memory architecture with MIPS R10000, 64-bit superscalar processors. The memory is physically distributed throughout the system for fast processor access. Shared-bus symmetric multiprocessing systems typically utilize a snoopy protocol to provide cache coherence. It implements a hierarchical memory-accessing structure known as NUMA (Non-Uniform Memory Access). From lowest latency to highest, there are four levels of memory access: (1) processor registers; (2) cache – the primary and secondary caches residing on the processor; (3) home memory; and (4) remote cache. Because of the hardware design, it allows users to run two different programming models — a thread programming model and a distributed-memory programming model, such as MPI code. Each model requires a different approach.

Recently, we had access to NASA's Columbia supercomputer, which ranked third on the 2005 TOP500 List of the world's fastest computers. It has 20 *SGI Altix*

3700 superclusters, each with 512 processors and global shared memory across each supercluster. Currently, the processor speed is 1.5 GHz, which makes it possible for NASA to achieve breakthroughs in science and engineering for the agency's missions, including the Vision for Space Exploration. Columbia's highly advanced architecture will also be made available to a broader national science and engineering community. We report here our early performance data from the SGI Origin 2000, and some recent performance data from the Columbia supercomputer, as well.

3.2 An MPI Programming Model for the ROMS

A parallel-thread version of ROMS has recently been developed on the SGI Origin 2000 by the UCLA ocean research group and works well for many test cases. However, the thread version is still limited by certain hardware used. In order to improve its portability and efficiency, the design of an MPI version of ROMS was required. Our objective was to design a parallel MPI version of ROMS and to minimize the modifications to the code so that the original numerical algorithms remained unchanged, and any user of ROMS could easily use this parallel version without any specific training in parallel computing.

To achieve this objective, we focused on the data structures of the code to discover all possible data dependencies. After the entire package was investigated, the horizontal 2D computing domain was chosen as our candidate for parallelization since the depth length scale is much smaller compared with the horizontal scale. Based on domain decomposition techniques and the MPI communication API (Application Programming Interface) [6], a parallel MPI ROMS has been developed. In order to achieve load balancing and to exploit maximal parallelism, a general and portable parallel structure based on domain decomposition techniques was used for the flow domain, which has 1D and 2D partition features and can be chosen according to different geometries.

For example, if the computational region is narrow, a 1D partition structure can be used so the computation domain is divided into N subdomains in one direction. For a rectangular region, a 2D partition structure with $N \times M$ subdomains in the horizontal will be used for parallelization to minimize the communication across the partition boundaries. Since ROMS covers various computational domains, the flexible partition structure is necessary to control the parallel efficiency. The depth has a much smaller scale compared with the surface area, so the 1D and 2D partition method will give a very good efficiency for this kind of application.

The MPI software is used for the internal communication encountered when each sub-domain on each processor needs its neighbor's boundary data information — two ghost cells data are needed by the numerical algorithm. The module for communication is implemented separately from the main ROMS package, and it can also be used by other sequential software applications with similar data structures for parallelization. After various tests of the communication module, the combinations of UNBLOCK RECEIVES, MPI BLOCK SENDS and MPI WAIT were used for data exchange on the partition boundaries, which provided the best results among the choices available in the MPI communication module. When a 2D partition structure

is used, an internal subdomain needs to exchange data within two ghost cells on its four side boundaries and four corners with its neighbor subdomains.

Besides communications among some internal grid points, the communication module is also required for boundary points communications if periodic boundary conditions are applied. Also, for a subdomain that has other physical boundaries rather than periodic boundary conditions or internal subdomain neighbors, it only needs to exchange its information with neighbors that are internal grid points; elsewhere it will take the physical boundary conditions instead of doing MPI communications.

With the 2D partition structure, the MPI communications for a subdomain are outlined as follows. The boundary conditions that need to be handled separately are not discussed here.

MPI communication module for ROMS :

```
Begin module

Loop sides
If (side == internal ) then
Pack data
MPI_Irecv the neighbor data
MPI_Send data on the boundary to the neighbor
Unpack data
End if
End Loop

Loop corners
If (corner == internal ) then
Pack data
MPI_Irecv the neighbor data
MPI_Send data on the boundart to the neighbor
Unpack data
End if
End Loop

Loop sides
If (side == internal ) then
MPI_Wait
End if
End Loop

Loop corners
If (corner == internal ) then
MPI_Wait
End if
End Loop
```

```
MPI_Barrier

End module
```

The implementation was carried out on the distributed memory systems, and the code ran well on the SGI Origin 2000. It can be easily ported to other parallel systems that support MPI. Recently, the code has been successfully ported to other supercomputers, like Columbia at NASA Ames (SGI Altix), the world's third-fastest supercomputer.

4 Performance

For the parallel version of ROMS, timing tests were performed on the SGI Origin 2000 and SGI Altix. Figure 1 shows the wallclock time of the MPI code required on the SGI Origin 2000 to integrate a model grid size of $1024 \times 1024 \times 20$ for a fixed total simulation time. Figure 2 shows the wallclock time of the MPI code required on the SGI Altix to integrate a model grid size of $1520 \times 1088 \times 30$ for a fixed total simulation time. The total wallclock time is significantly reduced by using more processors for both small and medium grid size problems on both systems.

Figures 3 and 4 show the speedup of the parallel MPI ROMS with a couple of different problem sizes on both the old system (SGI Origin 2000) and the new system (SGI Altix). They give excellent speedup versus the number of processors. From the point of view of scalability, a large grid size problem gives better scaling results — superlinear scalability is achieved on 20 processors for a problem with a grid size $256 \times 256 \times 20$, and the scaling results for a problem with a grid size $256 \times 128 \times 20$ are also excellent for smaller numbers of processors. On the SGI Altix, superlinear scalability is achieved on up to 200 processors for a problem with a grid size of $1520 \times 1088 \times 30$.

The usual explanation for superlinear speedup is that when more processors are involved, code fits into the cache better. Once the size of the problem becomes smaller than the number of CPUs multiplied by cache size, the superlinear speedup ends and communication overhead causes performance degradation.

The SGI Origin 2000 is an older system, and the SGI Altix is a modern system, but our code shows excellent scalability on both systems. These systems are suitable to run large-scale scientific applications with a large number of processors if the parallel code is designed to take advantage of the hardware strengths. The speedup curves have a slight non-linear bend when more processors are applied, which is due to the increase of communication work for a fixed size problem. Once we find the region with the superlinear scalability, we can adjust our problem size and the number of processors to efficiently use the scalable computing system.

Additional numerical experiments were conducted with various grid sizes and numbers of processors. Table 1 gives the performance data of the parallel ROMS code with different grid sizes for 6000 time steps. The largest problem has a global grid of

PES	1	2	4	8	16	32	64
Grid size	128x128 x20	128x256 x20	256x256 x20	256x512 x20	512x512 x20	512x1024 x20	1024x1024 x20
MPI Code	184	185	195	201	207	229	327

Table 1: Wallclock times (seconds) using different numbers of processors and grid sizes on the SGI Origin 2000.

PES	4	8	16	32	64
MPI Code	6410	3108	1635	1121	758

Table 2: Wallclock times (seconds) using different numbers of processors for the Pacific Ocean model with a fixed total simulation time on the SGI Origin 2000 for a small size problem (1024 x 1024 x 20).

PES	32	64	128	256
MPI Code	134725	57595	27815	19915

Table 3: Wallclock times (seconds) using different numbers of processors for the North Pacific Ocean model with a fixed total simulation time on the SGI Altix for a medium size problem (1520 x 1088 x 30).

$1024 \times 1024 \times 20$ distributed on 64 processors. From this table, the MPI code scales well with up to 16 processors, but when the number of processors goes to 64, the MPI code efficiency decreases due to the increases in the communication overhead, the hierarchical memory access structure of the SGI Origin 2000, and the original design of the sequential code. The code was initially developed on sequential computing systems, so despite technological advancements, the ROMS code performance on modern computers is seriously constrained because traditional programming methods for a single-CPU system will not fully exploit the benefits of today's supercomputers and parallel systems. These systems require modern programming methods to use their fast CPUs and large memory systems. In order to use these systems efficiently, most existing codes require certain modifications. Once appropriate optimization techniques are applied, the code performance will improve dramatically [7].

Table 2 gives the performance data of the parallel ROMS code with different numbers of processors on the SGI Origin 2000 for the above application with $1024 \times 1024 \times 20$ for a fixed total simulation time. It shows very good speedup to 64 processors for this test problem.

On the SGI Altix, Table 3 gives the detailed performance data of the parallel ROMS code with different numbers of processors for the North Pacific model with a larger grid size problem of $1520 \times 1088 \times 30$ for a fixed total simulation time. It shows very good speedup to 256 processors for this real 3D problem. Numerical results of the application are discussed in the next section.

5 Applications

One of the unique applications of MPI ROMS is to simulate both the large-scale ocean circulation (usually at coarse-spatial resolutions) over a particular ocean basin (e.g., the Pacific Ocean) or the whole globe, as well as the small-scale ocean circulation (usually at high-spatial resolutions), over one or more selected areas of interest. Using the MPI ROMS described in previous sections, we have developed a scale model of the whole Pacific basin at 12.5-km resolutions and two 5-km regional models of the coasts of the North and South American continents. The scientific objective of the regional modeling approach is to simulate a particular oceanic region with sufficient spatial resolutions. The nested-modeling approach coupling a regional, high-resolution ocean model within a coarse-resolution ocean model, usually over a much larger domain, will allow one to model the oceanic response to both local and remote forcing.

The Pacific Ocean model domain extends in latitude from 45^oS to 65^oN, in longitude from 100^oE to 70^oW, and is discretized into 1520×1088 grid cells with a horizontal resolution of 12.5-km. The underlying bottom topography is extracted from ETOPO5 [8], with a minimum depth of 50 m near the coastal wall and a maximum depth of 5500 m in the deep ocean. Water depth is discretized on to 30 layers following the s-coordinates [2], with stretching parameters θ=7 and θ_b=0 to allow a good representation of the surface boundary layer everywhere in the domain. The prognostic variables of the model are the sea surface height ζ, potential temperature T, salinity S, and horizontal components of the velocity u, v. Initial T and S are

obtained from a long-term climatology [9]. The model was spun-up first for 8 years, forced with climatological (or long-term mean) air-sea fluxes. Then it was integrated for 15 years, forced with real-time air-sea fluxes during 1990-2004. The surface forcing consists of the monthly mean air-sea fluxes of momentum, heat, and fresh water derived from the Comprehensive Ocean-Atmosphere Data Set (COADS) climatology [10]. For the heat flux, a thermal feedback term is also applied [11].

Our open boundary conditions are based on the combined method of the Sommerfeld radiation conditions and a nudging term:

$$\frac{\partial \phi}{\partial t} + c_x(\frac{\partial \phi}{\partial x}) + c_y(\frac{\partial \phi}{\partial y}) = -\frac{1}{\tau}(\phi - \psi) \tag{1}$$

with τ as the time scale for nudging the model solution ϕ to external data ψ, which is obtained from monthly climatology. The phase speeds c_x and c_y are projections of the oblique radiation of Raymond and Kuo [12]. Although the techniques described above have been widely used in computational mathematics, there are many issues and difficulties related to MPI coding because the message passing involves the boundary parallelization. These issues have not been resolved in the original sequential code. In this study, we have put a great effort initially in designing the compatibility between the open boundary algorithms and the external data. By carefully using the MPI communication calls with the open boundary algorithms and the external data, the parallel code reproduced the same data from the original sequentical code. As shown in Figure 5, the open boundary conditions are now working properly in real applications.

Figure 5 shows a snapshot of the simulated sea surface temperature during the fall season toward the end of the eighth year. In agreement with observations, both the "warm pool" in the western equatorial Pacific and the "cold tongue" in the eastern equatorial Pacific are well simulated. Tropical Instability Waves are also reproduced in the eastern equatorial Pacific. These waves have a wavelength on the order of 1000 km and a periodicity of approximately 30 days propagating westward. Both the cold tongue and the Tropical Instability Waves have tremendous impact on how much cold water upwells from the deep ocean and play major roles in biological productivity and the carbon cycle in the ocean. Away from the equator, there is a clockwise ocean flow in the North Pacific and a counterclockwise one in the South Pacific. These middle latitude circulation "gyres" are forced by atmospheric winds and play an important role in the air-sea interactions that impact weather and climate.

The parallel MPI ROMS model was extensively verified against the shared-memory ROMS model that had already been verified against the original sequential code. From our current numerical results, qualitatively, the MPI-ROMS reproduces many of the observed features in the Pacific Ocean, another testimony to the success of the parallelization procedure. Encouraged by this initial success, we are currently in the process of systematically evaluatng the model solution against various existing observational datasets.

6 Concluding Remarks

We have successfully developed a parallel version of the ocean model ROMS on a shared-memory and distributed-memory system. An efficient, flexible, and portable parallel ROMS code has been designed using the MPI programming model. It produces significant speedup in execution time, which will dramatically reduce the total data processing time for complex ocean modeling. The code scales excellently for certain numbers of processors, and achieves good speedup in performance as the number of processors increases. Superlinear scalability has been obtained on up to 200 precessors on the Columbia, which is sufficient for a regional ocean model to achieve high-resolution simulations with a moderate numbers of processors. Although the code experienced slow down for larger number of processors, it can be improved if optimization techniques are applied. The code is ready to be ported to any shared-memory or distributed-memory parallel system that supports the thread programming model or the MPI programming model.

Based on the parallel version of ROMS, new numerical simulations of ocean flows have been carried out, and a detailed numerical study of the Pacific coast shows many interesting features. The parallel code has been also applied to study other coastal oceans. Several coastal models have been successfully implemented by using the parallel ROMS, and the data have been verified by the real ocean data from satellites, which will be reported separately. The present results illustrated here clearly demonstrate the great potential for applying this approach to various realistic scientific applications.

7 Acknowledgments

The research described in this paper was performed at the Jet Propulsion Laboratory (JPL), California Institute of Technology, under contract to the National Aeronautics and Space Administration. The SGI Origin 2000 is provided by the JPL Supercomputer Project, and the SGI Altix is provided by NASA Advanced Supercomputing Division. The authors wish to acknowledge Dr. Alexander F. Shchepetkin and Professor James C. McWilliams at UCLA for providing the original ROMS code and for helping during the course of implementation of the parallel MPI ROMS. The authors also wish to acknowledge Dr. Peggy Li for the visualization work. The write up of this paper was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under Contract W-7405-ENG-48.

References

[1] A. F. Shchepetkin and J. C. McWilliams. The regional oceanic modeling system: A split-explicit, free-surface, topography-following-coordinate ocean model. *Ocean Modelling, in press.*

[2] Y. T. Song and D. Haidvogel. A semi-implicit ocean circulation model using a generalized topography- following coordinate system. *J.Comp. Physics*, 115:228–244, 1994.

[3] Y. T. Song. A general pressure gradient formation for ocean models, part I: Scheme design and diagnostic analysis. *Mon. Wea. Rev.*, 126:3213–3230, 1998.

[4] A. Arakawa and V. R. Lamb. *Computational design of the basic dynamic processes of UCLA general circulation model.* Methods in Comput. Phys., Academic Press, 1997.

[5] K. S. Hedstrom. Draft user's manual for an s-coordinate primitive equation ocean circulation model, Report. *Institute of Marine and Coastal Sciences*, 1997.

[6] W. Gropp, E. Lusk, and A Skjellum. *Using MPI: Portable Parallel Programming with the Message-Passing Interface.* MIT Press, 1999.

[7] P. Wang, D. S. Katz, and Y. Chao. Optimization of a parallel ocean general circulation model. *Proceedings of Super Computing 97*, 1997.

[8] ETOPO5. NOAA Product Information Catalog. *Washington, DC: US Dept. of Commerce*, page 171, 1998.

[9] S. Levitus and T.P. Boyer. World Ocean Atlas 1994. *NOAA Atlas NESDIS*, page 117, 1994.

[10] A.M. Da Silva, C.C. Young, and S. Levitus. Atlas of surface marine data. *NOAA Atlas NESDIS*, 1:83, 1994.

[11] B. Barnier, Siefridt L., and P. Marchesiello. Thermal forcing for a global ocean circulation model from a three-year climatology of ECMWF analysis. *J. Marine System*, pages 363–380, 1995.

[12] W. H. Raymond and H. L Kuo. A radiation boundary condition for multidimensional flows. *Quarterly Journal of the Royal Meterological Society*, 110:535–551, 1984.

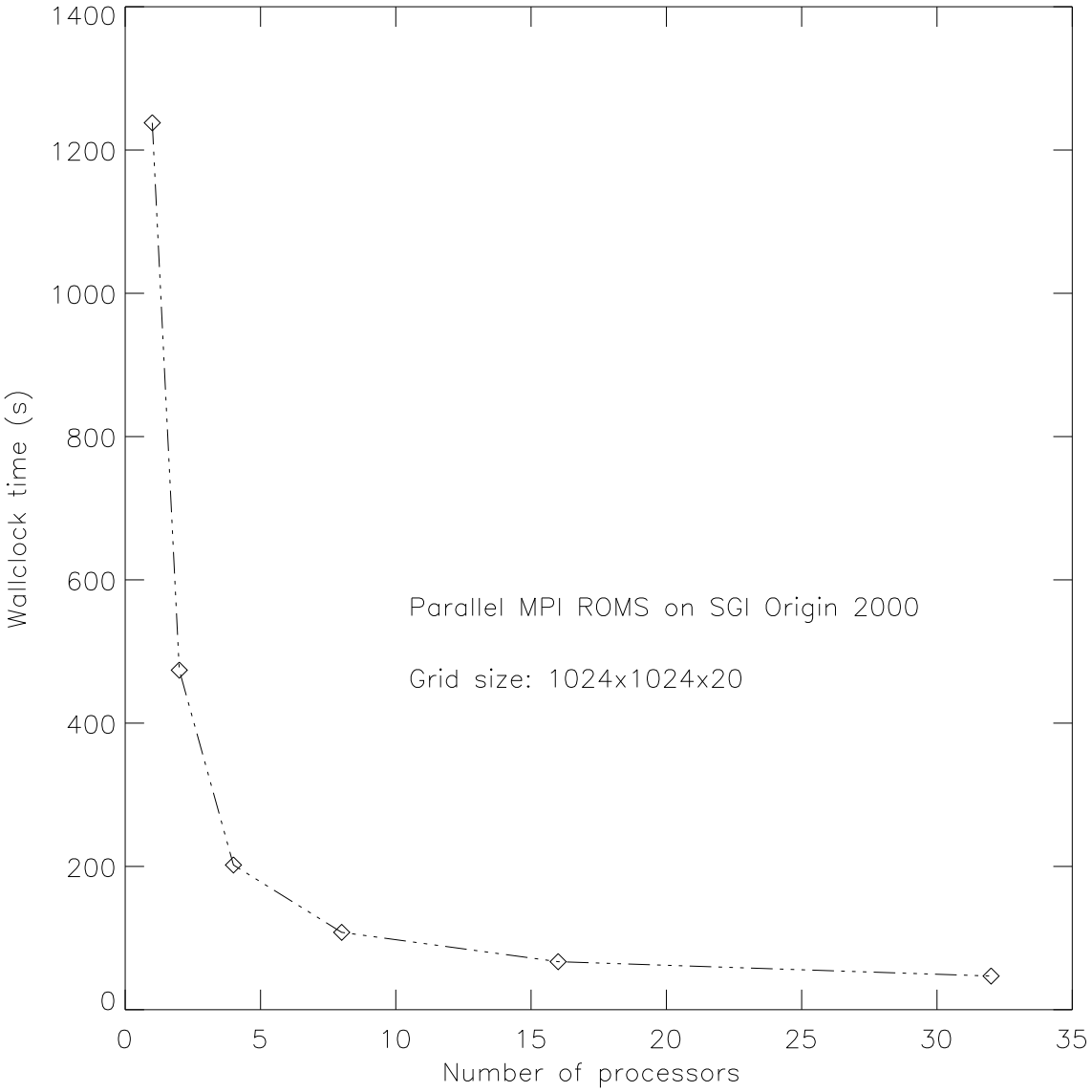

Figure 1: The wallclock time for running the parallel MPI ROMS on the SGI Origin 2000 system using different numbers of processors.

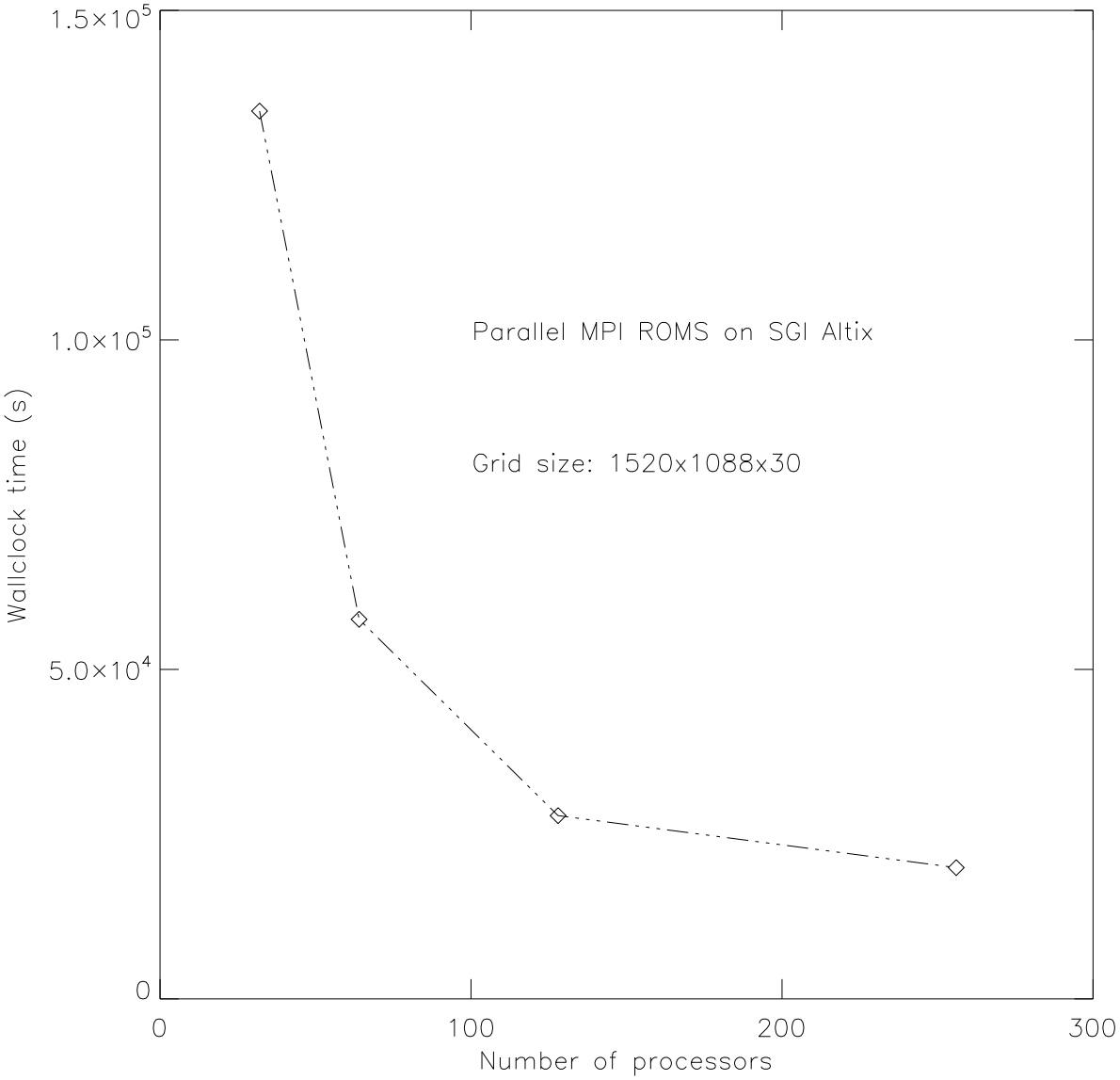

Figure 2: The real time for running the parallel MPI ROMS on the SGI Altix system using different numbers of processors .

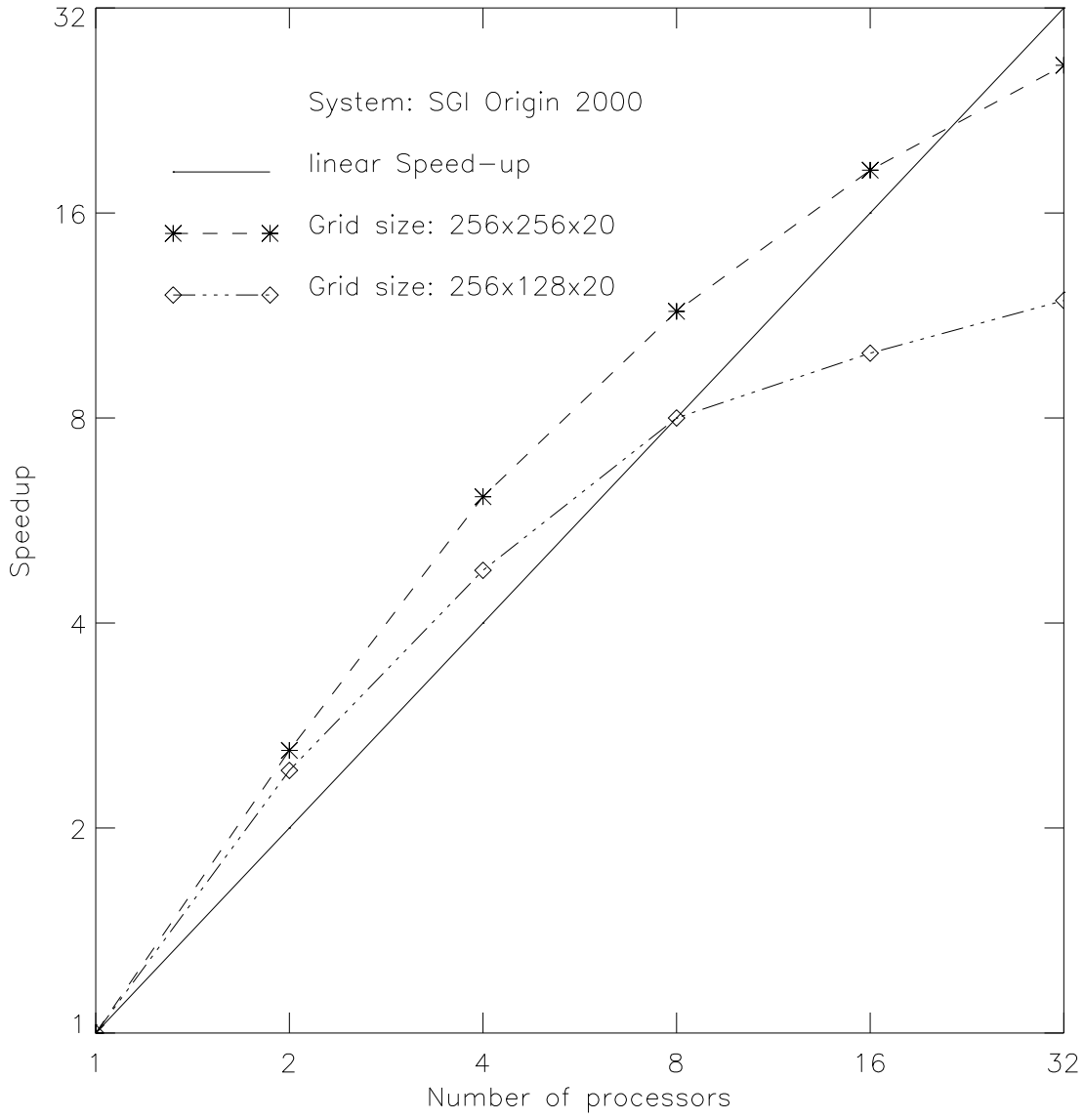

Figure 3: Speedup of the parallel MPI ROMS on the SGI Origin 2000 system with two different grid sizes.

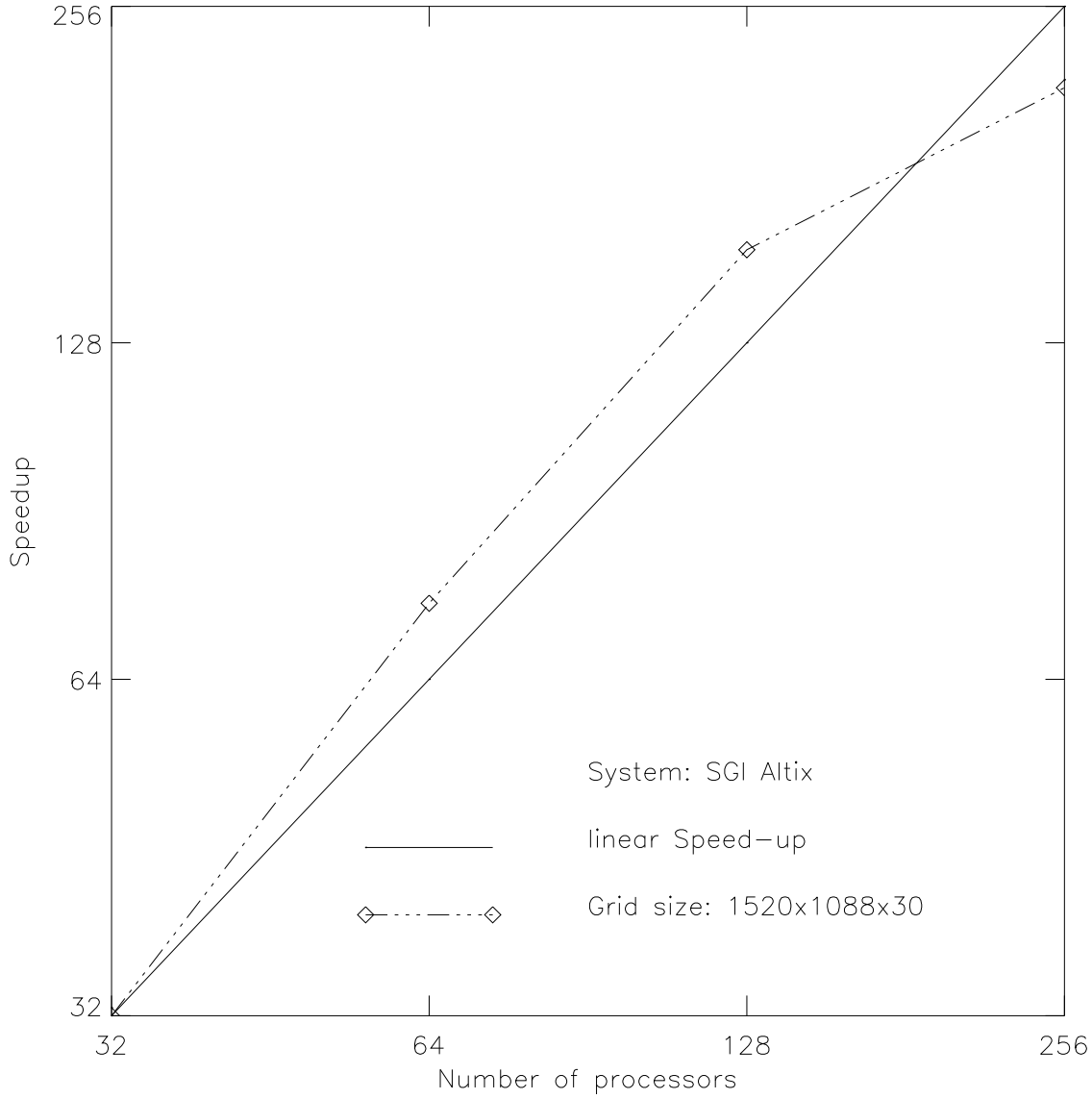

Figure 4: Speedup of the parallel MPI ROMS on the SGI Altix system.

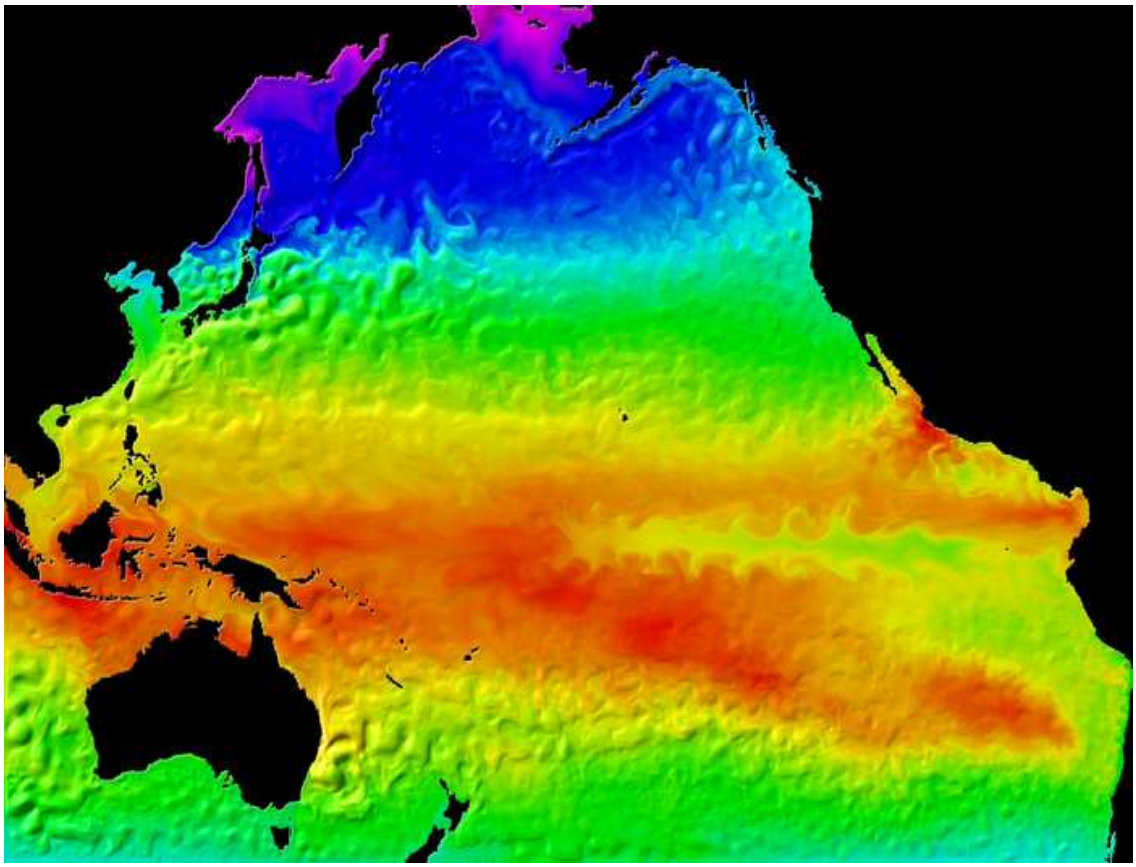

Figure 5: Snapshot of simulated sea surface temperature from the MPI-ROMS North Pacific model with a domain in latitude from 45^oS to 65^oN and in longitude from 100^oE to 70^oW .