

Performance Analysis and Optimization of the Regional Ocean Model System on TeraGrid

Yue Zuo, Xingfu Wu, and Valerie Taylor
Department of Computer Science, Texas A&M University
Email: {zuoyue, wuxf, taylor}@cs.tamu.edu

Abstract

The Regional Ocean Modeling System (ROMS) is a large-scale application code that can be configured for any region of the global ocean, ranging from local to basin scales, and is widely used in oceanography and atmospheric sciences. In this paper, we optimize the ROMS application for efficient execution on a grid environment, the TeraGrid. The strategy used to optimize the ROMS entails combining multiple communication steps and overlapping the communication with the computation as much as possible. In particular, the communication function, *MP_Exchange* is rewritten, and several new communication modules are added. To demonstrate the advantages of making this change to the communication function, we focus on optimizing the function *step2d* of the ROMS code, which dominates execution time of the code. Experiments are conducted on the TeraGrid resources at NCSA, UC/ANL, and SDSC. The experimental results show that the overall application performance is improved up to 36%.

1. Introduction

Distributed systems, such as the Distributed TeraGrid Facility [5], the Open Science Grid [3], and the European Data Grid [2] are available and provide vast compute and data resources. Large-scale applications such as ocean modeling, cosmology and gravitational-wave physics often require computational and/or data grids to obtain the larger compute and/or storage resources necessary for execution. For large-scale applications, the major challenge for grid environments is how to efficiently utilize the geographically distributed resources given the large communication latency introduced by wide area networks interconnecting the different sites.

The Regional Ocean Model System (ROMS) [4] is a free-surface, hydrostatic, primitive equation ocean model that uses stretched, terrain-following coordinates in the vertical and orthogonal curvilinear coordinates in the horizontal. This free-surface, primitive equations

ocean model is used by a rapidly growing user community for applications ranging from the basin scale to coastal and estuarine scales. Further, the ROMS has been used to model the circulation in a variety of different regions of the global ocean [1].

In this paper, we optimize the ROMS to efficiently utilize a grid environment, the TeraGrid. The optimization entails combining multiple communication steps and overlapping the communication with the computation as much as possible. The experimental results indicate that the overall application performance is improved by up to 36% depending on the network latency and the problem sizes.

The remainder of the paper is organized as follows. Section 2 describes the ROMS and the experiment platforms, the TeraGrid sites. Section 3 discusses the performance of the original ROMS on a single site and the grid environment. Section 4 presents the performance of the optimized ROMS on the grid environment. Section 5 summarizes and concludes this paper.

2. Regional Ocean Model System (ROMS) and three TeraGrid sites

The ROMS is widely used in oceanography and atmospheric sciences. In the ROMS code (version 2.2), the application code can be divided into three main kernels: initialization, simulation (*main3d*) and finalization. The simulation involves a main loop, whereby each pass of the loop corresponds to one time-step. Because the initialization and finalization steps take very little execution time, we focus on the simulation component. When instrumenting the ROMS, we divide the application into 11 events, with event 1 being initialization, event 11 being finalization, and the nine remaining events corresponding to the simulation component *main3d* as given in Table 1. The MPI version of the ROMS is configured and built using three benchmark configurations shown in Table 2, where the total number of time steps is 200 for all three benchmarks. The last three columns identify the grid size in the X, Y, and Z directions for the particular model.

Table 1. Events of main3d

event	Subroutines	Description
2	get_data	Data management
3	set_massflux, rho_eos, set_data, diag	Data management
4	bulk_flux, set_vbc	Boundary Conditions
5	lmd_vmix, omega	Vertical Mixing
6	rhs3d, wvelocity	3D Engine
7	Output	Data management
8	step2d	2D Engine
9	step3d_uv	3D Momentum
10	omega, step3d_t	3D Tracers

Table 2. Three benchmarks of ROMS

Configuration	# Time Steps	I-direction (X)	J-direction (Y)	K-direction (Z)
Benchmark 1	200	512	64	30
Benchmark 2	200	1024	128	30
Benchmark 3	200	2048	256	30

Table 3 provides the specifications of the three TeraGrid sites used in our experiments: NCSA, SDSC, and UC/ANL. Each site has the very similar IA-64 Linux cluster.

Table 3. Three Sites Used in ROMS Experiments

SITE	CPU TYPE	# Nodes	Memory per node
NCSA	IA-64 Cluster: Intel Itanium2 -1.5GHz	631	2GB
SDSC	IA-64 Cluster: Intel Itanium2 - 1.5GHz	256	2GB
UC/ANL	IA-64 Cluster: Intel Itanium2 - 1.5GHz	62	1GB

3. Performance Analysis: Original ROMS

In this section, we provide the performance of the original ROMS executed on a single site. In particular, we analyze the scalability of the ROMS for the single site. The ROMS code was instrumented with PAIDE [7] and the performance data was archived in the Prophecy database [6].

Figure 1 provides the relative speedup (normalized to single processor execution time) for the three benchmarks executed at NCSA. The figure indicates

that ROMS scales with increasing problem size and number of processors. We found similar results for the other two sites, given the common architecture at the sites.

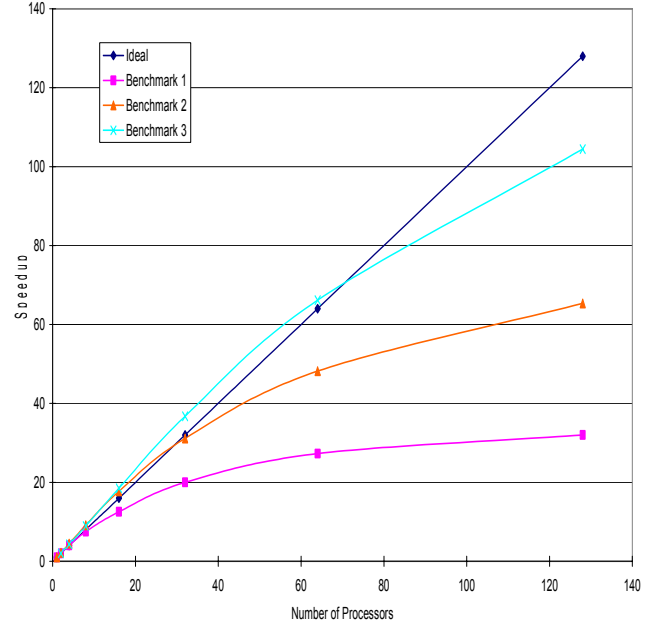
Speedup for Different Benchmarks on single site NCSA**Figure 1. Relative speedup for three benchmarks on NCSA IA-64 cluster**

Figure 2 provides execution time for the original ROMS executed across two sites for the three benchmarks; this figure also includes the execution time on a single site, NCSA, to compare the performance. It is noted that results are not given for NCSA & SDSC for Benchmark 1. For the case of multiple sites, the number of processors at each site is equal to the total number of processor divided by two such that each site has equal number of processors. For example, 8 processors on SDSC&NCSA imply 4 processors on NCSA and 4 processors on SDSC. The experimental results indicate that the execution times on multiple sites are much larger than that on a single site. This is especially the case for SDSC&NCSA as compared to UC&NCSA with benchmarks 2 and 3. This occurrence was expected as the communication latencies of the wide area networks connecting NCSA and SDSC are much larger.

Figure 2 also provides that optimization is needed with the ROMS to better utilize the TeraGrid environment. In terms of optimization, we conduct more detailed experiments on the single site execution of the ROMS to determine a good starting point for the optimization.

Cross-site Execution Time

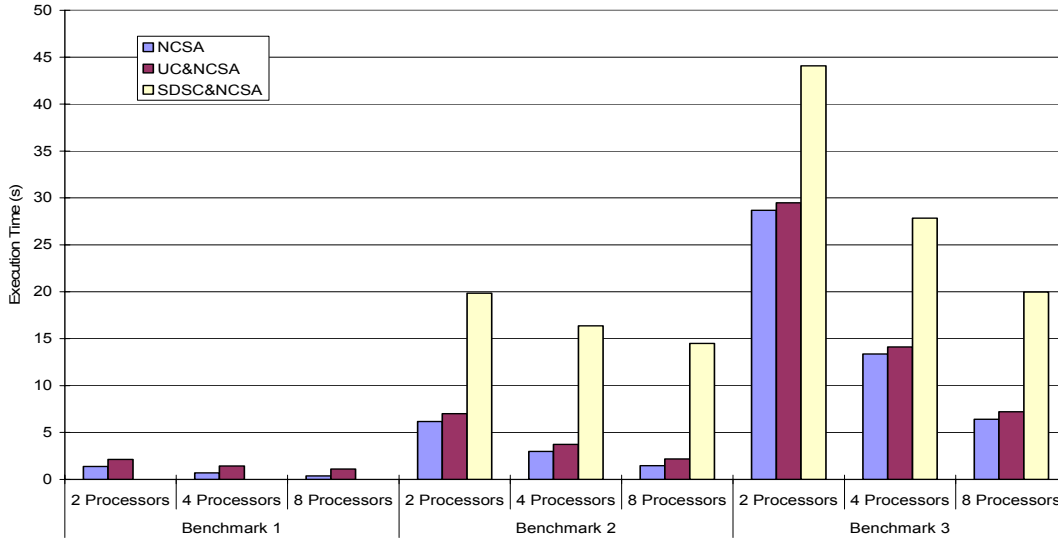


Figure 2. Performance of the original ROMS across different sites

Table 4. The number of communication calls and percentage

event	# Communication Calls	percentage
3	14	5%
5	3	15%
6	7	30%
8	168	30%

In the original ROMS code, there is only one communication function `MP_Exchange` to be called from all routines requiring communication. For the communication function `MP_Exchange`, each processor communicates data to the four neighbors based upon the following exchange pattern: EW direction (sending messages to East & West neighbors and receiving from them) and then SN direction (sending messages to South & North neighbors and receiving from them). Table 4 gives the number of communication calls for the application events along with percentage of execution time attributed to communication for that event. For example, event 6 calls the communication function, `MP_Exchange`, 7 times, and the percentage of execution time of event 6 attributed to communication time of event 6 is 30%. For event 8 (step2d), it calls the communication function 168 times, and the percentage of execution time of event 8 attributed to the 168 function calls is 30%. Our experimental results also indicate that step2d (event 8) dominates the total

application execution time and requires significant communication time. Hence, in the following section, we focus on optimizing step2d or event 8, and rewrite the communication function `MP_Exchange` to allow for overlapping of communication and computation in order to optimize the application performance.

4. Performance Optimization of ROMS

In this section, we present the optimization of the original communication routine, `MP_Exchange`. The optimization involves combining the communication patterns of EW direction and SN direction and splitting the routine into two functions to allow for overlapping the communication with computation. The original `MP_Exchange` is split into two functions: `MP_Exchange_1` and `MP_Exchange_2`. Further, the buffer allocation is removed from the inner function.

The details of the changes made to `MP_Exchange` and the impact on step2d are illustrated in Figure 3. Figure 3 provides the control flow of step2d. The original control flow is given on the left side and the modified version is given on the right. The communication function `MP_Exchange` is called four times in the original version and their sub-event numbers are: #1002, #6001, #4001 and #1015. The events #1002 and #1015 are modified by combining the communications, while #6001 and #4001 are modified by overlapping the communication with the computation.

Figure 4 provides the performance improvement of step2d across different sites for all three benchmarks.

Figure 5 provides the full application performance improvement using modified step2d across different sites for the three benchmarks. Again, each site has the equal number of processors. Figure 4 indicates that the optimized step2d results in up to 49% performance improvement on SDSC&NCSA because the application optimization reduces large communication latency.

Performance improvement of the ROMS also occurs on NCSA&UC, but is smaller than that of SDSC&NCSA due to smaller latency of the wide area network interconnecting the sites. Figure 5 shows that applying the modified step2 to the ROMS code results in up to 36% performance improvement on SDSC&NCSA.

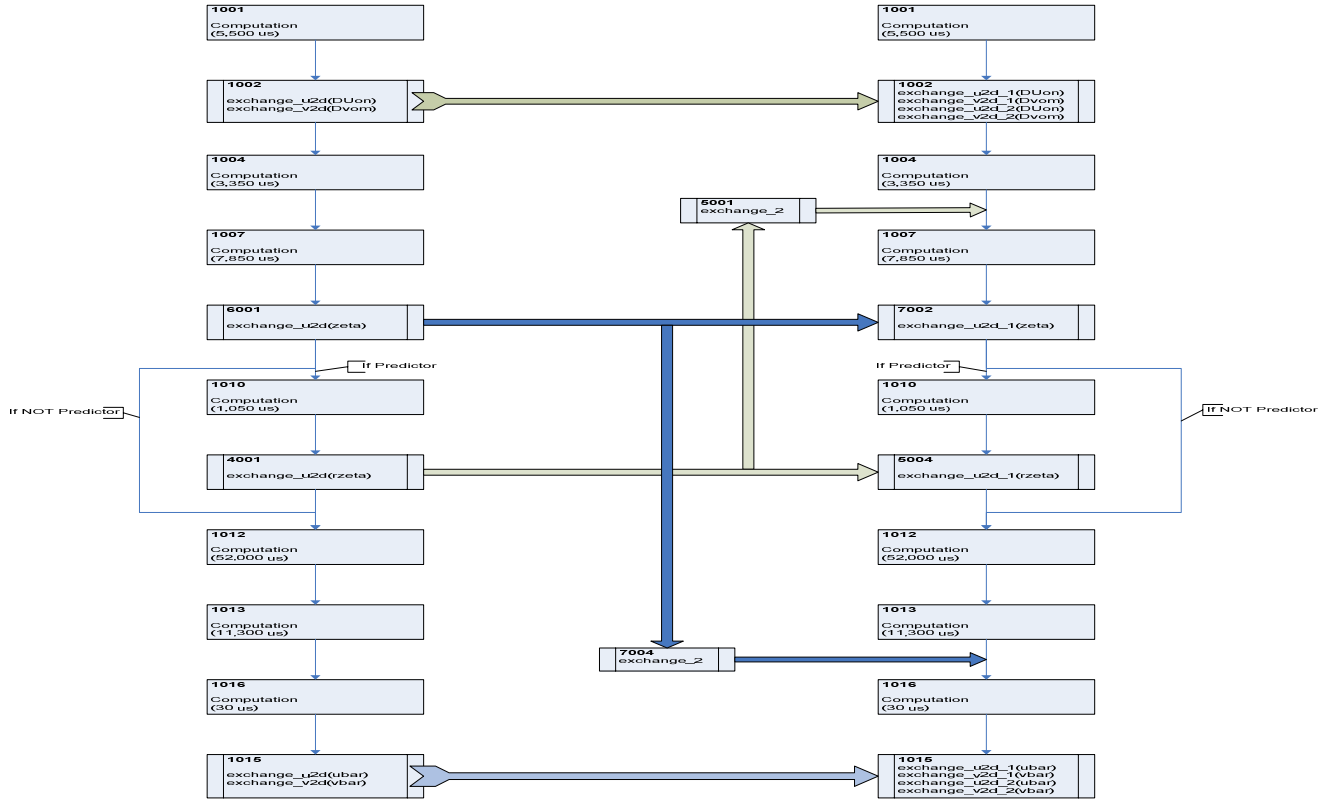


Figure 3. Control flow of the step2d

Improvement of the Modified "Step2d" Function

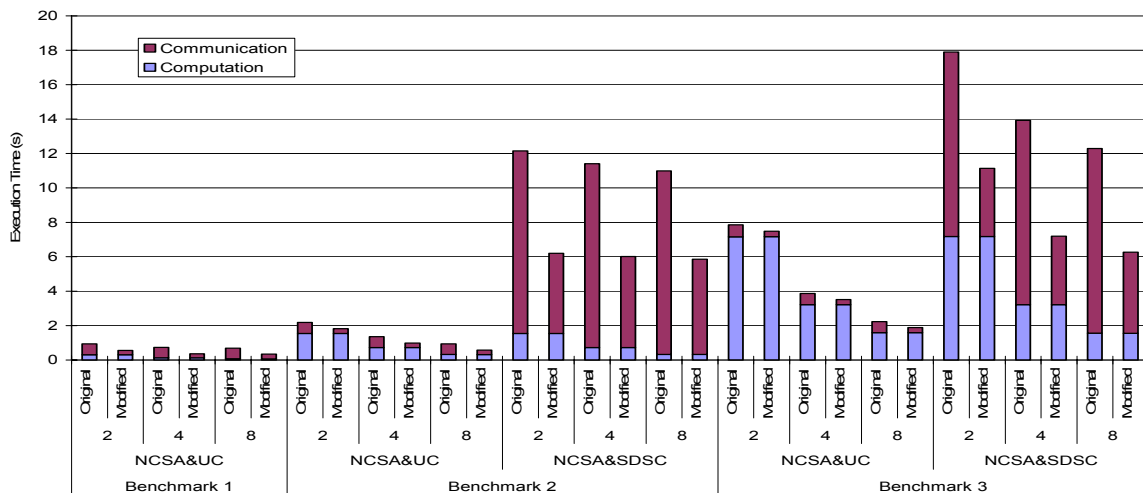


Figure 4. Performance of the modified step2d

Cross-site Execution Time Using the Modified "step2d" Function

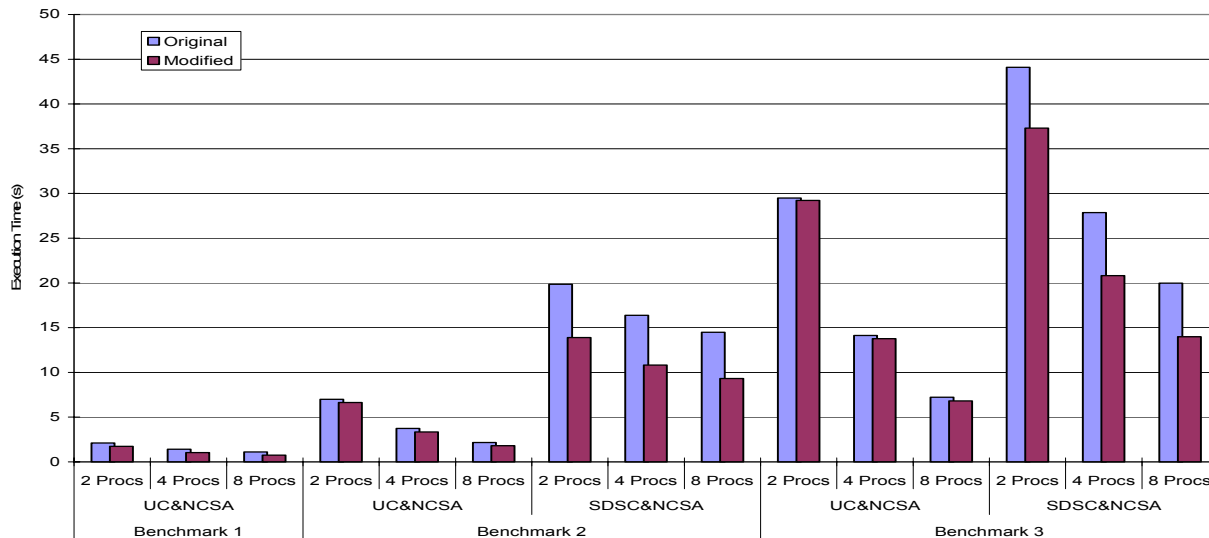


Figure 5. Performance of ROMS using the modified step2d

Conclusions

In this paper, we provided an optimization strategy to the large-scale community code, ROMS, by combining multiple communication steps and overlapping the communication with the computation for efficient execution on the TeraGrid. The experiment results showed that the overall application performance is improved up to 36%. The ROMS communication function, *MP_Exchange* was rewritten, and several new communication modules were added. Because of difficulty in reserving large number of resources simultaneously across different Grid sites, our experiments were conducted on small number of processors. Future work will explore the efficient data partitioning and load balancing issues of the ROMS on larger number of processors across the TeraGrid sites.

Acknowledgements

The authors would like to acknowledge the NCSA, SDSC and UC/ANL for the use of IA-64 clusters. This work is supported in part by NSF ITR-0086044, ITR-0085952, ANI-0225642, NASA grant NCC 2-1363, and TeraGrid TG-ASC040036T.

References

[1] H. G. Arango, A. M. Moore, A. J. Miller, B. D. Cornuelle, E. D. Lorenzo, and D. J. Neilson, *The*

ROMS Tangent Linear and Adjoint Models: A Comprehensive Ocean Prediction and Analysis System, Institute of Marine and Coastal Sciences, Rutgers University.

- [2] The European Data Grid, <http://eu-datagrid.web.cern.ch/eu-datagrid/>.
- [3] The Open Science Grid, <http://www.opensciencegrid.org/>
- [4] The Regional Ocean Model System (ROMS), <http://www.ocean-modeling.org/>
- [5] TeraGrid, <http://www.teragrid.org>.
- [6] Valerie Taylor, Xingfu Wu, and Rick Stevens, Prophecy: An Infrastructure for Performance Analysis and Modeling System of Parallel and Grid Applications, *ACM SIGMETRICS Performance Evaluation Review*, Volume 30, Issue 4, March 2003.
- [7] Xingfu Wu, Valerie Taylor and Rick Stevens, Design and Implementation of Prophecy Automatic Instrumentation and Data Entry System, in *Proc. of the 13th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS2001)*, August, 2001.
- [8] Yue Zuo, Xingfu Wu, and Valerie Taylor, Performance Analysis and Optimization of ROMS, *on-site poster, 2005 ROMS/TOMS Workshop: Adjoint Modeling and Applications*, Scripps Institute of Oceanography, La Jolla, CA, Oct. 24-26, 2005.