Effective Methods in Reducing Communication Overheads in Solving PDE Problems on Distributed-Memory Computer Architectures

Chris H.Q. Ding and Yun He
NERSC Division, Lawrence Berkeley National Laboratory
University of California, Berkeley, CA 94720, USA
chqding@lbl.gov, yhe@lbl.gov

Abstract

In solving Partial Differential Equations, such as the Barotropic equations in ocean models, on Distributed Memory Computers, finite difference methods are commonly used. Most often, processor subdomain boundaries must be updated at each time step. This boundary update process involves many messages of small sizes, therefore large communication overhead. Here we propose a new ghost cell expansion (GCE) approach which expands the ghost cell layers and thus updates boundaries much less frequently — reducing total message volume and grouping small messages into bigger ones. Together with a technique for eliminating diagonal communications, the method speedup communication substantially, upto 170%. We explain the method and implementation in details, provide systematic timing results and performance analysis on the Cray T3E and IBM SP.

Keywords: PDE, ghost cells, near neighbor communication, bandwidth, latency.

1 Introduction

As processor clock speeds double every 18 months (Moore's Law) and reach or surpass 1 Giga Hertz, the large gap between CPU processing speed and memory access rate and inter-node communication rate is becoming even bigger.

To bridge this gap, more methods need to be developed to reduce communications. A common and easily implemented approach is to change algorithms such that small messages are grouped into one big message, thus achieves higher communication bandwidth and lower communication latency. Another useful technique for multiple messages to/from multiple processors is to use asynchronous send/receive and post receives with appropriate memory buffers ahead of time.

Here we concentrate on solving partial differential equations (PDE) problems on regular domains using finite difference method, a popular method adopted in many applications. On a distributed system, each processor holds a subset of the problem domain, referred to as problem subdomains. Each processor subdomain contains one or several boundary layers, which are usually called ghost cells. Ghost cells contain most recent values of the corresponding active cells on neighboring processors. They must be updated at every time step. This is achieved by pair-wise inter-processor communication, exchanging the most recent values of ghost cells.

The number of ghost cell layers is usually determined by the order of the accuracy of the numerical discretization method. For stencils of second order accuracy for elliptic equations only one layer is required. But in many applications, such as in climate modeling, third or fourth order is commonly used. Such an example is the barotropic equations in ocean models. In these cases, two layers of ghost cells are needed. There are applications that use even higher order accuracy and more layers of ghost cells.

In the most common approach(e.g.,[1]), subdomain boundaries (ghost cells) are updated in every time step. It is straightforward, easily implemented and scales to large number of processors reasonably well. However, this update typically involves many messages of small sizes. So far to our knowledge, no study has been done that addresses the problem of substantial communication time due to the large amount of small messages exchanged between processors in every time step.

To speed up the communication, one idea is to combine messages for different time steps and exchange the bigger combined message but less frequently. In this way, the communication latency could be reduced and the communication bandwidth is increased since the message sizes are bigger. In the following, we examine the feasibility of this idea and give a brief analysis of the method. More detailed explanation could be found in [5].

1.1 Ghost Cell Expansion Method

We propose to expand the layers of ghost cells so that they can be updated much less frequently, and small messages can be combined into bigger messages. A further important advantage is that the total message volume is in fact reduced.

Consider the case of 2 layers of ghost cells. If we expand ghost cells to 2+4=6 layers, we only need to update ghost cells once every 5 time steps, with a total of 6 layers being exchanged. Without ghost cell expansion, we must update the 2 ghost cell layers every time step, leading to total of 10 layers of ghost cells being exchanged in 5 time steps. The net message volume reduction is about (10-6)/10 = 40%, not including the reduced communication latency.

We denote the number of additional ghost cell layers as expansion level `e`. The following pseudo code outlines the algorithm. Here (n_x, n_y) is the owner subdomain size, `L` is the number of ghost cell layers required for the specific PDE problem. The 2D field is declared as `field(1-L-e:nx+L+e, 1-L-e:ny+L+e)`.

```
do istep = 1, total_steps
   j = mod(istep-1,e+1)
   if (j==0) update ghost_cells
   y_start = 1 -e + j
   y_end   = ny + e - j
   x_start = 1 -e + j
   x_end   = nx + e - j
   if subdomain touches real boundaries,
      set y_start, y_end to 1 or ny
      set x_start, x_end to 1 or nx
   do iy = y_start, y_end
   do ix = x_start, x_end
         update field(ix, iy)
   enddo
   enddo
enddo
```

One can easily see that for $L = 1$ this will produce identical results for different expansion levels e, because the active domain is reduced successively and the field values are always updated using the most recent layer ghost cells.

However, for $L \geq 2$, several points are updated using several stale values on next nearest neighors. This is not a problem, because (1) the most important contributions in stencil computations are the fields on nearest neighbors (for example, 16/60 vs. 1/60 in Eq.8), which always use most recent values. (2) as convergence is approached, the differences between most recent and next-most-recent values become increasingly small. At convergence they are identical. Therefore, the results are correct. One may view this issue as the Additive Schwarz domain decomposition method on overlapping boundaries. The overlapping depth is e+1 in our case. A well known result there is that the convergence rate scales as $1/(e+1)$. This implies that equivelantly, we need update the ghost cells once every e+1 time steps.

In this ghost cell expansion method, messages are exchanged every `e+1` time steps instead of every step, we substantially reduce the communication overhead. The disadvantage is the slightly increased memory and computation costs.

1.2 Eliminating Communications with Diagonal Processors

For inter-processor communications with relatively small-size messages, the number of messages is of primary concern. For regular grids, there are 8 neighboring processors in 2D and 26 neighboring processors in 3D (see Figure 1). Although not all finite difference operators (stencils) in PDE require all the neighboring points, some of them, especially in climate model (primitive equations), do require most neighboring points. Furthermore, in ghost cell expansions, those corner points need to be updated to keep the results correct and consistent. Thus we consider the full neighbor case.

It is clear that if a given processor directly exchanges ghost cells with all its neighbors, there must be 8 messages in 2D and 26 messages in 3D. This standard approach is implemented in some applications.

Here we describe a *diagonal communication elimination* technique (first implemented in [2] in a slightly different form) that reduces the number of messages to the minimum. In 2D, the total messages are reduced from 8 to 4, and in 3D, total messages are reduced from 26 to 6. In essence, this technique requires only communications with *nearest* neighbors.

For simplicity, consider a 2D data array using a 2D domain decomposition (Figure 1). The key idea in diagonal communication elimination technique is to let the diagonal blocks go with the main ghost cell blocks. Consider the left and right communication step in Figure 1. In this step, we send to the right processor corner blocks 5 and 7 together with the main block 3. The processor receives similar blocks from right processor as well. Another similar exchanges with left processor deals with blocks 6, 8 and 4. After these two right/left exchanges, the processor covering block 1 has also diagonal blocks 5 and 6. The processor covering block 2 has also diagonal blocks 7 and 8.

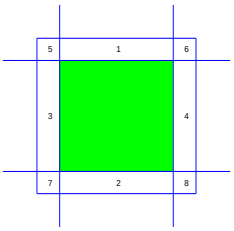

Figure 1: Illustration of ghost cells in 2D. Shaded area is active region (subdomain). 4 ghost cells blocks (1,2,3,4) are from immediate neighbors (horizontal and vertical neighbors). Another 4 corner ghost cell blocks (5,6,7,8) are from 2nd nearest neighbors (diagonal processors). In 3D, there will be 6 ghost cell blocks from immediate neighbors, 12 planar corner blocks from 2nd nearest neighbors, and 8 cubic corner blocks from 3rd nearest neighbors.

Now in the next two up/down exchanges, each proces-

sor exchanges blocks 1, 5 and 6 with its down neighbor, and exchanges blocks 2, 7 and 8 with its upper neighbor. After these two exchanges all 8 ghost cell blocks are in exactly the correct ghost cell buffers. Therefore, we need only 4 exchange communications, instead of 8 exchanges. This technique is easily generalized to 3D, where only 6 exchanges are required to communicate 26 ghost cell blocks with 26 neighbors.

To be clear, in this technique the corner blocks are moved twice to reach their final destinations in 2D, and the cubic diagonal blocks are moved 3 times to reach their final destinations. Since corner blocks are far smaller than the main blocks, they do not affect the communication time. However, the substantial reduction in total number of messages reduces the communication latency significantly, and also reduces programming complexity and traffic congestions resulting from much more messages in the communication network.

2 Analysis of GCE Method

2.1 Message Volume

For a 2D domain decomposition with the subdomain size of (n_x, n_y), the amount of total message volume (ghost cells) for each update for the conventional method and the ghost cell method (average) are:

$$V^{old} = 2L \cdot (n_x + n_y + 2L) \tag{1}$$

$$V^{new}(e) = (2L + 2e) \cdot (n_x + n_y + 2L + 2e)/(e + 1) \tag{2}$$

In most cases, we have $n_x + n_y \gg L + e$. The ratio of total message volumes for a 2D decomposition is

$$\frac{V^{new}}{V^{old}} \simeq \frac{L + e}{L(e + 1)} \tag{3}$$

For `L = 2`, and `e = 4`, this ratio is 3/5. Thus by using ghost cells layers expansion, we not only reduce the message exchange frequency, but also decrease the total message volume.

In 1D decomposition, there are two messages exchanged with up or down processors. In 3D decomposition, using the diagonal communication elimination technique, only a total of 6 messages are exchanged with its neighbors. The ratio of communication volume between the new method and the conventional method remains the identical as in Eq.3 for all three decompositions.

2.2 Communication Time

To analyze the communication time T_{comm}, we assume it can be approximated by a simple "(message-volume)/bandwidth + latency" model.

For a 2D domain decomposition, the communication times for updating the ghost cells in each time step for the conventional method and new method (average) are:

$$T^{old}_{comm} = 2L(n_x + n_y + 2L)8/B + 4T_L \tag{4}$$

$$T^{new}_{comm}(e) = \frac{(2L + 2e)(n_x + n_y + 2L + 2e)8/B + 4T_L}{e + 1} \tag{5}$$

where B is the bandwidth and T_L is the communication latency. The communication time for 1D and 3D decompositions could be calculated in a similar fashion.

For large messages, we have measured the communication bandwidth and latency B, T_L [4]: On the Cray T3E, $B = 300$MB/sec, $T_L = 17\mu$sec, and for the IBM SP, $B = 133$MB/sec, $T_L = 26\mu$sec. Using $n_x = 800, n_y = 800$ as an example, the theoretical maximum speedup for 2D decomposition could be more than two-fold for $L = 2$ on both machines.

2.3 Memory Usage and Computational Cost

The new method has the disadvantage of using slightly more memory and computation than the conventional method. This is because even though we only need to update fields in active cells, some fields in ghost cells are updated in order to maintain the correct timelineness. The overhead is summarized in Table 1. In all practical cases, they remain very small. In 2D decomposition, for $e = 4$ and $n_x = n_y = 800$, the overhead for memory and computation are 2% and 0.5%, respectively.

Table 1: Overhead in memory usage ($\Delta M/M$) and in computational cost ($\Delta C/C$) for GCE method expressed as a ratio in 1D, 2D and 3D decompositions.

	$\Delta M/M$	$\Delta C/C$
1D	$2e/n_x$	$e/2n_x$
2D	$2e/n_x + 2e/n_y$	$e/2n_x + e/2n_y$
3D	$2e/n_x + 2e/n_y + 2e/n_z$	$e/2n_x + e/2n_y + e/2n_z$

3 Test problem

Although the original motivation for this work is on atmosphere and ocean models, we test the ghost cell expansion method on a simpler 2D static heat distribution problem, to clearly illustrate some performance issues.

The 2D problem is governed by the Laplacian equation,

$$\frac{\partial^2 u}{\partial^2 x} + \frac{\partial^2 u}{\partial^2 y} = 0 \tag{6}$$

on a rectangular region with Dirichlet boundary conditions. After discretization on a regular grid, the problem is solved by a finite difference method. In 2nd order accuracy, we perform Gauss-Seidel iterations using 5-point

stencils

$$u(x,y) = \frac{1}{4}[u(x-1,y)+u(x+1,y)+u(x,y-1)+u(x,y+1)] \quad (7)$$

This stencil requires one layer of ghost cells (L=1). In 4th order accuracy, we use 9-point stencils

$$\begin{aligned} u(x,y) &= \frac{1}{60}[16u(x-1,y) + 16u(x+1,y) \\ &+ 16u(x,y-1) + 16u(x,y+1) - u(x-2,y) \\ &- u(x+2,y) - u(x,y+2) - u(x,y-2)] \end{aligned} \quad (8)$$

This stencil requires two layers of ghost cells (L=2).

The tests are performed on both CRAY T3E and IBM SP at NERSC. Four tests with the following parameters are studied:

- Test 1: Global size 3200×3200, P = 16, L=1.
- Test 2: Global size 3200×3200, P = 16, L=2.
- Test 3: Global size 6400×6400, P = 64, L=1.
- Test 4: Global size 6400×6400, P = 64, L=2.

where P is the total number of processors. Note that using 2D decomposition, all four tests have the same local domain size of 800×800. We performed a fixed total of 512 Gauss-Seidel iterations in all 4 tests.

4 Performance Analysis

In Figure 2, calculation time (a), communication time (b), and total time (c) at different ghost cell expansion levels (e) are shown on the left for the four tests on IBM SP. One can see that calculation times remain very much the same, confirming the analysis in Section 2.3. Note that since the local domain size is 800×800 in all tests, calculation time for Tests 1 and 3 should be same. Similarly, curves for Tests 2 and 4 coincide too.

As expansion level increases, the communication times steadily decrease, leading to fairly large (upto a factor of 2.8) communication speedup. The total time of communication and calculation also decrease steadily. The speedup of total solution time increases by 25% at $e = 8$.

In Figure 3, timing results on Cray T3E are shown. They are similar to that in SP. Again calculation times remain almost same, while the communication times speedup by 170%. On the T3E, however, the total problem speedup is only about 3%. This is because the communication on the T3E is much faster, so the communication time is only about 3% of the total computational time. Even though this small proportion is speedup by 170%, the total time does not drop very much.

On communication time reduction or speedup, two layer ghost cell (L=2) cases always have higher speedup, than those with one-layer (L=1) cases. The speedups are comparable to the theoretical estimation in Section 2.2. The differences are: tests with P=16 has the smaller communication speedup than those with P=64 on IBM

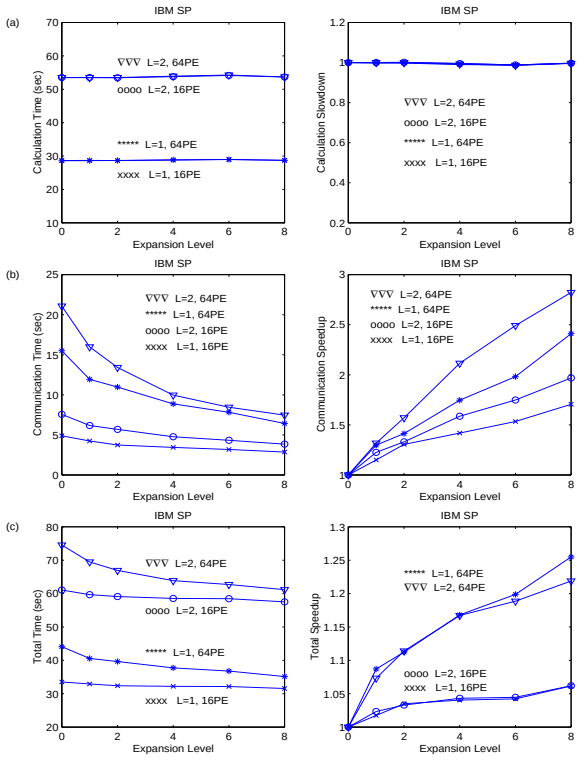

Figure 2: (a) calculation time and speedup, (b) communication time and speedup, (c) total time and speedup for these four tests on IBM SP.

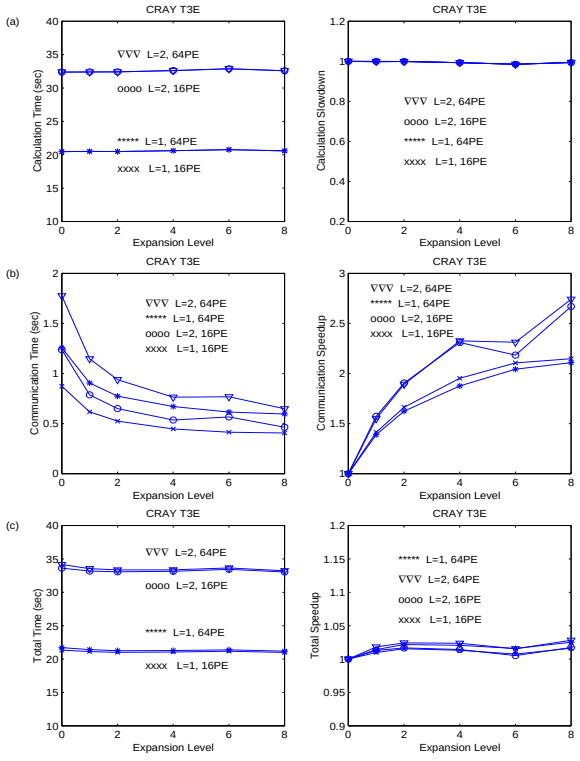

Figure 3: (a) calculation time and speedup, (b) communication time and speedup, (c) total time and speedup for these four tests on Cray T3E.

SP, while they are very close on Cray T3E. This is due to the optimal message bandwidth was not reached on IBM SP for a small size problem.

Comparison of the T3E timing with and the SP timing shows some interesting points. For these stencils type finite difference computations, the T3E (450 MHz Alpha EV5, peak 900 MFlop/*sec*) achieves a higher computational speed than the SP (200 MHz Winterhawk, peak 800 MFlop/*sec*) does on per processor basis. For the 9-stencil calculations, the T3E obtained 121.4 MFlop/*sec* while the SP obtained 73.2 MFlop/*sec* per processor.

On communication, the T3E is much faster than SP: the T3E requires 0.65 *sec* while the SP requires 7.5 *sec* for $L = 2$ case on 64 processors. This is factor of 12 difference. Although on the measured point-to-point message latency and bandwidth, the SP is only factor of 2 slower than the T3E, actual measurements on communications involving more than 2 processors always show the SP much slower than the T3E [4]. This indicates a very significant communication traffic congestion in the SP interconnect, possibly due to the serial access on the adaptor between the SMP node and the commnucation switch.

Scaling from smaller number of processors (16) to larger number of processors (64), the communication time T_{comm} always increases on both the T3E and the SP. T_{comm} increases about 40% on the T3E (Figure 3), and is almost doubled on the SP (Figure 2). This is another indication that the SP communication has some serious traffic congestion.

5 Conclusions and Discussions

In this paper, we propose and analyze a ghost cell expansion method for reducing communications in solving PDE problems in detail. Both the total message volume and total number of messages are reduced leading to significant speedup in communications. The key to implement GCE method is to support variable layers of ghost cells and update them efficiently. The diagonal communication elimination technique we introduced is critical for efficiency. It reduces the total number of messages exchanged between processors from 8 to 4 in 2D domain decomposition and from 26 to 6 in 3D domain decomposition. We provide key implementation steps and carry out systematic experiments and performance analysis on IBM SP and Cray T3E. On both computers, the GCE method speedup communications up to 170%.

In general when solving PDE problems involving 3D fields, 3D domain decomposition is best because it has the least total amount of communication volume in updating ghost cells. However, many other considerations are sometimes more important than communication cost. For example, the vertical direction is very special in atmosphere or ocean modeling, where parallelization along vertical direction is either undesirable or simply too complicated. In these cases, typically a 2D [6, 9] or even 1D [7], where parallelization along domain decomposition is adopted for the 3D fields, and vertical direction are entirely local to a processor. Remapping to other decompositions[4] are often necessary to facilitate other tasks such as spectral transform [6], polar filtering [8] and parallel I/O [3].

In addition, due to array indexing, some data packing and unpacking are necessary. In these procedures, moving a block of data, rather than moving one array element at a time, will increase speed.

Acknowledgment We thank Dr. K. Bryan for valuable discusions on using more memory for reducing communications. This work is supported by the Office of Biological and Environmental Research, Climate Change Prediction Program, and by the Office of Computational and Technology Research, Division of Mathematical, Information, and Computational Sciences, of the U.S. Department of Energy under contract number DE-AC03-76SF00098.

References

[1] V. Balaji, 2000. "Abstract Parallel Dynamical Kernels for Flexible Climate Models." Talk presented at ECMWF TeraComputing Workshop, Reading, England, Nov, 2000.

[2] C.H.Q. Ding, 1991. "Simulating Lattice QCD on a Caltech/JPL Hypercube," Int'l J. Supercomp. Appl., 5, pp:73-80.

[3] C.H.Q. Ding and Y. He. "Data Organization and I/O in a parallel ocean circulation model", Proc. Supercomputing '99, Nov 1999.

[4] C.H.Q. Ding. 2001. "An Optimal Index Reshuffle Algorithm for Multidimensional Arrays and Its Applications for Parallel Architectures", IEEE Trans. Para. Distr. Sys., 12, pp.306-315.

[5] C.H.Q. Ding and Y. He. "A Ghost Cell Expansion Method for Reducing Communications in Solving PDE Problems", Proc. Supercomputing '01, Nov 2001.

[6] J.Drake, I.Foster, J.Michalakes, B.Toonen and P. Worley, "Design and performance of a scalable parallel community climate model", Parallel Computing, v.21, pp.1571-1581,1995.

[7] J.J. Hack, J.M.Rosinski, D.L.Williamson, B.A.Boville and J.E. Truesdale, "Computational Design of NCAR community climate model", Parallel Computing, v.21, pp.1545-1555, 1995.

[8] A.A. Mirin, D. Shumaker, M.F. Wehner. "Efficient Filtering Techniques for Finite-Difference Atmospheric General Circulation Models on Parallel Processors." Parallel Computing, v.24, pp.729-740, 1998.

[9] R.D. Smith, J.K. Dukowicz, and R.C. Malone. Parallel ocean general circulation modeling. *Physica*, D60, 38, 1992. See also http://gnarly.lanl.gov/Pop/Pop.html.