A Ghost Cell Expansion Method for Reducing Communications in Solving PDE Problems

Chris Ding and Yun He
NERSC Division, Lawrence Berkeley National Laboratory
University of California, Berkeley, CA 94720
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 approach which expands the ghost cell layers and thus updates boundaries much less frequently — reducing total message volume and groupping 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 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, both efforts in system architecture design and user application are required. In memory access, for example, large multi-level (level 3) and multi-way (128-way) set associative caches are appearing in latest processors. In communication inter-connect, fast high bandwidth switches are also emerging. On applications side, blocking data into small segments to fit processor cache structure seems to be a promising approach to reduce memory access time [4].

Reducing communication time often involves algorithmic changes. 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.

Yet with the large and growing gap between processing speed and communication speed, more methods need to be developed to reduce communications. 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 cells layers are 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, subdomain boundaries (ghost cells) are updated in every time step. It is straight forward, easily implemented and scales to large number of processors reasonably well. However, so far no study has been done that addresses the problem of substantial communication time due to the large amount of small messages to be exchanged between processors.

To speed up the communication, one idea is to combine messages for different timesteps 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 an in-depth analysis of the method.

1.1 Ghost Cell Expansion (GCE) 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, results in 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 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 appropriate
    y_start, y_end, x_start, x_end to  1 or nx or ny
   do ix = x_start, x_end
   do iy = y_start, y_end
         update field(ix, iy)
   enddo
```

```
    enddo
  enddo
```

One can easily see that this will produce identical results for different expansion level e, because the active domain is reduced successively and and the field values are always updated using the most recent L layer ghost cells. Note messages are exchanged every `e+1` time steps instead of every step, we substantially reduce the communication overhead. The disadvantage is the increased memory and computation cost.

1.2 Eliminating Communications with Diagonal Processors

For inter-processor communications with relatively small-size messages, the number of message 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 needs 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 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.

Now in the next two up/down exchanges, each processor 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 move 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 due to much more messages in the communication network.

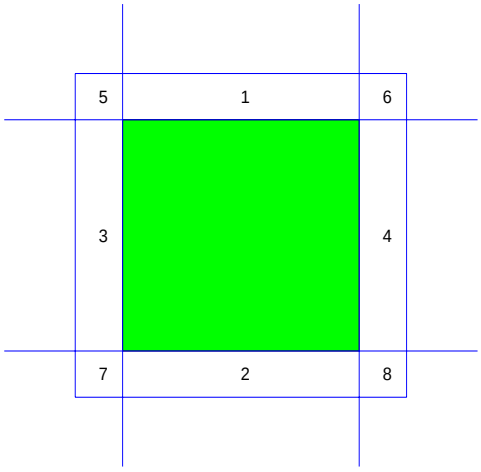

Figure 1: Illustration of ghost cells in 2D. Shaded area is active region (subdomain). 4 ghost cells blocks (1,2,3,4) are from immediate neighbors (horizontal and vertical neighbors). Another 4 ghost cells corner blocks (5,6,7,8) are from 2nd nearest neighbors (diagonal processors). In 3D, there will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

2 Analysis of GCE Method

2.1 Message Volume

For a 2D domain decomposition, the amount of total message volume (ghost cells) for each update for the conventional method is:

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

Message volume for each time step in ghost cell expansion method is:

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

Here we update ghost cells once every `e+1` time steps, therefore, we divide the volume of each update by `e+1` to get volume per time step.

The ratio of total message volumes for a 2D decomposition is:

$$\frac{V^{new}}{V^{old}} = \frac{(2L + 2e) \cdot (n_x + n_y + 2L + 2e)}{2L \cdot (n_x + n_y + 2L) \cdot (e + 1)} \simeq \frac{L + e}{L(e + 1)} \tag{3}$$

We have $n_x + n_y \gg L + e$ in most cases. 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.

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 time for updating the ghost cells in each timestep for the conventional method is:

$$T_{comm}^{old}(2D) = \frac{2L(n_x + n_y + 2L)8}{B} + 4T_L \tag{4}$$

where B is the bandwidth and T_L is the communication latency.

When ghost cells are expanded e layers, the ghost cell update is done once every `e+1` time steps) for the new method. Thus the average communication time per timestep is:

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

For large messages, we have measured B, T_L [3]: For Cray T3E,

$$B = 300\text{MB/sec}, \quad T_L = 17\mu\text{sec},$$

and for IBM SP,

$$B = 133\text{MB/sec}, \quad T_L = 26\mu\text{sec}.$$

Using $n_x = 800, n_y = 800$ as an example, we calculate the ratio of communication time for the new method and the old method as the function of required level L and expansion level e for 2D domain decomposition. as shown in Figure 2. We see that the speedup increases when L increases. And the speedup increases first while e increases up to 8. When $L = 2$, the maximum speedup could be more than two-fold. Due to different bandwidth and latency on two machines, the speedup on T3E is slightly larger than on SP.

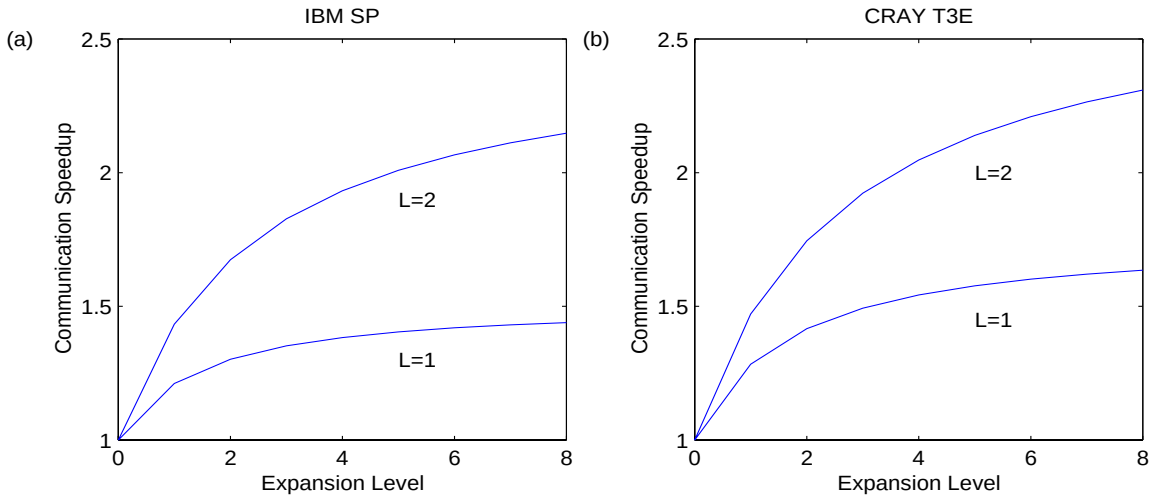

Figure 2: Communication speedup as the function of the ghost cells layers required level `L` and expansion level `e` for a 2D domain decomposition. Here n_x `=` n_y `= 800`.

2.3 Memory Usage and Computational Cost

The new method has the disadvantage of using slightly more memory and computation than the conventional method.

The extra memory used in the new method for each subdomain is

$$\begin{aligned} \Delta M &= (n_x + 2L + 2e)(n_y + 2L + 2e) - (n_x + 2L)(n_y + 2L) \\ &= (n_x + n_y + 4L)2e + 4e^2 \end{aligned} \tag{6}$$

Comparing to the total memory usage for the conventional method, $M = (n_x + 2L)(n_y + 2L)$, this is a rather small amount:

$$\Delta M/M \simeq 2e/n_x + 2e/n_y \tag{7}$$

We have $n_x, n_y \gg L, e$. For $e = 4$ and $n_x = n_y = 800$, this ratio is 2%.

The new method shrinks the local computational region by the size of 1 at both dimensions every time step, so the total computational cost for each time step (here divide by `e+1` since boundary update is done every `e+1` time steps) is:

$$\Delta C = \sum_{k=1}^{e} [(n_x + k)(n_y + k) - n_x n_y]/(e + 1) = (n_x + n_y)e/2 + e(2e + 1)/6$$

Comparing to the computational cost in the conventional method for each time step, $C = n_x n_y$, this is a rather small fraction:

$$\Delta C/C \simeq e/(2n_x) + e/(2n_y) \tag{8}$$

For $e = 4$ and $n_x = n_y = 800$, this ratio is 0.5%.

2.4 1D and 3D domain decompositions

The above analysis gives the number of messages, message volume, communication, memory usage and computational cost for solving a 2D PDE problem with 2D domain decomposition. It should be noted that Eqs.(3, 7, 8) remain identical for 3D problems using 2D domain decomposition.

For 3D problems, one may use 3D domain decomposition, as well as 2D and 1D domain decompositions. In all these cases, the GCE method can be beneficially employed.

Consider the number of messages in each ghost cell update. 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.

Next, consider the communication volume and time. In 1D decomposition, the communication time spent on ghost cell update for the 3D array are

$$T_{comm}^{old}(1D) = \frac{2Ln_x n_z \cdot 8}{B} + 2T_L \tag{9}$$

$$T_{comm}^{new}(1D) = (\frac{(2L + 2e)n_x n_z \cdot 8}{B} + 2T_L)/(e + 1). \tag{10}$$

In 3D decomposition, the communication time spent on ghost cell update for the 3D array are

$$T_{comm}^{old}(3D) = \frac{2L(n_x n_y + n_x n_z + n_y n_z) \cdot 8}{B} + 6T_L, \tag{11}$$

$$T_{comm}^{new}(3D) = (\frac{(2L + 2e)(n_x n_y + n_x n_z + n_y n_z) \cdot 8}{B} + 6T_L)/(e + 1). \tag{12}$$

For all cases, the ratio of communication volume remains identical as Eq.(3).

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

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.

The overhead due to GCE method in memory usage and computational time can be calculate following same procedures leading to Eqs.(7,8). They are summarized in Table 1. In all practical cases, they remain very small.

In general when solving PDE problems involving 3D fields, 3D domain decomposition is best because it has the least total amount communication volume in updating ghost cells. However, many other considerations are sometimes more important than consideration of communications. 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 or even 1D domain decomposition is adopted for the 3D fields, and vertical direction are entirely local to a processor. Remapping to other decompositions[3] are often necessary to facilitate other tasks such as spectral transform in atmospheric models, polar filtering and parallel I/O.

2.5 Implementation

The key to implement GCE method is to support variable layers of ghost cells and update them efficiently. The diagonal communication elimination technique described earlier is critical for efficiency. 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. A number of existing software supports variable ghost cell layers [1, 5, 6].

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,

$$\frac{\partial^2 u}{\partial^2 x} + \frac{\partial^2 u}{\partial^2 y} = 0 \tag{13}$$

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

$$u(x,y) = \frac{1}{4}[u(x-1,y) + u(x+1,y) + u(x,y-1) + u(x,y+1)]. \tag{14}$$

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

$$u(x,y) = \frac{1}{60}[16u(x-1,y) + 16u(x+1,y) \quad + \quad 16u(x,y-1) + 16u(x,y+1) \\ - \; u(x-2,y) - u(x+2,y) \quad - \quad u(x,y+2) - u(x,y-2)] \tag{15}$$

This stencil requires two layers of ghost cells (L=2). The iteration stop either when a convergence tolerance is rearched or maximum numbers of iteration steps is reached.

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 total of 512 Gauss-Seidel iterations.

4 Performance Analysis

In Figure 3, 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 test on IBM SP. One can see that calculation times remain very much the same, confirming the analysis in section 2. Note that since the local domain size is 800×800 in all tests, calculation time for Tests 1 and 3 should be same, as confirmed in Figure 1(a). 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 4, 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 T3E, however, the total problem speedup is only about 3%. This is because the communication on 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 270%, 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 those theoretical estimation in Figure 2. The differences are: tests with P=16 has the smaller communication speedup than those with P=64 on SP, while they are very close on T3E. This is due to the optimal message bandwidth was not reached on SP for a small size problem.

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

On communication, T3E is far more faster than SP: T3E requires 0.65 sec while 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 [see Eqs.(6,7)], SP is only factor of 2 slower than

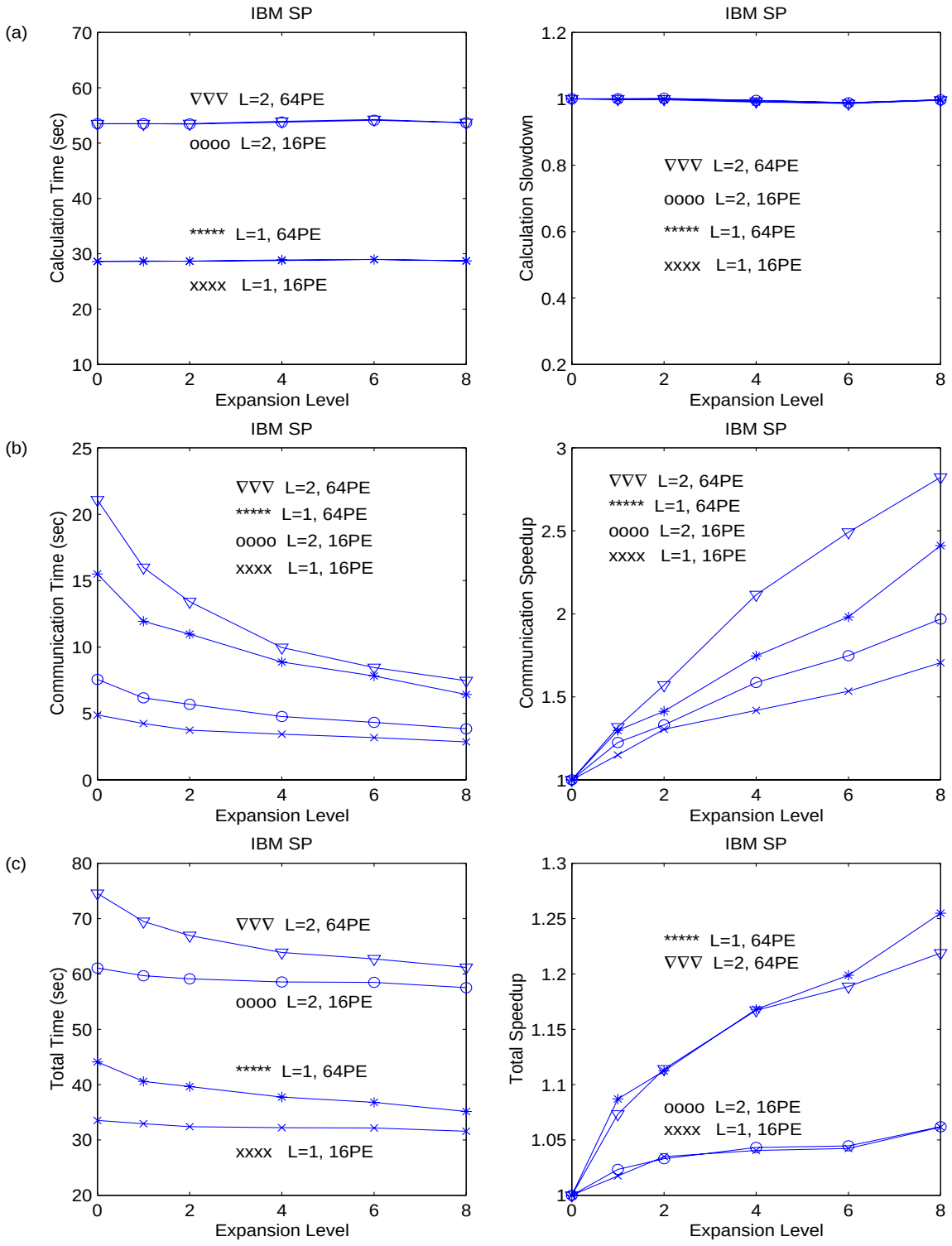

Figure 3: (a) calculation time and slowdown, (b) communication time and speedup, (c) total time and speedup for these four tests on IBM SP.

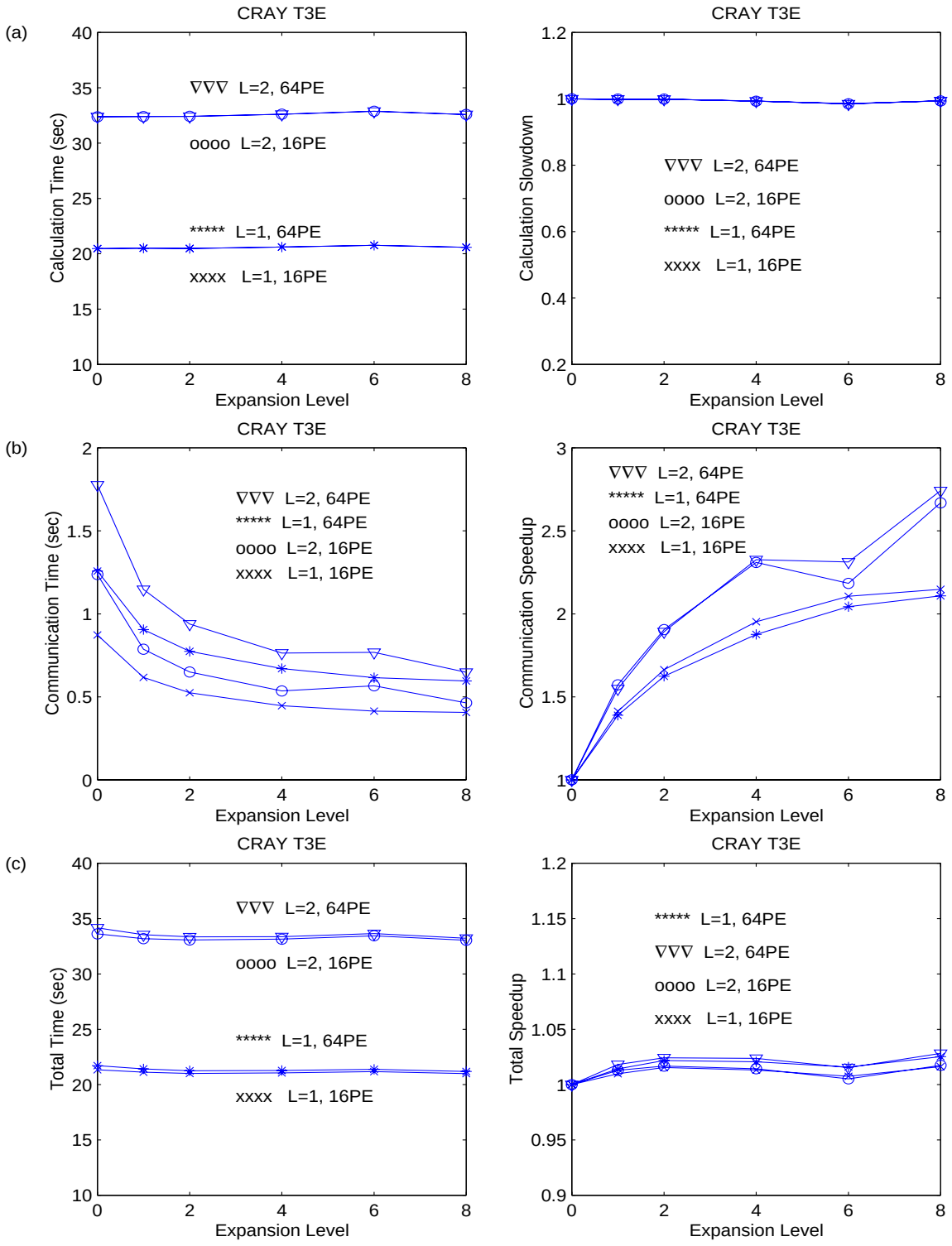

Figure 4: (a) calculation time and slowdown, (b) communication time and speedup, (c) total time and speedup for these four tests on Cray T3E.

T3E, actual measurements on communications involving more than 2 processors always show SP much slower than T3E [3]. This indicates a very significant communication traffic congestion in the SP interconnect, possibly due to the adaptor between the node and switch.

Scaling from smaller processors (16) to larger processors (64), communication time T_{comm} always increases in both T3E and SP. T_{comm} increases about 40% on T3E (Figure 4), and is almost doubled on SP (Figure 3). This is another signal that SP communication has some congestion.

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. 2001. An Optimal Index Reshuffle Algorithm for Multidimensional Arrays and Its Applications for Parallel Architectures. IEEE Trans. Para. Distr. Sys., 12, pp.306-315.

[4] J. Drake and P. Worley. 2000. Unpublished results.

[5] POOMA: Parallel Object-Oriented Methods and Applications. http://www.acl.lanl.gov/pooma/

[6] OVERTURE: an object-oriented code framework for solving partial differential equations. http://www.llnl.gov/CASC/Overture/