International Journal of Multimedia and Ubiquitous Engineering
Vol. 5, No. 1, January, 2010

Parallel Mass Transfer Simulation of Nanoparticles using Nonblocking Communication

Chantana Chantrapornchai (Phonpensri) [1], Banpot Dolwithayakul[1]
Kanok Huankumnerd[2] and Sergei Gorlatch[3]

[1]*Dept. of Computing, Faculty of Science, Silpakorn University, Thailand*
[2]*Dept. of Computing, Faculty of Science, Silpakorn University, Thailand*
[3]*Universität Münster, Institut für Informatik, Münster, Germany*
[1]*ctana@su.ac.th,* [2]*kanok_h@hotmail.com,* [3]*gorlatch@math.uni-muenster*

Abstract

This paper presents experiences and results obtained in optimizing parallelization of the mass transfer simulation in the High Gradient Magnetic Separation (HGMS) of nanoparticles using nonblocking communication techniques in the point-to-point and collective model. We study the dynamics of mass transfer statistically in terms of particle volume concentration and the continuity equation, which is solved numerically by using the finite-difference method to compute concentration distribution in the simulation domain at a given time. In the parallel simulation, total concentration data in the simulation domain are divided row-wise and distributed equally to a group of processes. We propose two parallel algorithms based on the row-wise partitioning: algorithms with nonblocking send/receive and nonblocking scatter/gather using the NBC library. We compare the performance of both versions by measuring their parallel speedup and efficiency. We also investigate the communication overhead in both versions. Our results show that the nonblocking collective communication can improve the performance of the simulation when the number of processes is large.

Keywords: *Message Passing Interface; Parallel Simulation; Nonblocking collective operations; Scatter and Gather; Communication optimization; High Gradient Magnetic Separation (HGMS)*

1. Introduction

High Gradient Magnetic Separation (HGMS) is a powerful method for the removal of weakly magnetic particles from fluids [8]. In this method, high gradient of magnetic field and magnetic energy density are produced in the separation process to maximize the magnetic force that acts on the magnetic particles. HGMS has been applied in many fields including mineral beneficiation [9], blood separation in biochemistry [10], waste water treatment [11], and food industry [12]. HGMS can be also used in other research and industrial areas that rely on the separation of colloidal particles. The mass transfer process is studied via statistical approach. Sequential simulation of diffusive capture of weakly magnetic nanoparticles in HGMS had been developed and reported in [13].

To investigate the process of mass transfer in a particular situation, the governing equations describing the process dynamics are solved to obtain the distribution behavior of target particles in the considered regions. Equations of a mass transfer process frequently occur as non-linear partial differential equations of second or higher order which are hard to

be solved analytically, hence numerical methods are used. The finite-difference method is a standard approach for that: The distribution configuration of the particles is computed numerically at many discrete points in the considered regions. The increase the number of discrete points, the higher is the accuracy of the results the more time is needed to accomplish the computation. Parallelization is necessary to improve the accuracy of the results and reduce the computing time.

This paper proposes two parallel algorithms for the parallelization of HGMS. Both algorithms use nonblocking communications. The first algorithm using point-to-point model with MPI_Isend/Irecv in a ring communication. The second algorithm uses the nonblocking collectives Igather/Iscatter from libNBC [22]. We compare the efficiency of both communication styles. We found that the first algorithm has a small overhead compared to the second one in our experiments. However, the communication cost of the ring communications increases as the number of processes increases while the communication time in the collective style remains quiet constant. Thus, when computing using more number of processes, the nonblocking collective approach can perform well.

This research paper is organized as followings: next section introduces backgrounds in HGMS. Section 3 and Section 4 present the both nonblocking algorithms respectively. Section 5 presents comparative results of the approaches. Section 6 concludes the work.

2. Backgrounds

Our case study is the mass transfer process of weakly-magnetic nanoparticles during magnetic separation. As a particular application we study the separation of such particles from static water by a magnetic method. The system consists of static water with an assembly of monotype weakly-magnetic nanoparticles as a suspension and a capture center modeled as a long ferromagnetic cylindrical wire of radius. All compositions of the system are considered as linear isotropic homogeneous magnetic media. A uniform magnetic field is applied perpendicular to the wire□s axis. We define the particle volume concentration, denoted by c, as the fraction of particle volume contained in an infinitesimal volume element of the system. According to the geometry of the capture center and the symmetry of the problem, the normalized polar coordinates, as shown in Figure 1, are used. The distance is the radial distance from the wire□s axis in the unit of wire radius, is the angle defined in a plane perpendicular to the wire□s axis. The mass transfer process is studied in normalized time domain which is defined based on real time, particle diffusion coefficient and wire radius. The governing equation of our case study, derived by Davies and Gerber [16, 17], can be expressed as

$$\frac{\partial c}{\partial \tau} = \frac{\partial^2 c}{\partial r_a^2} + \frac{1}{r_a}\frac{\partial c}{\partial r_a} + \frac{1}{r_a^2}\frac{\partial^2 c}{\partial \theta^2} - \frac{G_r c}{r_a} - G_r \frac{\partial c}{\partial r_a} - c\frac{\partial G_r}{\partial r_a} - \frac{G_\theta}{r_a}\frac{\partial c}{\partial \theta} - \frac{c}{r_a}\frac{\partial G_\theta}{\partial \theta} \quad (1)$$

and

$$\left(\frac{\partial c}{\partial \tau}\right)_I = \frac{1}{(r_a)_I^2}\left(\frac{\partial^2 c}{\partial \theta^2}\right)_I - \left(\frac{G_\theta}{r_a}\frac{\partial c}{\partial \theta}\right)_I - \left(\frac{c}{r_a}\frac{\partial G_\theta}{\partial \theta}\right)_I \quad (2)$$

$$+\frac{(G_r c)_{I+1}}{\delta r_a} - \frac{1}{\delta r_a}\left(\frac{\partial c}{\partial r_a}\right)_{I+1}$$

where functions G_r, G_θ and factor G_0 depend on the magnetic properties of the wire, the fluid, the particle, the strength of applied magnetic field and the position in the region [18]. The equation (1) is used for ordinary discrete points whereas equation (2) is used for special discrete points that are adjacent to the wire surface or other impervious surfaces. The governing equations are solved numerically as an initial and boundary value problem, by using the finite-difference method.

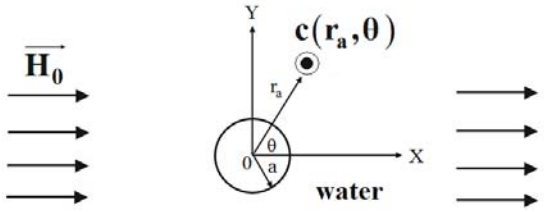

Figure 1. Normalized polar coordinates[15]

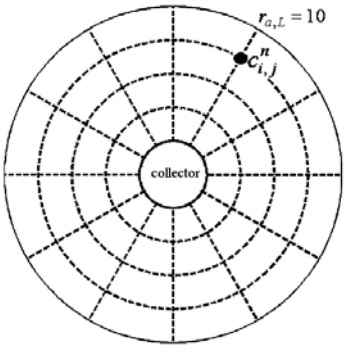

Figure 2. Grid construction.

Firstly, a uniform mesh is constructed in an annular region around the wire as shown in Figure 2. The outer boundary of the region locate at $r_{a,L} = 10$. Then the particle volume concentration at ordinary discrete points $(r_{a,i}, \theta_j)$ and special discrete points $(r_{a,I}, \theta_j)$ at a given normalized time τ_n are computed numerically by using the following equations (3) and (4), respectively. It is seen that the new value of particle concentration at any discrete points depends on the old values of particle concentration at adjacent discrete points. Figure 3(a) shows the representation of Figure 2 in a 2D representation which stores the concentration data computed from Equations 3-4. Figures 3(b) and 3(c) show the data dependency pattern from the representation.

$$
\begin{aligned}
c_{i,j}^{n+1} = & \left[1 - \frac{2(\Delta\tau)}{(\Delta r_a)^2} - \frac{2}{(r_a)_i^2}\left(\frac{\Delta\tau}{(\Delta\theta)^2}\right) - \left(\frac{(G_r)_{i,j}}{(r_a)_i} + \left(\frac{\partial G_r}{\partial r_a}\right)_{i,j} + \frac{1}{(r_a)_i}\left(\frac{\partial G_\theta}{\partial \theta}\right)_{i,j} \right)(\Delta\tau) \right] c_{i,j}^{n} \\
& + \left[\frac{\Delta\tau}{(\Delta r_a)^2} + \left(\frac{1}{2(r_a)_i} - \frac{(G_r)_{i,j}}{2} \right)\left(\frac{\Delta\tau}{\Delta r_a}\right) \right] c_{i+1,j}^{n} \\
& + \left[\frac{\Delta\tau}{(\Delta r_a)^2} - \left(\frac{1}{2(r_a)_i} - \frac{(G_r)_{i,j}}{2} \right)\left(\frac{\Delta\tau}{\Delta r_a}\right) \right] c_{i-1,j}^{n} \\
& + \left[\frac{1}{(r_a)_i^2}\left(\frac{\Delta\tau}{(\Delta\theta)^2}\right) - \frac{(G_\theta)_{i,j}}{2(r_a)_i}\left(\frac{\Delta\tau}{\Delta\theta}\right) \right] c_{i,j+1}^{n} \\
& + \left[\frac{1}{(r_a)_i^2}\left(\frac{\Delta\tau}{(\Delta\theta)^2}\right) + \frac{(G_\theta)_{i,j}}{2(r_a)_i}\left(\frac{\Delta\tau}{\Delta\theta}\right) \right] c_{i,j-1}^{n}
\end{aligned} \tag{3}
$$

$$
\begin{aligned}
c_{I,j}^{n+1} = & \left[1 - \frac{2(\Delta\tau)}{(r_a)_I^2(\Delta\theta)^2} - \frac{(\Delta\tau)}{(r_a)_I}\left(\frac{\partial G_\theta}{\partial \theta}\right)_{I,j} \right] c_{I,j}^{n} \\
& + \left[\frac{(\Delta\tau)}{(r_a)_I^2(\Delta\theta)^2} - \frac{(G_\theta)_{I,j}(\Delta\tau)}{2(r_a)_I(\Delta\theta)} \right] c_{I,j+1}^{n} \\
& + \left[\frac{(\Delta\tau)}{(r_a)_I^2(\Delta\theta)^2} + \frac{(G_\theta)_{I,j}(\Delta\tau)}{2(r_a)_I(\Delta\theta)} \right] c_{I,j-1}^{n} \\
& - \left[\frac{(\Delta\tau)}{(\Delta r_a)^2} + \frac{(G_r)_{I+1,j}(\Delta\tau)}{(\Delta r_a)} \right] c_{I+1,j}^{n} + \left[\frac{(\Delta\tau)}{(\Delta r_a)^2} \right] c_{I+2,j}^{n}
\end{aligned} \tag{4}
$$

3. Nonblocking Point-to-Point Communication Algorithm

In the process of mass transfer simulation, old and new concentration data at every discrete point are stored in two identical two-dimensional arrays. In the parallel simulation based on the distributed memory model, all data are decomposed into equal parts, by using a row-wise partitioning scheme and distributed to a group of processes. Consequently, each process holds its subarray. First, the old data at a given point and the old data of necessary adjacent points are read. Second, the new value of particle concentration is computed. Let the maximum column and row index of the subarray occupied by a process be $i_{\max}$ and $j_{\max}$ respectively, and let I be the column index that contains data at a given special discrete point. The iterative computation in every row of the subarray can be described in general as follows:

1. Search and specify the column index "I" corresponding to the special discrete point.
2. Assign the initial concentration in column $i_{\max}$
3. Iteratively compute the new concentration by using (3), starting from column

$i_{\max}-1$ down to column $I+1$.

4. Compute the new concentration, by using Equation (4), at column of index I.

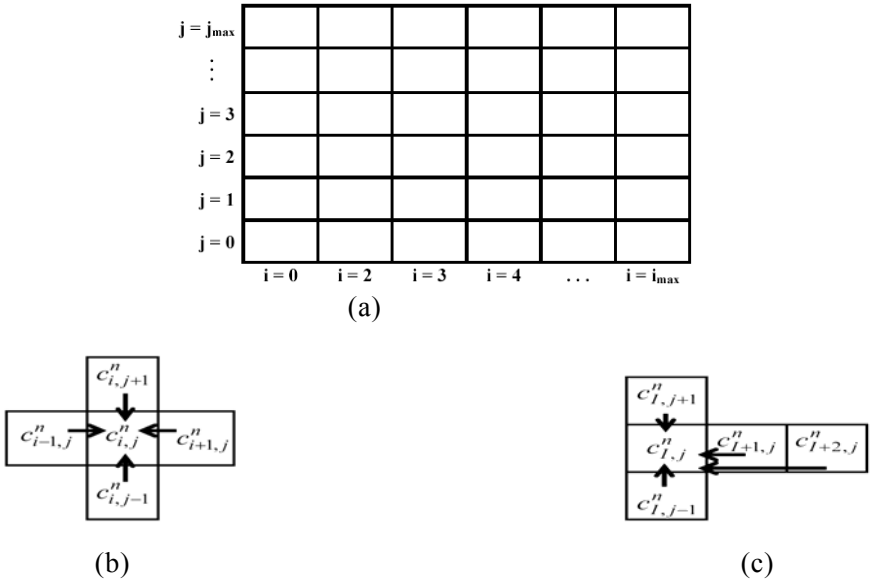

Figure 3. (a) 2D representation. (b)-(c) Patterns of data dependency among discrete points

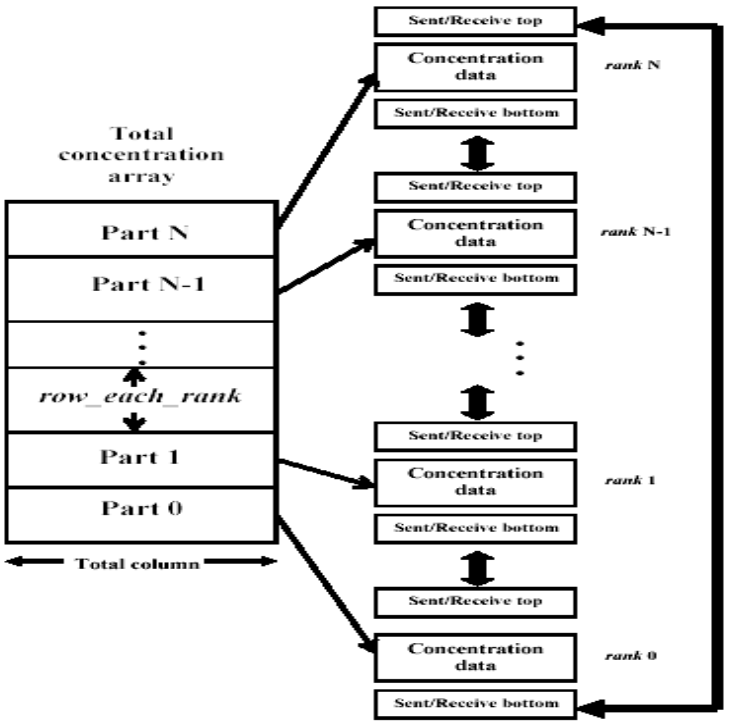

Figure 4. The configuration of data distribution and the ring pattern of data communications between adjacent processes

Finally, the new data replaces the old data. The simulation continues until the final value of normalized time is reached. According to data dependences shown in Figure 3, the computation in row j requires data from adjacent rows of index *j-1* and $j+1$. Consequently, the computation in the first and last rows of the subarray of each process requires the data in the subarrays of the two adjacent processes. On the other hand, each process has responsibility to send its data in the first and last row of its subarray to its two adjacent processes. Moreover, data exchange between the process 0 and the process (N-1) is necessary. An individual process communicates to its neighbors in a ring pattern. Each process uses row arrays called *Sent_bottom* and *Receive_bottom* to exchange data with the lower rank process and uses row arrays *Sent_top* and *Receive_top* to exchange data with the higher rank process.

Figure 4 shows the row-wise data redistribution and the ring pattern of data exchanges between processes. Let 0 be the minimum rank and N the maximum rank of the process. Communication between adjacent processes are performed via non-blocking *MPI_Isend* and *MPI_Wait* communication procedures for process 0 and *MPI_Wait* . and *MPI_Isend* for the remainders. Steps of the parallel algorithm are rearranged using nonblocking communications as follows:

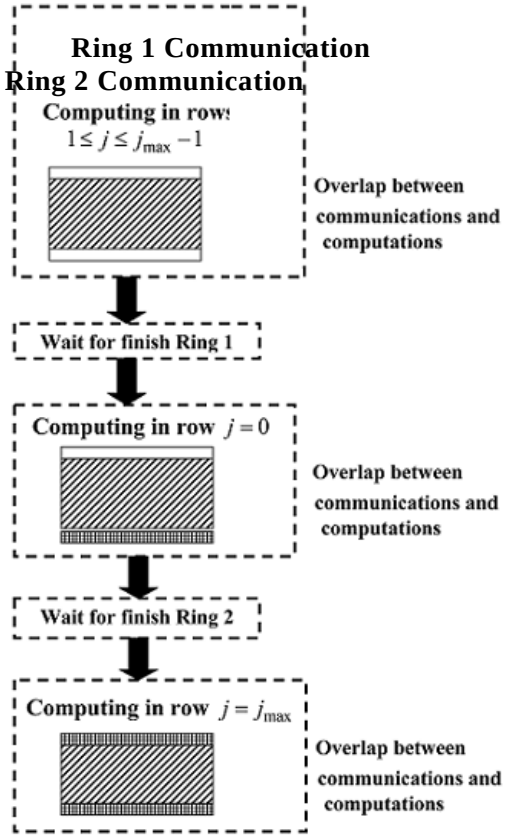

Figure 5. Parallel algorithm using nonblocking communications.

Step 1: Start first ring communication using *MPI_Isend* and *MPI_Irecv*.
Step 2: Start second ring communication using *MPI_Isend* and *MPI_Wait* nonblocking communication procedures.

Step 3: Perform iterative computing of new concentration in all rows of index $1 \le j \le j_{\max} - 1$

Step 4: Perform MPI_Wait operation to ensure the available of data necessary for Iterative computing in the row $j = 0$.

Step 5: Perform iterative computing of new concentration in the row $j = 0$.

Step 6: Perform MPI_Wait operation to ensure the available of data necessary for Iterative computing in the row

Step 7: Perform iterative computing of new concentration in the row index $j = j_{\max}$.

Then Steps 1-5 are repeated until convergence is achieved.

The idea of using non-blocking communication is to overlap communication with computation. In the original algorithm, once the data is needed, the communication is required. In this algorithm, the communication in Step 1 in the previous algorithm is moved to Step 3 to hide the communication latency. Figure 5 shows the scheme of parallel simulation by using non-blocking communication. The nonblocking communication is done first and then computations start. When data is required, MPI_Wait() is performed to ensure that the needed data arrived. The computation of rows $1 \le j \le j_{\max} - 1$ is used to overlap with the communication. Here, it can be observed that the overlapped computation depends directly on the size of domain for each process.

The main title (on the first page) should begin 1 3/16 inches (7 picas) from the top edge of the page, centered, and in Times New Roman 14-point, boldface type. Capitalize the first letter of nouns, pronouns, verbs, adjectives, and adverbs; do not capitalize articles, coordinate conjunctions, or prepositions (unless the title begins with such a word). Please initially capitalize only the first word in other titles, including section titles and first, second, and third-order headings (for example, “Titles and headings” — as in these guidelines). Leave two blank lines after the title.

4. Nonblocking Collective Algorithm

The test of communication algorithms of both blocking ring and blocking scatter/gather in our recent work [23] indicated that communication algorithm using scatter/gather provides a better efficiency than the blocking ring approach. Therefore, we hypothesize that the non-blocking scatter/gather would also be better than non-blocking ring as well. We then implement our new communication algorithm using non-blocking collective using libNBC [22].

To compare with the ring communication algorithm, the algorithm is devised in the same manner. Using the collective style, the root process, 0, collects the necessary updated data at the end of each iteration using gather and distribute the updated rows to each relevant process using scatter. We use the *buffers Scatter_top, Scatter_bottom*, to hold data scattered from the root for each process. Also, the buffer *Gather_top, Gather_bottom*, is used by the root to hold data gathered from the others. After the root gathers all updated rows from other processes, it needs to update and rearrange the concentration data before scattering in the next round. We also include the computation in the root process as well. The iterative process is changed as follows:

Step 1: Root process, *p*=0, scatters the arranged data in its *scatter_bottom* and *scatter_top* buffers to *receive_top* and *receive_bottom* buffers of every process, respectively using NBC_Iscatter().

Step 2: Root process, rank *p* = 0, gathers data in the *sent_bottom* and *sent_top* buffers of each process into its *gather_bottom* and *gather_top* buffer respectively using NBC_Igather().

Step 3: All processes, including the root, perform iterative computation, from the first row+1 to the last row -1 of its subarray,

Step 4: All processes performs NBC_Wait() for *receive_top* and *receive_bottom* respectively. Then they compute the first row and the last rows accordingly.

Step 5: All processes copy data in the new concentration subarray into the old concentration subarray.

Step 6: All processes put data in the first and last row into its *sent_bottom* and *sent_top* buffers, respectively.

Step 7: Root process performs NBC_Wait() for *gather_bottom, gather_top.*

Step 8: Root rearranges data in its *gather_bottom* and *gather_top* buffers and then put the arranged data in the *scatter_bottom* and *scatter_top* buffers, respectively. The pattern of rearrangement is as follows.

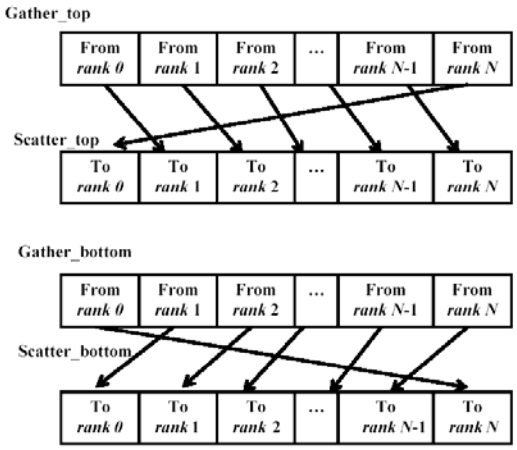

Figure 6. The scheme of the scattering operation

From the algorithm, it is seen that Step 1 and Step 2 performs the nonblocking communications using NBC scatter/gather (similar to Figure 5). The overlapped computations are in step 3. The number of rows implies the amount of overlapped computation. In our case, if the number processes are large the number of rows is reduced, the communications are overlapped less.

5. Experimental Results

In our experiments, we simulate mass transfer of paramagnetic $Mn_2P_2O_7$ particle of radius b_p = 12 nm. dispersed in static water. The effective magnetic susceptibility of the system (water + $Mn_2P_2O_7$ particle) is $\chi = +4.73 \times 10^{-3}$[17]. The ferromagnetic wire is homogeneously saturated magnetized perpendicular to its axis by a uniform external magnetic field $H_0 = 1 \times 10^6$ A/m which is perpendicular to the wire s axis. The factors

G_0 = -16.62 and K_W = 0.80. The initial concentration at every discrete point is equal to C_0 = 0.0010 and the saturation concentration is C_{sat} = 0.10 [20]. Grid steps are Δr_a = 0.010, $\Delta\theta$ = 0.10 and $\Delta\tau$ = 0.0000010. Hence there are in total 3,600 rows and 901 columns in the whole computational domain.

We perform the experiments on a 32 nodes, totally 64 cores Linux cluster, with a Gigabit Ethernet interconnection at Louisiana Technology University, USA. In the cluster, each core is Intel Xeon 2.8GHz with 512 MB RAM. The cluster runs LAM-MPI 7.1 and on Gigabit Ethernet network.

The speedup of parallel simulation is defined as $S_p = t_1 / t_p$, where t_1 is the average sequential simulation time and t_p is the average parallel simulation time on p processes. The parallel efficiency is computed by $E_p = S_p / p$ [21].

Table 1. Speedup results

Number of Processes	NBC Ring speedup	NBC Collective speedup
4	4.091502	4.312826
6	5.985637	6.235232
8	7.950379	8.183956
10	8.753601	9.445269
12	11.63329	11.404066
16	14.89231	14.111646
20	17.87601	16.600328

Table 2. Efficiency results

Number of Processes	NBC Ring speedup	NBC Collective speedup
4	1.02287556	1.07820639
6	0.99760621	1.03920526
8	0.99379741	1.02299452
10	0.87536013	0.94452691
12	0.9694407	0.95033881
16	0.93076926	0.88197789
20	0.89380027	0.83001640

Table 1 and Table 2 show data of speedup and efficiency of nonblocking ring and nonblocking collective algorithms, respectively, for various number of processes cases. In other words, we divide the computation by rowwise according to the given number of processes. From the data, it is surprising that the nonblocking collective algorithm performs about the same as the nonblocking ring approach and worse in some case, unlike in the original collective algorithm which performs better than the traditional ring approach[23].

From both tables, we discovered that speedup of the algorithm is always higher than nonblocking ring algorithm when the number of processes is low (less than 12) but when the processes become higher (greater than 12) nonblocking collective performance will become lower than ring algorithm. This is because incurring overheads of libNBC is more than that of the original MPI and when the overlapping

computation is less as the number of processes grows (see the cases for 12 or more processes), the overheads cannot be hidden totally.

Table 3. Average communication time comparison

Number of Processes	Nonbloking ring	Nonblocking collective	% Diff
4	342.8	606.3	-76.89
6	304.5	620.3	-103.72
8	283.7	606.5	-113.80
10	315.4	827.0	-162.24
12	350.6	665.0	-89.69
16	423.5	666.1	-57.28
20	534.3	685.4	-28.27

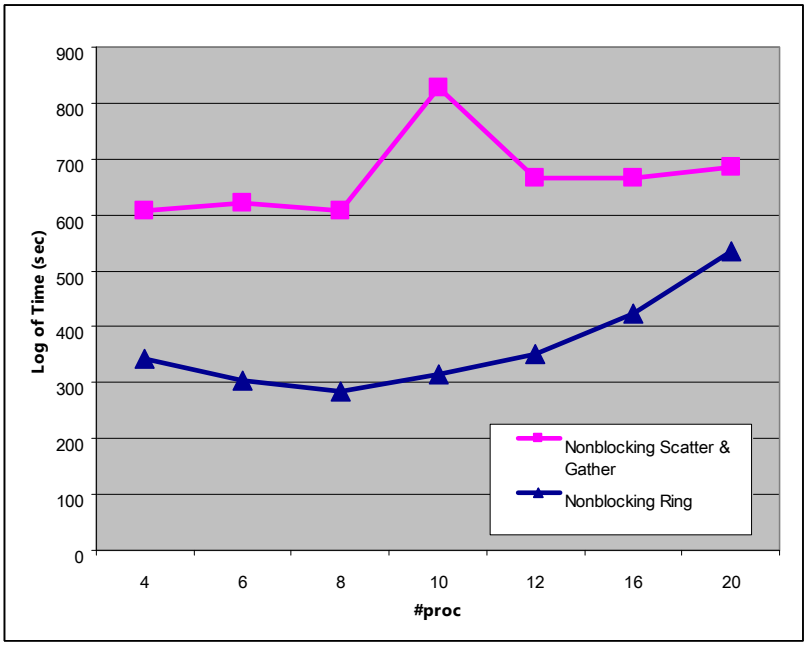

Figure 9. Communication time comparison graph

We further inspect the communication overheads of the libNBC approach as in Table 3. The communication time shown here is the total communication time for all NBC_Iscatter, NBC_Igather, and NBC_wait for the collective case and is the total time for all MPI_Isend, MPI_Irecv, MPI_Wait for the nonblocking ring case. Figure 9 plots the comparison of overall times between two schemes. It is seen that the libNBC approach produces more overheads for each case. However, it is noticed that the overheads grow very slowly when the number of processes increases. On the contrary, the overheads grow faster for the nonblocking ring case. The difference of communication overheads between the two approaches are reduced as the number of processes increases (see Column '%Diff' in Table 3). Also, in Figure 10, we analyze the time spent for each libNBC call. It is seen that the time spent most are on wait3, wait1, wait2, and wait4 calls accordingly. For wait3 call, it is the NBC_Wait for the

first NBC_Igather. For wait1 call, it is the average waiting time for first NBC_Iscatter for all processes, and for wait2 call, as well as wait4 call, they are the average waiting time for the second NBC_Igather, NBC_Iscatter respectively. We can see that for wait4 call, it is the time that the root process requires to gather all updated top rows from other processes. For wait1 call, every process waits for the root to scatter the updated top rows in the new iteration.

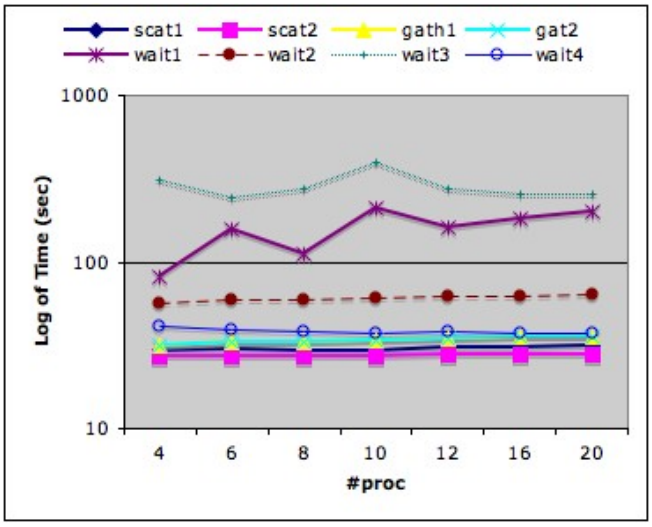

Figure 10 Details communication time of the nonblocking collective case

6. Conclusion

We propose the two nonblocking parallel algorithms for High Gradient Magnetic Separation (HGMS) of nanoparticles. In both scheme, we distribute the domain of computation equally by row-wise. The first algorithm is based on MPI_Isend/Irecev ring style communication where the second algorithm is based on NBC_Iscatter/Igather collective style communication. The results show that in the tested environment, the nonblocking ring algorithm performs better. This is because the overhead incurred by the nonblocking MPI calls are less for all the test cases and can be hidden totally in the overlapped computation. For the nonblocking collective algorithm using libNBC, it has more communication overheads and needs more overlapped computation time to hide them.

We found that communication time of the nonblocking collective algorithm is always lower than nonblocking ring algorithm when the number of processes is small. When the number of processes becomes larger higher, the communication time of the collective style only slightly increases while the communication time of the nonblocking ring increase at a faster rate. We predict that if we increase more number of processes and the work size, the collective communication style will perform better than the nonblocking ring approach. This exploration of communication style and domain partitioning will be investigated in the next paper.

Acknowledgement

We would like to thank Assoc. Prof. Dr. Box Leangsuksun for allowing us to use the cluster for testing the experiment. We also thank Torsten Hoefler from Indiana University for helpful comments regarding the use of LibNBC.

References

[1] Balay, S., Buschelman K., Jkhout V., Gropp, W., Kaushik, D., Knepley, M., McInnes, L. C., Smith, B., Zhang, H.: PETSc Users Manual, Technical Report. ANL-95/11 - Revision 2.3.2.
[2] The ScaLAPACK Project, http://www.netlib.org/scalapack/index.html
[3] Shu, J. W., Lu,Q., Wong, W. O., Huang, H.: Parallelization strategies for Monte Carlo simulations of thin film deposition. Compu Phys Commun. 144, 34--45 (2002)
[4] Pure QCD Monte Carlo simulation code with MPI, http://insam.sci.hiroshima-u.ac.jp/QCDMPI/
[5] Roy, S., Jin, R. Y., Chaudhary, V., Hase, W. L.: Parallel molecular dynamics simulations of alkane/hydroxylated α -aluminum oxide interfaces. Comput. Phys. Commun. 128, pp.210--218 (2000).
[6] Xu, J., Ostroumov, P. N., Nolen, J.: A parallel 3D Poisson solver for space charge simulation in cylindrical coordinates. Comput. Phys. Commun. 178, 290--300 (2008)
[7] Wu, J. S., Hsu, K. H., Li, F. L., Hung, C. T., Jou, S. Y.: Development of a parallelized 3D electrostatic PIC-FEM code and its applications. Comput. Phys. Commun. 177, 98--101 (2007)
[8] Gerber, R., Birss, R. R.: High Gradient Magnetic Separation. John Wiley& Sons, New York (1983).
[9] Kelland, D. R.: High gradient magnetic separation applied to mineral beneficiation. IEEE Trans. Magn. 9, 307--310 (1973)
[10] Takayasu, M., Kelland, D. R., Minervini, J. V.: Continuous magnetic separation of blood components from whole blood. IEEE Trans. Appl. Supercond. 10, 927--930 (2000)
[11] Delatour, C.: Magnetic separation in water pollution control. IEEE Trans. Magn. 9, 314--316 (1973)
[12] Safarik, I., Sabatkova, Z., Tokar, O., Safarikova, M.: Magnetic Cation Exchange Isolation of Lysozyme from Native Hen Egg White. Food Tech. Biotechnol. 45, 355--359 (2007)
[13] Hournkumnuard, K., Natenapit, M.: Diffusive Capture of Magnetic Particles by an Assemblage of Random Cylindrical Collectors. Sep. Sci. Technol. 43, 3448--3460 (2008)
[14] Bleaney, B. I., Bleaney, B.: Electricity and Magnetism. Claredon Press, Oxford (1965)
[15] Arfken, G. B., Weber, H. J.: Mathematical Method for Physicists. Academic Press, California (1995)
[16] Gerber, R., Takayasu, M., Friedlaender, F. J.: Generalization of HGMS theory: The capture of ultra-fine particles. IEEE Trans. Magn. 19, 2115--2117 (1983)
[17] Davies, L. P., Gerber, R.: 2-D simulation of ultra-fine particle capture by a single-wire magnetic collector. IEEE Trans Magn. 26, 1867--1869 (1990)
[18] Hournkumnuard, K.: Diffusive Capture of Magnetic Particles by an Assemblage of Random Cylindrical Collectors. MSc. Thesis, Chulalongkorn University (2004)
[19] Mitchell, A. R., Griffiths, D. F.: The Finite Difference Method in Partial Differential Equations. John Wiley& Sons, New York (1980)
[20] Gerber, R.: Magnetic filtration of ultra-fine particles. IEEE Trans. Magn. 20, 1159--1164 (1984)
[21] Quinn, M. J.: Parallel Programming in C with MPI and OpenmP. McGraw-Hill, New York (2003)
[22] Hoefler, T., Squyres, J. M., Bosilca, G., Fagg, G., Lumsdaine, A., Rehm, W.: Non-Blocking Collective Operations for MPI-2. (2006)
[23] Phongpensri, C., Gorlatch, S., Hoefler, T.: A Parallel Simulation of Mass Transfer in High Gradient Magnetic Separation of Nanoparticles Using MPI Collective Operations. Proceedings of NCSEC 2009, Bangkok, Thailand (2009).