Balancing Communication and Execution Technique for Parallelized Sparse Matrix-Vector Multiplication

Seiji Fujino and Takeshi Nanri
Research Institute for Infoemation Technology
Kyushu University
Fukuoka, Japan
Email: {fujino, nanri}@cc.kyushu-u.ac.jp

Ken'itiro Kusaba
Mitsubishi Electric Co. Ldt.
Chiyoda-ku, Marunouti
Tokyo, Japan

***Abstract*—This paper proposes a runtime optimization technique for load balancing parallelized SpMxV (sparse matrix-vector multiplication) with consideration of cost for both communication and execution time. In state-of-the-art iterative methods that call SpMxV repeatedly during iterations, this optimization utilizes the estimated time of communication and the measured time of execution for adjustment of balance of load among processes. Through numerical experiments of an iterative solver for linear systems, it will make it clear that the proposed optimization technique outperforms compared with the conventional technique.**

Keywords*-Sparse Matrix-Vector Multiplication; Load-Balancing; Bi_IDR(s) method.*

I. Introduction

In many scientific computations, SpMxV (sparse matrix vector multiplication) [5] [8] plays a crucial role. For example, for the purpose of solution of linear systems of equations which stem from FEM (Finite Element Method) analysis, SpMxV is often repeatedly executed and shares major part of the entire executing time. As a strategy for accelerating this operation, parallelization of SpMxV has been studied in many researches. The most popular way for parallelizing SpMxV is to attach calculation to processes by row of the target matrix. Since most of the entries in sparse matrices are zero, they are compressed so that only nonzero entries need to be stored and calculated. The execution time of each row depends heavily on the pattern of nonzero entries of matrix. This causes difficulty in balancing loads among processes in parallelized SpMxV.

This paper proposes a technique for load-balancing MPI-based SpMxV operation. There are a lot of researches for parallelizing SpMxV on distributed memory systems. Graph-partitioning based decomposition algorithms [1] are major approach among them. Due to the complexity of these algorithms, they are generally applied statically. On the other hand, to achieve better load-balance in MPI-based SpMxV, Lee *et* a*l*. proposed a runtime technique called NRET (Normalized Row-Execution-Time-based iteration-to-process mapping) to adjust the mapping of rows to processes repeatedly [4]. In NRET, averages of measured time for calculation on each process are used to estimate the time of computing each row. Then, the mapping is adjusted according to this estimation so that it achieves better load-balance. However, since the technique does not consider the time for communication, the total loads among processes still remains to be unbalanced.

To solve this issue of unbalancing, we proposes Balancing Communication and Execution Technique (abbreviated as Balancing-CET). In a program that invokes SpMxV repeatedly, the technique adjusts mappings of rows to processes at each invocations in the beginning several iterations. The adjustments are done according to the estimation of the time for computation and communication on each process. The execution time is estimated from the measurement in the previous iteration, while the communication time is estimated by a linear performance model with the amount of data to be sent and received by each process. By repeating adjustment, this technique becomes close to the optimal distribution. This paper verifies effectiveness of Balancing-CET through numerical experiments of an iterative solver for linear systems.

This paper is organized as follows. In section 2 we make an overview of runtime load-balancing technology. In section 3, we describe costs for communication and execution in SpMxV. In section 4, through numerical experiments, we verify effectiveness of the proposed Balancing-CET. In section 5, we will draw some conclusions.

II. Overview of Runtime Load-Balancing Technology

The proposed Balancing-CET assumes a kind of program in which SpMxV is invoked many times with the same sparse matrix. In addition, it assumes that the program uses the result vector of a SpMxV as the vector to be multiplied to the matrix at the next SpMxV. These assumptions can apply to many scientific programs. Figure 1 presents the parallelized version of SpMxV program. Since this program is parallelized in a row-based manner, the result vector of SpMxV is distributed among processes.

Therefore, to use the result vector as the vector to be multiplied to the matrix in the next SpMxV, communication among processes must be done. The simplest but inefficient way of such communication is to invoke `MPI_Allgather`. Instead, this paper assumes that the program uses more efficient way proposed by Lee *et* a*l*. [4]. It reduces the amount of data to be transferred by letting each process send a minimum block that contains necessary data for each target process to calculate its part of SpMxV. Figure 2 sketches the flow of the communications.

After each execution of SpMxV, Balancing-CET checks if the load-balance is fine enough by comparing the execution time of each process. If the ratio of the difference between the maximum time and the minimum time among all processes is larger than a specified threshold, the algorithm invokes adjustment of distribution.

In the adjustment, Balancing-CET first calculates the average time of the total cost of previous SpMxV among all processes. Then it uses this time tCT as the target time for mapping rows to processes. Mappings are done row by row. As mapping one row to a process, estimated cost of communication and execution is added to the predicted execution time of the process. If the predicted execution time exceeds tCT, Balancing-CET changes the target process to the next one.

III. Costs for Communication and Execution in SpMxV

To reduce the overhead of adjustment, estimation of costs for communication and execution on each row are done in a simple way. Execution cost of each row is estimated by using the same method as NRET. In this method, the computation cost of *i*th row is estimated by the average time of calculating rows at the process which calculated *i*th row in the previous SpMxV.

Communication cost, on the other hand, is estimated by using a linear performance model of communication, $\alpha \times m + \beta$. In the model, α, β and m stands for the time for transferring one unit data, the latency of communication and the amount of data to be transferred, respectively. α and β are calculated at the beginning of a program by using the results of measuring time for some pingpong communication between processes. On the other hand, m is calculated for each row at the time the row is applied to a process. In applying a row to a process, the amount of data that should be sent and received is determined for each of

```
for i = start(myrank) to start(myrank + 1) - 1
   temp = 0.0
   for j = rowptr(i) to rowptr(i+1) - 1
      temp = temp + val(j) * x(colind(j))
end for
```

Figure 1. An Example of Parallelized SpMxV program.

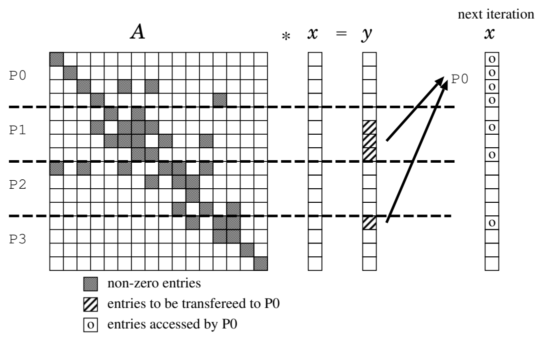


Figure 2. A Sketch of Communications for Transferring Minimum Block of Data Necessary for the Next SpMxV.

other processes by using location of nonzero entries of the matrix. Figure 3 depicts how to derive the receiver processes of an entry. From these information, the additional costs for sending and receiving with this new mapping are calculated.

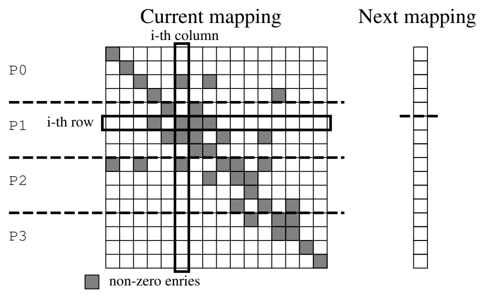


Figure 3. A Diagram of Determining the Target Process of Communications for an Entry.

IV. Numerical Experiments

To examine the effect of Balancing-CET, some numerical experiments have been done. The threshold for determining load-balance in Balancing-CET is set to 5%. In addition, the number that this technology is invoked is limited by 20 times. The experimental platform is a cluster of 16 PCs connected by InfiniBand. Each PC consists of 4 cores of Intel Xeon 3.0GHz and 8GB of RAM and runs with RedHat Enterprise Linux 3. The program is written by Fortran90 and MPI. The compiler and the MPI library are products of Fujitsu Corp. Each process of MPI programs is attached to one CPU core.

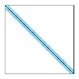
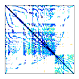
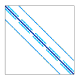
A. Effect of Balancing-CET on SpMxV

The performance of a program that repeats 1000 times of SpMxV is measured to examine the effects of NRET and Balancing-CET. After each SpMxV, communications are performed to use the result vector of it in the next SpMxV. As for test matrices, three matrices selected from Florida Sparse Matrix Collection [2] are used. Table I lists specifications of test matrices with images that show their patterns of nonzero entries. As the images show, each matrix has different characteristics. In matrix matrix_9, nonzero entries are mainly located on a diagonal band with some exceptions on the right-most column. On the other hand, they are distributed throughout in matrix ecl32, and they are confined to a diagonal band in matrix xenon1.

Table I
INPUT MATRICES.

name	dimension	total nonzero entries	ave. nonzero entries
matrix9	103,430	2,121,550	20.51
ecl32	51,993	380,415	7.32
xenon1	48,600	1,181,120	24.30

Table II
SPECIFICATION OF TEST MATRICES.

pattern of nonzero entries	name	dimension	total nonzero entries	ave. nonzero entries
	matrix_9	103,430	2,121,550	20.51
	ecl32	51,993	380,415	7.32
	xenon1	48,600	1,181,120	24.30

Tables III - V show communication time and total execution time for matrices of matrix_9, ecl32 and xenon1. In these Tables, "No Opt.", "NRET" and "Balancing-CET" mean the results without any load-balancing, with NRET and with Balancing-CET, respectively. "Procs" is the number of processes. "Comm" is the time for communication and "Total" is the total execution time. Hereafter time in Tables is measured in seconds.

Table III
COMMUNICATION TIME AND TOTAL EXECUTION TIME FOR MATRIX MATRIX_9.

Procs	No Opt.		NRET		Balancing-CET	
	Comm	Total	Comm	Total	Comm	Total
1	0.725	11.037	0.002	9.741	0.001	9.708
2	1.483	9.377	0.758	8.947	0.756	8.928
4	2.265	7.294	0.746	5.370	0.611	5.297
8	2.701	5.294	0.945	2.982	0.678	2.669
16	2.960	4.337	1.171	1.761	0.412	1.190
32	3.187	3.962	0.764	1.033	0.388	0.796

As shown in these Tables, NRET and Balancing-CET reduced the execution time significantly. With matrices matrix_9 and ecl32, the effect of Balancing-CET is better than

Table IV
COMMUNICATION TIME AND TOTAL EXECUTION TIME FOR MATRIX ECL32.

Procs	No Opt.		NRET		Balancing-CET	
	Comm	Total	Comm	Total	Comm	Total
1	0.320	2.423	0.001	1.912	0.001	1.934
2	0.726	2.001	0.300	1.380	0.262	1.351
4	1.036	1.843	0.474	1.111	0.467	1.110
8	1.181	1.554	0.812	1.109	0.800	1.067
16	1.135	1.334	1.074	1.231	0.956	1.143
32	1.185	1.332	1.143	1.235	0.960	1.101

Table V
COMMUNICATION TIME AND TOTAL EXECUTION TIME FOR MATRIX XENON1.

Procs	No Opt.		NRET		Balancing-CET	
	Comm	Total	Comm	Total	Comm	Total
1	0.347	7.436	0.002	7.013	0.002	7.016
2	0.719	5.237	0.177	4.573	0.156	4.555
4	0.969	3.573	0.262	2.620	0.260	2.628
8	1.089	2.220	0.298	0.970	0.301	1.024
16	1.091	1.532	0.197	0.412	0.179	0.441
32	1.126	1.345	0.139	0.274	0.156	0.355

NRET in most of the cases. On these matrices, it can be concluded that the considerations of communication costs in load-balancing worked well. With matrix xenon1, on the other hand, NRET showed better effects than Balancing-CET. Since nonzero entries of matrix xenon1 are confined in a diagonal band, imbalance of load for communication and execution is not significant. Therefore, both techniques could achieve sufficiently fine load-balance. In this case, the overhead for estimating costs of communications caused longer execution time than NRET.

To examine the detailed behavior of NRET and Balancing-CET, the number of rows attached to each process is studied. Figure 4 plots the case of matrix matrix_9 with 16 processes. "Opt." is the number of rows attached to each process with Balancing-CET. Since this matrix has nonzero entries on the right-most column, the process P15 needs to invoke larger number of send operations than others. Therefore, Balancing-CET attached fewer number of rows to P15. In addition to that, it also reduced the number of rows on P3, P4, P7, P8 and P11 because they performed inter-node communications that are slower than intra-node communications.

B. *Effect on an Iterative Solver for Linear Systems*

To see the effects of Balancing-CET on more realistic situations, parallelized Bi_IDR(s) method [3] [6] [7] for solving linear systems has been examined. This solver is an enhanced version of IDR(s) method to achieve better stability with lower computation cost. In this program, not only SpMxV but some other vector operations, such as summation and dot-product, are performed repeatedly. Those operations are also parallelized into processes. In parallelized dot-product, MPI_Allreduce is invoked to calculate

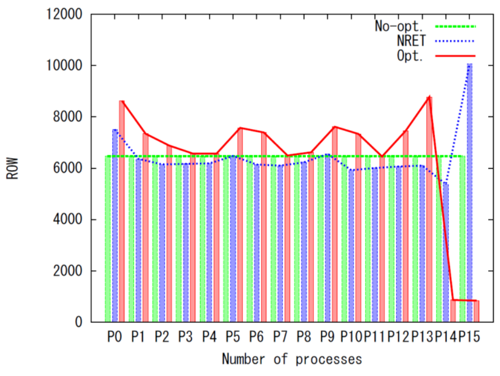


Figure 4. Fluctuation of Number of Rows Attached on each Process for matrix matrix_9.

the total sum among all processes.

Tables VI-VII tabulate total time in seconds, ratio and average of Bi_IDR(s) method with parameter s= 4 and 8 for matrix matrix_9. This program stops if the ratio of the 2-norm of the residual and of the initial residual is less than 10^{-12}. "Iter." and "Total" mean the number of iterations and the total time to satisfy this convergence test, respectively. "Ratio" as shown in Balancing-CET column means the time ratio of Balancing-CET to that of NRET. "TRR" means True Relative Residual for the approximate solution $\boldsymbol{x}_{k+1}$ as $||\boldsymbol{b}-A\boldsymbol{x}_{k+1}||_2/||\boldsymbol{b}-A\boldsymbol{x}_0||_2$.

As shown in Table VI, a mark "$_{**}$" indicates that measurement of time did not succeed. In the case with s=4 and number of processes is 8 of NRET in Table VI, TRR is very poor as "(-9.43)". However TRR of results of Balancing-CET is always sound. Moreover, in Table VII, the performance with Balancing-CET is more efficient than that with NRET as number of processes becomes larger. The performance degradation of Balancing-CET in the smaller number of processes is caused by the overhead for estimating communication cost. TRR of results of Balancing-CET is fairly good compared with that of results of NRET. Accordingly we may expect higher performance of Balancing-CET when we will solve large-scale problems.

Table VI
TOTAL TIME, RATIO AND AVERAGE OF BI_IDR(s) METHOD WITH $s = 4$ FOR MATRIX_9.

procs	No Opt.		NRET			Balancing-CET			
	Iter.	Total	Iter.	Total	TRR	Iter.	Total	Ratio	TRR
serial	5248	73.975	-	-	-	-	-	-	-
1	5248	146.126	5248	132.453	-11.80	5248	118.487	0.895	-11.80
2	5134	88.591	5192	80.878	-11.88	5593	79.072	0.978	-11.57
4	5421	72.605	5654	44.107	-11.55	5367	42.907	0.973	-11.88
8	5467	82.283	5334	**	-11.84	5417	23.175	-	-11.95
16	5887	91.571	5366	15.006	-11.37	5370	12.255	**0.817**	-11.71
32	5493	111.989	5305	12.723	(-9.43)	5308	11.240	0.883	-11.94
ave.	5442	-	5350	-	-11.31	5384	-	-	**-11.81**

Table VII
TOTAL TIME, RATIO AND AVERAGE OF BI_IDR(s) METHOD WITH $s = 8$ FOR MATRIX_9.

procs	No Opt.		NRET			Balancing-CET			
	Iter.	Total	Iter.	Total	TRR	Iter.	Total	Ratio	TRR
serial	4166	73.975	-	-	-	-	-	-	-
1	4166	125.880	4166	116.508	-10.31	4166	144.246	1.238	-10.31
2	4261	90.098	4350	80.769	-10.14	4192	83.897	1.039	-11.49
4	4134	62.663	4441	41.725	-10.36	4142	41.507	0.995	-10.50
8	4087	65.310	4285	23.563	-10.63	4249	22.364	0.949	-10.88
16	4098	65.551	4280	16.970	-10.85	4272	15.844	0.934	-10.49
32	4195	85.406	4256	15.389	-11.38	4145	12.299	**0.799**	-11.25
ave.	4157	-	4296	-	-10.61	**4194**	-	-	**-10.82**

V. CONCLUSIONS

This paper proposed Balancing-CET for load-balancing in SpMxV. It estimates costs of communication and execution time for achieving better load-balance. Through numerical experiments are simple, numerical results demonstrated effects of Balancing-CET.

REFERENCES

[1] UV. Catalyurek and C. Aykanat: Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication, IEEE Transactions on Parallel and Distributed systems, vol. 10, no. 7, pp. 673-693, 1999.

[2] Sparse Matrix Collection: http://www.cise.ufl.edu/research /sparse/matrices/index.html

[3] S. Fujino, P. Sonneveld, Y. Onoue and M.B. van Gijzen: A Proposal of IDR(s)-SOR Method, Transaction of JSIAM, vol. 20, no. 4, pp. 289-308, 2010.

[4] S. Lee and R. Eigenmann: Adaptive runtime tuning of parallel sparse matrix-vector multiplication on distributed memory systems, Proceedings of the 22nd annual International Conference on Supercomputing 2008, pp. 195-204, June, 2008.

[5] Y. Saad: Iterative Methods for Sparse Linear Systems, 2nd ed., SIAM, Philadelphia, 2003.

[6] P. Sonneveld, M.B. van Gijzen: IDR(s): a family of simple and fast algorithms for solving large nonsymmetric linear systems, SIAM J. Sci. Comput., vol. 31, pp. 1035-1062, 2008.

[7] P. Sonneveld, M.B. van Gijzen: An elegant IDR(s) variant that efficiently exploits bi-orthogonality properties, Depart. of Applied Math. Anal., TR08-21(2008), Delft University of Technology.

[8] H.A. van der Vorst: Iterative Krylov Preconditionings for Large Linear Systems, Cambridge Univ. Press, Cambridge, 2003.