Reordering of sparse matrix for speeding up computations on multi-core processors

Eun-Jin Im
School of Computer Science,
College of Electrical Engineering and Computer Science
Kookmin University
861-1 Jeongneung-dong Songbuk-gu
Seoul 136-702 Korea
ejim@kookmin.ac.kr

Abstract. The advent of multi-core processors opened a new opportunity and challenge for high-performance computing. The added level of parallelism can be fully exploited through an effective optimization of target operation and data structure. In this paper, we explore reordering techniques for sparse matrix computations, aiming to improve cache performance as a result of localization of memory accesses. We use a row and column renumbering scheme such as Reverse Cuthill-McKee and we also apply a graph partitioning algorithm to reorder matrix columns and rows.

Keywords: Sparse Matrix, Optimization, Reordering, Multi-core

1 Introduction

The widespread use of multi-core processors promise a greater performance, and the prospect of increasing number of cores in a single chip is being realized in a breadth-taking speed. However, the promised performance is not achieved at no cost. The multi-core architecture allows a hierarchy of parallelism where main memory is shared between cores in the same chip and a number of such multi-core processors are connected through a high-speed network. For a single application, in order to fully achieve maximum performance with such hierarchically parallel platform, the role of performance tuning is critical.
The sparse matrix operations are basic building blocks in a large group of high performance computing applications. As in many other operations, the performance bottleneck of sparse matrix operations is a limited memory bandwidth. The author has conducted extensive research on register blocking and cache blocking methods for sparse matrix-vector multiplications in previous research [1],[2]. In this paper, the author explores reordering techniques for sparse matrix-vector multiplications, aiming to improve cache performance as a result of localization of memory access within each core.

2 Preliminary Result

In this research, we try to reduce bandwidth of sparse matrices by reordering the rows and columns of a sparse matrix. We use a row and column renumbering scheme such as Reverse Cuthill-McKee[3] and we also apply a graph partitioning algorithm, such as multilevel partitioning[4], to reorder matrix columns and rows. The table below summarize our preliminary result of matrix reordering. A suite of sparse matrices are reordered with Reverse Cuthill-McKee algorithm and the maximum and median column bandwidths of sparse matrices in the suite are measured and compared before and after reordering.

Table 1. Bandwidth reduction of sparse matrices after reordering.

matrix	max. bandwidth before reordering	max. bandwidth after reordering	bandwidth reduction ratio (max.)	median bandwidth before reordering	median bandwidth after reordering	bandwidth reduction ratio (median)
3dtube.psa	5144	4721	1.09	4157	3608	1.15
494_bus.rsa	442	158	2.80	151	81	1.86
af23560.rua	608	620	0.98	608	608	1.00
bayer02.rua	13874	451	30.76	118	188	0.63
bayer10.rua	13303	1041	12.78	117	372	0.31
bcspwr01.psa	38	15	2.53	13	9	1.44
bcsstk35.rsa	20762	4802	4.32	215	2977	0.07
coater2.rua	1044	953	1.10	290	296	0.98
crystk02.psa	920	1550	0.59	920	854	1.08
crystk03.rsa	1142	1964	0.58	1142	1064	1.07
ct20stif.psa	49350	6842	7.21	1481	3543	0.42
finan512.psa	74724	2525	29.59	29957	2240	13.37
gupta1.psa	21513	21013	1.02	14075	16150	0.87
lhr10.rua	8115	642	12.64	4363	231	18.89
lp_cre_b.rra	76549	60639	1.26	41725	29224	1.43
lp_cre_d.rra	73480	61970	1.19	19765	20764	0.95
lp_fit2p.rra	13524	13512	1.00	6026	6798	0.89
lp_nug20.rra	72220	54972	1.31	46120	38858	1.19
nasasrb.rsa	1733	2072	0.84	569	1109	0.51
onetone2.rua	7664	5607	1.37	357	4019	0.09
pwt.psa	35634	652	54.65	51	246	0.21
rdist1.rua	123	204	0.60	39	102	0.38
rim.rua	22513	934	24.10	186	449	0.41
venkat01.rua	60623	4834	12.54	391	3278	0.12
vibrobox.rsa	12290	8640	1.42	5732	6060	0.95
wang4.rua	1800	1348	1.34	1800	1106	1.63
wm1.rra	272	196	1.39	66	78	0.85

In Table 2, the performance of sparse matrix-vector multiplications are measured and compared before and after reordering on 2.33 GHz eight-core clovertown Xeon processor.

Table 2. Performance and speedup of sparse matrix-vector multiplication after reordering.

Matrix	Best Performance before Reordering (GFlop/s)	Best Performance after Reordering (GFlop/s)	Speedup
3dtube.psa	2.021	1.948	0.96
494_bus.rsa	0.705	0.759	1.08
af23560.rua	14.043	8.407	0.60
bayer02.rua	3.825	4.153	1.09
bayer10.rua	4.604	4.977	1.08
bcspwr01.psa	0.079	0.108	1.37
bcsstk35.rsa	10.102	7.097	0.70
coater2.rua	6.053	6.107	1.01
crystk02.psa	13.834	9.215	0.67
crystk03.rsa	5.154	4.699	0.91
ct20stif.psa	2.204	2.132	0.97
finan512.psa	5.117	5.628	1.10
gupta1.psa	2.432	2.619	1.08
lhr10.rua	7.254	8.056	1.11
lp_cre_b.rra	5.585	4.821	0.86
lp_cre_d.rra	5.329	4.839	0.91
lp_fit2p.rra	3.57	3.934	1.10
lp_nug20.rra	6.773	7.238	1.07
nasasrb.rsa	2.676	2.282	0.85
onetone2.rua	4.484	3.695	0.82
pwt.psa	5.59	6.35	1.14
rdist1.rua	7.724	7.378	0.96
rim.rua	7.931	8.219	1.04
venkat01.rua	7.902	5.043	0.64
vibrobox.rsa	6.312	4.719	0.75
wang4.rua	5.979	5.91	0.99
wm1.rra	1.373	1.682	1.23

Acknowledgments. The author thanks BeBOP group at U.C.Berkeley for insightful discussion and for providing multi-core platforms, including eight-core Xeon and Sun Niagara, for experiment.

References

1. Eun-Jin Im, Katherine A. Yelick and Richard Vuduc: SPARSITY: An Optimization Framework for Sparse Matrix Kernels**.** In: International Journal of High Performance Computing Applications, 18 (1), pp. 135--158 (2004)
2. Eun-Jin Im, Ismail Bustany, Cleve Ashcraft, James W. Demmel and Katherine A. Yelick: Performance tuning of matrix triple products based on matrix structure**.** In: Jack Dongarra, Kaj Madson, Jerzy Wasneiwski (eds.) Applied Parallel Computing. LNCS, vol. 3732, pp. 740--746. Springer, Heidelberg (2006)

3. W.-H. Liu and A.H. Sherman: Comparative Analysis of the Cuthill-McKee and the reverse Cuthill-McKee ordering algorithms for sparse matrices. In: SIAM J. Numerical Analysis pp. 198--213 (1976)
4. B. Hendrickson and R. Leland : A multilevel algorithm for partitioning graphs. Tech. Rep. SAND93-1301 Sandia National Laborotories, (1993)