A comparison of SuperLU solvers on the intel MIC architecture

Mehmet Tuncel, Ahmet Duran, M. Serdar Celebi, Bora Akaydin, and Figen O. Topkaya

Citation: AIP Conference Proceedings **1776**, 090030 (2016); doi: 10.1063/1.4965394
View online: http://dx.doi.org/10.1063/1.4965394
View Table of Contents: http://scitation.aip.org/content/aip/proceeding/aipcp/1776?ver=pdfcov
Published by the AIP Publishing

A Comparison of SuperLU Solvers on the Intel MIC Architecture

Mehmet Tuncel[1, 2], Ahmet Duran[1, a)], M. Serdar Celebi[2], Bora Akaydin[2] and Figen O. Topkaya[3]

[1]*Istanbul Technical University, Department of Mathematics, Istanbul 34469, Turkey*
[2]*Istanbul Technical University, Informatics Institute, Istanbul 34469, Turkey*
[3]*Bilgi University, Department of Industrial Engineering, Istanbul 34469, Turkey*

[a)]Corresponding author: aduran@itu.edu.tr

Abstract. In many science and engineering applications, problems may result in solving a sparse linear system $AX=B$. For example, SuperLU_MCDT, a linear solver, was used for the large penta-diagonal matrices for 2D problems and hepta-diagonal matrices for 3D problems, coming from the incompressible blood flow simulation (see [1]). It is important to test the status and potential improvements of state-of-the-art solvers on new technologies. In this work, sequential, multithreaded and distributed versions of SuperLU solvers (see [2]) are examined on the Intel Xeon Phi coprocessors using offload programming model at the EURORA cluster of CINECA in Italy. We consider a portfolio of test matrices containing patterned matrices from UFMM ([3]) and randomly located matrices. This architecture can benefit from high parallelism and large vectors. We find that the sequential SuperLU benefited up to 45 % performance improvement from the offload programming depending on the sparse matrix type and the size of transferred and processed data.

INTRODUCTION

It is valuable to evaluate the strengths and limitations of state-of-the-art solvers for a sparse linear system having matrices coming from various applications in science and engineering. SuperLU solver [2] has two different parallel versions: Message Passing Interface (MPI) based SuperLU_DIST and thread based SuperLU_MT. In this work, offload programming model is applied to sequential SuperLU 4.3, SuperLU_MT 2.1 and SuperLU_DIST 3.3. They are tested for some randomly located sparse matrices and patterned matrices.

It is challenging to decide where to place offload pragmas and there are many potential places to consider. We examine the places which are among the top time consuming code blocks by using Intel® loop and function profile viewer which is a part of Intel Composer XE Suite. Most of the trial places for offload pragmas are in the factorization routine, because the factorization part is the dominant time consuming part. According to our measurements on Hydra cluster in RZG (Rechenzentrum Garching), SuperLU_DIST performs almost 40% efficiency until 16 processors using PAPI library [11]. This indicates that there is potential for improvement.

The remainder of this work is organized as follows: First, MIC programming models are presented. Later, test results are discussed. Finally, we conclude this work.

PROGRAMMING MODELS

Intel Xeon Phi (referred as MIC in the rest of the paper) is a new kind of processor on a PCI-Express card that can operate with a CPU. It can be more effective than CPU for big vector operations. In this paper, we focus on native and offload programming models to work with Intel MIC. In native model, the code is compiled only for MIC and runs on MIC directly [12]. The offload programming model is similar to the GPGPU kernels. Most of the code is compiled for the CPU but certain parts that are more appropriate to run on MIC are compiled for the MIC

Numerical Computations: Theory and Algorithms (NUMTA–2016)
AIP Conf. Proc. 1776, 090030-1–090030-4; doi: 10.1063/1.4965394
Published by AIP Publishing. 978-0-7354-1438-9/$30.00

coprocessor. We used the MIC processors of the EURORA cluster at CINECA, Italy [7] for all tests. The EURORA cluster became number 1 in Green500 rank in June 2013. The MIC has 61 physical cores and every physical core has also four logical cores. MIC has some similarities and also differences with GPGPU's [4]. The 60 cores of MIC are accessible to the programmer.

EXPERIMENTAL RESULTS

Description of Matrices

Table 1 describes a set of patterned and randomly located matrices to be used in sequential and parallel tests.

TABLE 1. Description of patterned and randomly located matrices

Matrix Name	Order	NNZ	Nonzero pattern symmetry	Numeric value symmetry	Condition number	Origin	Kind of problem
ADD32	4960	19848	100%	31%	136.677	UFMM	Circuit simulation
CAGE8	1015	11003	100%	14%	11.4135	UFMM	DNA
CAGE10	11397	150645	100%	17%	11.0175	UFMM	DNA
ECL32	51993	380415	92%	60%	9.41×10^{15}	UFMM	Semi-conductor
MARK3JAC140SC	64089	376395	7%	1%	5.83×10^{13}	UFMM	Economic
MIXTANK_NEW	29957	1990919	100%	99%	4.40×10^{11}	UFMM	CFD
PRE2	659033	5834044	33%	7%	3.11×10^{25}	UFMM	Circuit simulation
RAND_10K_3	10000	29997	Asymmetric	Asymmetric	$\approx 7.1068 \cdot 10^{5}$	ITU	
RAND_30K_3	30000	89997	Asymmetric	Asymmetric	$\approx 1.2466 \cdot 10^{6}$	ITU	
STOMACH	213360	3021648	85%	0%	8.01×10^{1}	UFMM	Electro-physics

Sequential Test Results

There is a tradeoff between handling the memory limitation with the data storage strategies and being able to use the vector operations which is a major advantage of MIC over CPUs. SuperLU partitions matrices into chunks according to row and column ordering and distribution of matrix data on rectangular mesh. Chunk size is not proportional to the matrix size. Big matrices can be partitioned either in small or large chunks but it is not expected to obtain big chunks in small matrices. We apply the offload programming approach for the sequential SuperLU by using 120 MIC threads and MIC affinity is set to 'balanced'. In Table 2, we obtain up to 45% performance improvement for the matrix *PRE2* because this is an illustrative example for the matrices having big chunks of data. The transferred and processed data size is around 60 MB for *PRE2*. However, we couldn't see significant difference for the other matrices due to the small sizes around 10 MB.

TABLE 2. Benchmark for sequential and offload programming approach

	Sequential Time (s)		Offload Programming Time (s)	
	Factorization	Solving	Factorization	Solving
RAND_30K_3	144.85	0.13	142.71	0.13
CAGE10	21.77	0.06	21.98	0.1
ECL32	50.17	0.18	50.49	0.18
MARK3JAC140SC	43.81	0.14	44.5	0.14
MIXTANK_NEW	77.1	0.19	77.73	0.2
PRE2	723.35	1.4	497.34	0.99
STOMACH	91.68	0.49	92.14	0.49

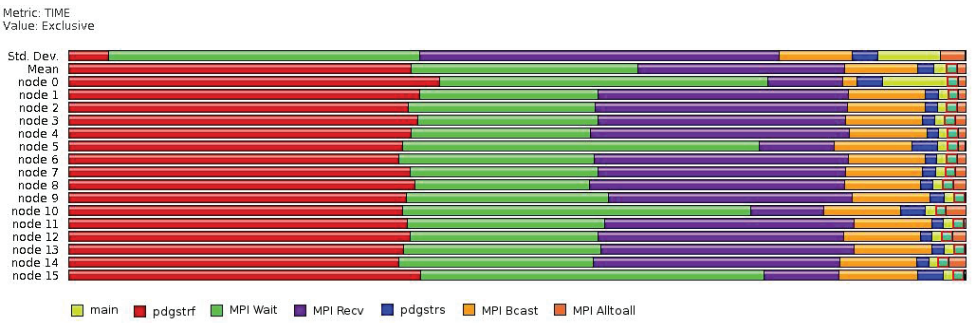

FIGURE 1. Profiling result of SuperLU_DIST in 16 processors (without MIC) for the matrix PRE2.

Parallel Test Results

This paper complements [5] and the profiling results of SuperLU_DIST are presented for a set of matrices in order to determine the most time consuming suitable parts for offloading. Figure 1 and Table 3 show the exclusive wall clock time values of SuperLU_DIST functions and MPI calls for the set of matrices described in Table 1. pdgstrf related to factorization is the most time consuming function and this is consistent with literature. MPI Wait which is a simple idle wait routine is the most time consuming routine among the communication/synchronization routines for 16 cores. MPI Wait time may decrease for a small number of cores on large matrices like PRE2 and STOMACH. Therefore, the selection of the optimal number of cores (see [1]) is important for efficient resource management depending on the application.

SuperLU_MT is tested for offload programming model. Matrix-vector operations are very important in solvers like SuperLU_MT. It is preferred to use SuperLU's own backsolve and matrix-vector multiplication (dmatvec function) code instead of BLAS library [10] because these routines are more convenient to compile and link to the SuperLU_MT code compared with the BLAS. The function 'dmatvec' receives relatively big data as an argument and it is considered that 'dmatvec' function is a good candidate for being offloaded onto the MIC. The matrix and vector data are scattered between MIC cores and the resulting vector is gathered using a wrapper function for offloading. Offloaded SuperLU_MT is tested with different matrices but very low performance is obtained (see [5]).

TABLE 3. Profiling results (in seconds) of SuperLU_DIST for the set of matrices by using TAU (Tuning and Analysis Utilities).

Matrices	# of cores	Main	pdgstrf	pdgstrs	MP_Wait	MPI_Recv	MPI_Bcast
RAND_30k_3	4	0.49	703.18	0.63	25.57	3.63	1.16
	16	0.1	98.92	1.18	16.76	1.68	1.12
PRE2	4	1.49	70.52	1.57	0.31	7.36	4.2
	16	20.24	17.28	0.78	11.46	10.44	3.76
STOMACH	4	0.74	26.81	1.69	0.34	4.06	1.97
	16	65	7.03	0.71	14.76	8.07	4.3
RAND_10K_3	4	0.03	10.3	0.08	1.98	0.44	0.03
	16	0.04	1.77	0.02	0.64	0.3	0.04
ECL32	4	0.2	8.52	0.23	1.8	1.13	0.24
	16	0.1	2.2	0.09	1.28	1.13	0.32
MIXTANK_NEW	4	0.4	6.65	0.6	1.69	1.14	1.07
	16	0.21	3.63	0.2	30.57	7.22	2.37
MARK3JAC140SC	4	0.12	3.52	0.43	0.56	0.1	0.44
CAGE10	4	0.01	3.13	0.05	0.41	0.22	0.2
ADD32	4	0.02	0.01	0.02	0.01	0.05	0.04
CAGE8	4	0.02	0.01	0.01	0.01	0.01	0.03

For SuperLU_DIST, we used automatic offloading. We employed Intel MKL functions [8] which are specifically optimized for Intel Xeon Phi to take advantage of this architecture. Intel compiler detects BLAS calls when linked to MKL [8], and decides to offload or to run in CPU according to size of data for specific BLAS functions (Xgemm, Xsymm, Xtrmm and Xtrsm [9]). The decision of offloading is made in runtime. However, the automatic offload approach seems to be not feasible for SuperLU_DIST on the set of matrices due to the offloading threshold (see [5]). The native programming approach tested for SuperLU_DIST also shows poor scale because of the non-vectorized code blocks on MIC whose clock rate is slower than CPU.

CONCLUSIONS

In this work, sequential, multithreaded and distributed versions of SuperLU solvers are compared on Intel Xeon Phi coprocessors using offload programming model for a set of patterned and randomly located matrices. This new architecture can benefit from high parallelism and large vectors. The offloading performs well when input data is at least around 60MB or more, as suggested by offload report measured on Xeon Phi. For example, when we apply the offload programming approach for the sequential SuperLU by using 120 MIC threads, we have performance improvement for the matrices having big chunks of data.

We observe that the complex algorithms like SuperLU_MT and SuperLU_DIST show low performance on MIC in the experiments. There may be several reasons for this. MIC uses PCI-E bus and this can be a bottleneck for overall performance because excessive numbers of very small floating point operations cannot be handled efficiently in MIC due to its bus access and slow clock rate compared to CPU. MICs can efficiently operate with huge vectors and/or matrices that can scale also well in CPU threads. For example, the most computationally intensive parts of the solvers are the BLAS library calls and therefore these calls are strong candidates for running on MIC or GPGPU. On the other hand, they are called in excessive number of times with different data. Consequently, these parts are not appropriate for MIC. In every call, the input data is driven to the PCI-E bus which is slower than the bus between RAM and cache memory, also cache memory and CPU. The PCI-E bus has latency and when this cost is multiplied by the number of BLAS calls, it takes very long time. Therefore, offloading these parts do not show any benefit from small matrix blocks in terms of CPU and wall clock time. We believe that this paper is important because we provide hints on minimum data dimension for effectively exploiting MIC architecture. There are several challenging experiences with MIC for different applications (see [6]) which are consistent with our findings, as well.

ACKNOWLEDGMENTS

This research was supported by the PRACE-1IP project funded in part by the EUs 7th Framework Programme (FP7/2007-2013) under grant agreement no. RI-261557 and the Project 2010PA1756 awarded under the 18th Call for PRACE Preparatory Access. The suggestions of the editors and two anonymous referees are also appreciated.

REFERENCES

[1] A. Duran, M.S. Celebi, S. Piskin, and M. Tuncel, J. of Supercomputing, **71**(3), 938-951 (2015).
[2] X.S. Li, J.W. Demmel, J.R. Gilbert, L. Grigori, M. Shao, and I. Yamazaki, "SuperLU Users' Guide", Tech. Report UCB, Computer Science Division, University of California, Berkeley, CA, (1999) update: 2011.
[3] T.A. Davis and Y. Hu, ACM Transactions on Mathematical Software, **38**(1), 1-25 (2011).
[4] F. Affinito, Introduction to GPGPU and CUDA programming, CINECA (2013).
[5] A. Duran, M.S. Celebi, B. Akaydin, M. Tuncel and F. Oztoprak, PN: RI-261557, PRACE-1IP Extension white paper, Evaluations on Intel MIC, WP 135 (2013).
[6] J. Fang, H. Sips, L.L. Zhang, C. Xu, Y. Che, and A.L. Varbanescu, "Test-driving Intel Xeon Phi," Proceedings of the 5th ACM/SPEC Int. Conf. on Performance Engineering, (ACM, New York, NY, 2014), pp. 137–148.
[7] http://www.cineca.it/en/content/eurora
[8] http://software.intel.com/en-us/intel-mkl
[9] http://software.intel.com/sites/default/files/11MIC42_How_to_Use_MKL_Automatic_Offload_0.pdf
[10] http://www.netlib.org/blas
[11] http://icl.cs.utk.edu/papi
[12] https://software.intel.com/en-us/node/528438

A Comparison of SuperLU Solvers on the Intel MIC Architecture

By: Tuncel, M (Tuncel, Mehmet)[1,2]; Duran, A (Duran, Ahmet)[1]; Celebi, MS (Celebi, M. Serdar)[2]; Akaydin, B (Akaydin, Bora)[2]; Topkaya, FO (Topkaya, Figen O.)[3]

View Web of Science ResearcherID and ORCID

NUMERICAL COMPUTATIONS: THEORY AND ALGORITHMS (NUMTA-2016)
Edited by: Sergeyev, YD; Kvasov, DE; DellAccio, F; Mukhametzhanov, MS
Book Series: AIP Conference Proceedings
Volume: 1776
Article Number: 090030
DOI: 10.1063/1.4965394
Published: 2016
Document Type: Proceedings Paper

Conference

Conference: 2nd International Conference on Numerical Computations - Theory and Algorithms (NUMTA)
Location: Pizzo Calabria, ITALY
Date: JUN 19-25, 2016
Sponsor(s): Univ Calabria, Dept Comp Engn, Modeling, Elect & Syst Sci; Natl Inst Adv Math F Severi, Italian Natl Grp Sci Computat; Natl Res Council, Inst High Performance Comp & Networking; Univ Calabria, Int Assoc Friends; Int Assoc Math & Comp Simulat; Int Soc Global Optimizat; Soc Ind Appl Math

Abstract

In many science and engineering applications, problems may result in solving a sparse linear system AX=B. For example, SuperLU_MCDT, a

linear solver, was used for the large penta-diagonal matrices for 2L) problems and hepta-diagonal matrices for 3D problems, coming from the incompressible blood flow simulation (see [1]). It is important to test the status and potential improvements of state-of-the-art solvers on new technologies. In this work, sequential, multithreaded and distributed versions of SuperLU solvers (see [2]) are examined on the Intel Xeon Phi coprocessors using offload programming model at the EURORA cluster of CINECA in Italy. We consider a portfolio of test matrices containing patterned matrices from LTEMM ([3]) and randomly located matrices. This architecture can benefit from high parallelism and large vectors. We find that the sequential Supertti benefited up to 45 % performance improvement from the offload programming depending on the sparse matrix type and the size of transferred and processed data.

Author Information

Reprint Address:
Istanbul Technical University Istanbul Tech Univ, Dept Math, TR-34469 Istanbul, Turkey.
Corresponding Address: Duran, A (corresponding author)

+ Istanbul Tech Univ, Dept Math, TR-34469 Istanbul, Turkey.

Addresses:

+ [1] Istanbul Tech Univ, Dept Math, TR-34469 Istanbul, Turkey
+ [2] Istanbul Tech Univ, Informat Inst, TR-34469 Istanbul, Turkey
+ [3] Bilgi Univ, Dept Ind Engn, TR-34469 Istanbul, Turkey

E-mail Addresses: aduran@itu.edu.tr

Funding

Funding Agency	Grant Number
PRACE-1IP project - EUs	RI-26I557 2010PA1756

View funding text

Publisher

AMER INST PHYSICS, 2 HUNTINGTON QUADRANGLE, STE 1NO1, MELVILLE, NY 11747-4501 USA

Categories / Classification

Research Areas: Mathematics; Physics

Web of Science Categories: Mathematics, Applied; Physics, Applied

Document Information

Language: English

Accession Number: WOS:000392692900083

ISBN: 978-0-7354-1438-9

ISSN: 0094-243X

Other Information