Performance Analysis of MG Preconditioning on Intel Xeon Phi: Towards Scalability for Extreme Scale Problems with Fractional Laplacians

N. Kosturski, S. Margenov, Y. Vutov

Institute of Information and Communication Technologies,
Bulgarian Academy of Sciences

Abstract. The Intel Xeon Phi architecture is currently a popular choice for supercomputers, with many entries of the Top 500 list, using it either as main processors or as accelerators/coprocessors. In this paper, we explore the performance and scalability of the Intel Xeon Phi chips in the context of large sparse linear systems, commonly arising from the discretization of PDEs. At the first step, the PCG [1] is applied as a basic iterative solution method in the case of sparse SPD problems. The parallel multigrid (MG) implementation from Trilinos ML package is utilized as a preconditioner. A matrix free algebraic multilevel solver is used to reduce the memory requirements, thus allowing the cores to be more efficiently utilized. The second part of the paper is devoted to the fractional Laplacian, that is, we consider the equation $-\Delta^{\alpha}\mathbf{u} = \mathbf{f}$, $0 < \alpha < 1$, $\Omega \subset \mathbb{R}^d$. The related elliptic boundary value problem describes anomalous diffusion phenomena also referred to as super-diffusion. The implemented method approximates the solution of the nonlocal problem by a series of local elliptic problems. The currently available numerical methods for fractional diffusion Laplacian have computational complexity, comparable e.g., to the complexity of solving local elliptic problem in $\tilde{\Omega} \subset \mathbb{R}^{d+1}$. The presented parallel results are for $\Omega = (0,1)^3$, including meshes of very large scale.The numerical experiments are run on the Avitohol computer at the Institute of Information and Communication Technologies, IICT-BAS. The presented results show very good scalability when the CPU-cores and MIC work together for a certain number of compute nodes.

1 Introduction

The paper is aimed at development of highly parallel algorithms for fractional diffusion problems with computational complexity of extreme scale. For this purpose, we investigate the parallel efficiency on the hybrid architecture of the supercomputer Avitohol (http://www.iict.bas.bg/avitohol/). The supercomputer consists of 150 compute nodes, each equipped with two 8 core (up to 16 threads) Intel Xeon E5-2650 processors and two 61 core (up to 244 threads) Intel Xeon Phi 7120P coprocessors. Each node has 32 GB of RAM and the accelerators have 16 GB. The nodes are connected via InfiniBand FDR.

Two model problems leading to large linear systems with sparse symmetric positive definite (SPD) matrices are considered. The first one is the Laplace's equation $-\Delta u = f$, in the unit cube $\Omega = [0,1]^3$ with a seven point stencil and homogeneous Dirichlet boundary conditions. The implementation of the developed parallel solution method is based on the `ml_MatrixFree` example distributed along with the Trilinos libraries. The second part of the study is devoted to the case of elliptic problems with fractional Laplacians. The numerical solution of such nonlocal problems is very expensive. A straightforward approach to the related discrete problems lead to linear systems with dense matrices. Three techniques to avoid this difficulty were recently proposed. They are based on transformation of the problem

$$\mathcal{L}^\alpha u = f$$

to a local elliptic [6] or pseudo-parabolic [7, 8] problem, or on a proper integral representation of the solution [2]. For all of them, the computational complexity is comparable to the complexity of solving local problems in $\tilde{\Omega} \subset R^{d+1}$. A comparative analysis of parallel properties of the related three algorithms for distributed memory computer architecture is presented in [3]. Some substantial advantages of the algorithm from [2] are observed there, which is the motivation to implement it in the present study.

More recently, an alternative approach aimed at reducing the computational complexity was proposed in [5], see also [4]. The linear algebraic system $\mathcal{A}^\alpha \mathbf{u} = \mathbf{f}$, $0 < \alpha < 1$ is considered, where $\mathcal{A}$ is a properly scaled sparse SPD matrix. The method is based on best uniform rational approximations (BURA) of the function $t^{\beta-\alpha}$ for $0 < t \leq 1$ and small natural β (e.g. $\beta = 1, 2$). It is important, that the algorithmic implementation of this method is practically identical to the method based on the integral representation of the solution.

The rest of the paper is organized as follows. The fractional diffusion elliptic problem and the method from [2] are presented in the next section. The developed parallel implementation approach is described in Section 3. The next Section 4 contains numerical tests of the parallel solvers for very large-scale problems. Short conclusions are given at the end.

2 Fractional Laplacian

Let us consider the elliptic boundary value problem: find $u \in V$ such that

$$a(u,v) := \int_\Omega \left(\mathbf{a}(x)\nabla u(x) \cdot \nabla v(x) + q(x)\right) dx = \int_\Omega f(x)v(x)dx, \quad \forall v \in V, \quad (1)$$

where $V := \{v \in H^1(\Omega) : \quad v(x) = 0 \quad \text{on} \quad \Gamma_D\}$, $\Gamma = \partial\Omega$, and $\Gamma = \bar{\Gamma}_D \cup \bar{\Gamma}_N$. The bilinear form $a(\cdot,\cdot)$ defines a linear operator $\mathcal{L} : V \to V^*$ with V^* being the dual of V. Namely, for all $u, v \in V$ $a(u,v) := \langle \mathcal{L}u, v\rangle$, where $\langle\cdot,\cdot\rangle$ is the pairing between V and V^*. One possible way to introduce $\mathcal{L}^\alpha$, $0 < \alpha < 1$, is through its spectral decomposition, i.e.

$$\mathcal{L}^\alpha u(x) = \sum_{i=1}^{\infty} \lambda_i^\alpha c_i \psi_i(x), \quad \text{where} \quad u(x) = \sum_{i=1}^{\infty} c_i \psi_i(x).$$

Here c_i are the Fourier coefficients of u in the L_2-orthogonal basis, $\{\psi_i(x)\}_{i=1}^{\infty}$ are the eigenfunctions of $\mathcal{L}$, orthonormal in L_2-inner product and $\{\lambda_i\}_{i=1}^{\infty}$ are the corresponding positive real eigenvalues.

As already noted, the numerical solution of the nonlocal problem $\mathcal{L}^{\alpha}u = f$ is computationally rather expensive. The following representation of the solution u is used in [2] in order to overcome the problem of non-locality:

$$\mathcal{L}^{-\alpha} = \frac{2\sin(\pi\alpha)}{\pi} \int_0^{\infty} t^{2\alpha-1} \left(\mathcal{I} + t^2\mathcal{L}\right)^{-1} dt.$$

Among others, an exponentially convergent quadrature scheme is introduced in [2]. Then, the approximation of u only involves evaluations of $(\mathcal{I}+t_i\mathcal{A})^{-1}f$, where $t_i \in (0,\infty)$ is related to the current quadrature node, and where $\mathcal{I}$ and $\mathcal{A}$ stand for the identity and the stiffness matrix corresponding to a certain approximation of the (local) diffusion equation (1). The computational complexity depends on the number of quadrature nodes. More precisely, the following quadrature formula is implemented in our parallel code:

$$L^{-\alpha} \approx \frac{2k\sin(\pi\alpha)}{\pi} \sum_{\ell=-m}^{M} e^{2\alpha y_\ell} \left(I + e^{2y_\ell}L\right)^{-1}, \tag{2}$$

where

$$k > 0, \qquad m = \left\lceil \frac{\pi^2}{4\alpha k^2} \right\rceil, \qquad M = \left\lceil \frac{\pi^2}{4(1-\alpha)k^2} \right\rceil, \qquad y_\ell = \ell k.$$

Let us assume that the utilized parallel AMG solver of the systems $(\mathcal{I}+t_i\mathcal{A})u = f$ has optimal complexity of $O(N)$, where N is the number of unknowns. Then, the computational complexity of the fractional diffusion solver is $O((m+M)N)$.

3 Parallel Implementation Approach

The developed code uses MPI for parallelization and is run on the Avitohol supercomputer at IICT-BAS introduced above. In MPI terms, the communicators encapsulate communication context and represent groups of processes that are able to communicate. All processors within a communicator have an unique number (rank). This number is used as an address for communications. All processes within a communicator participate in collective operations. There is a predefined communicator `MPI_COMM_WORLD`, which consists of all started processes. We spawn one extra process, and use it as a master which distributes the tasks.

Smaller communicators for the Laplace subproblems are created according to several parameters: *pph* – processors per host, *hpp* – hosts per (sub)problem, *ppa* – processors per accelerator and *app* – accelerators per (sub)problem. We create smaller communicators for our subtasks using `MPI_Comm_split` function. It has a color input argument and a source communicator. New communicators are created after the call, containing processors from the source one, with the same color.

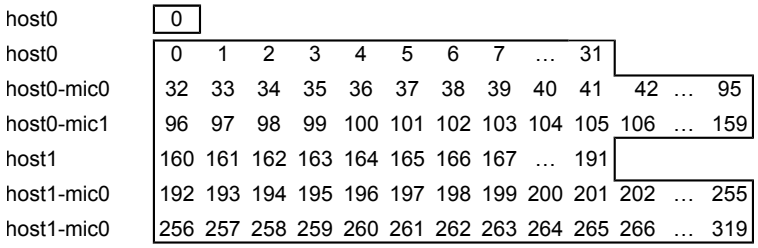

Fig. 1. Example of `ALL_COMM` communicator, where 32 processes are spawned on each host and 64 on each accelerator. An additional master process is started on `host0` to distribute the problems among groups of worker processes.

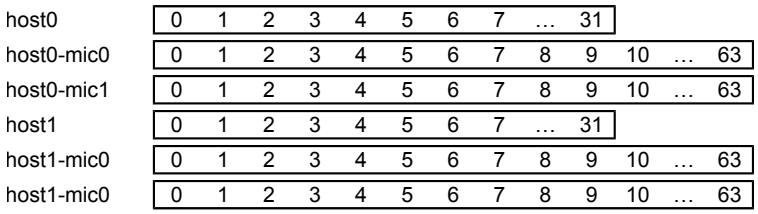

Fig. 2. Example of `HOST_COMM` communicators. These communicators represent all the processes on a particular host. It is used to construct `RANK_COMM` (see. Fig. 3) and limit the number of active processes.

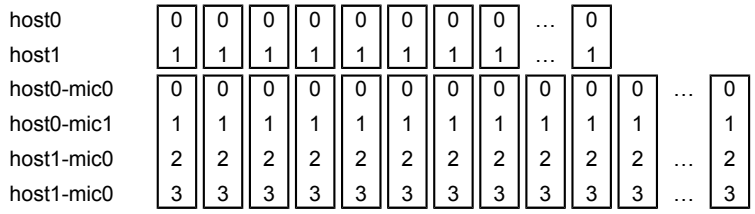

Fig. 3. Example of `RANK_COMM` communicators. They are used to group multiple hosts or accelerators into the same `COMP_COMM` (see. Fig. 4).

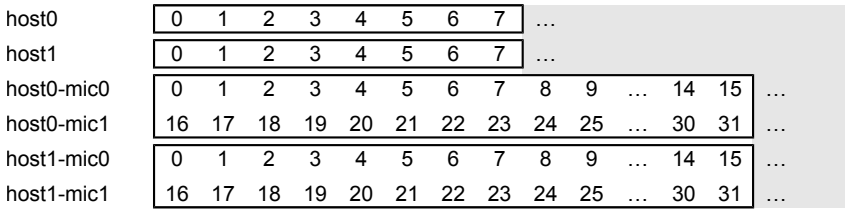

Fig. 4. Example of `COMP_COMM` communicators, where each diffusion problem is to be solved on either one node's CPUs or it's two accelerators. Gray areas correspond to idle processes, $pph = 8$, $ppa = 16$, $hpp = 1$, $app = 2$;

The first step we do is to create the communicator `ALL_COMM` which does not contain the master processor (see Fig. 1). Then using color obtained by hashing the host name, obtained from `MPI_Get_processor_name`, the communicators `HOST_COMM` are created (see Fig. 2). The ranks from `HOST_COMM` are used as colors to obtain `RANK_COMM`, (see Fig. 3). As a final step from the ranks of `RANK_COMM` and `HOST_COMM` along with the parameters *pph*, *hpp*, *ppa*, *app* communicators `COMP_COMM` are created (see Fig. 4). More precisely, the ranks of `HOST_COMM` are used to limit the number of active processes (*pph* and *ppa*) and the ones from `RANK_COMM` are used to group multiple hosts/accelerators into one communicator when required (by the *hpp* and *app* parameters). The examples in Figures 1- 4 are for distribution with following parameters: $pph = 8, hpp = 1, ppa = 16, app = 2$.

Since memory access speed is the bottleneck of sparse matrix operations for large matrices, we have applied a matrix-free multigrid algorithm. We use the multigrid preconditioner from the Trilinos ML package. The domain is partitioned in the three spacial directions across the available parallel processes. In the case of fractional diffusion problem, we solve in parallel systems with diagonally perturbed discrete Laplacians in the form $(\frac{1}{t_i}\mathcal{I} + \mathcal{A})$, $t_i > 0$.

To compute (2), the processor with rank 0 from each `COMP_COMM` requests work from the master processor. Then if there is work left, a value fore index ℓ is supplied from the master. After that processors in `COMP_COMM` proceed with the solution, accumulating partial sums. When the computations for all values of ℓ are done, a global summation is performed to obtain the final result.

4 Parallel Experiments

The first set of experiments is aimed at establishing the optimal number of processes (including a hyper-threading) per CPU and per accelerator respectively. For this purpose, we measured the solution times for the Laplace's equation, with the following checkerboard right-hand side

$$f(x,y,z) = g(x)g(y)g(z), \qquad \text{where} \qquad g(x) = \begin{cases} -1, & \text{if } x < 0.5, \\ 0, & \text{if } x = 0.5, \\ 1, & \text{if } x > 0.5. \end{cases} \tag{3}$$

The PCG is applied as a basic iterative solution method in the case of linear systems with sparse SPD matrices where a parallel multigrid (MG) implementation from the Trilinos ML package is the preconditioner. A matrix free solver is used to reduce the memory requirements thus allowing the cores to be more efficiently utilized. The PCG tolerance is set to 10^{-10} in all reported experiments. The C++ language is used in both Trilinos and our code.

Table 1 shows results on the CPUs. Even for very large problems requiring a lot of communications, the solver is able to utilize all available cores with reasonable efficiency. Moreover, using hyper-threading, albeit less efficiently, provides a faster solution in all considered cases.

Table 2 shows similar results for the case when the solver is run on the Xeon Phi accelerators. The optimal number of cores to be used for solving the problem

Table 1. Laplacian: parallel times and efficiency of the MG PCG solver on the CPUs.

DOFs	Nodes	Processes/Node	1	2	4	8	16	32
128^3	1	Time [s]	37.3	19.3	10.2	5.6	3.2	3.0
		Efficiency [%]		97	92	84	72	39
256^3	1	Time [s]	374	187	103	53	30	21
		Efficiency [%]		100	91	88	77	55
1024^3	32	Time [s]	878	512	242	126	76	58
		Efficiency [%]		86	91	87	72	48

Table 2. Laplacian: parallel times and efficiency of the MG PCG solver on the accelerators.

DOFs	Nodes	Processes/MIC	1	2	4	8	16	32	64	128
128^3	1	Time [s]	154	82	48	25	16	13	15	66
		Efficiency [%]		94	80	76	59	36	17	2
256^3	1	Time [s]	1390	720	361	191	98	70	110	
		Efficiency [%]		97	96	91	89	62	20	
512^3	8	Time [s]	1698	863	464	243	199	453	2757	
		Efficiency [%]		98	91	87	53	12	1	

Table 3. Laplacian: weak scalability of the MG PCG solver with respect to the number of nodes.

	CPUs		Accelerators	
Nodes	1	32	1	8
DOFs	256^3	1024^3	256^3	512^3
Time [s]	21.4	57.7	69.8	199.4
Efficiency [%]		74		35

in the case of Laplacian is 32 for the smaller problems and even 16 for the largest one, i.e. considerably smaller than the number of physical cores – 61. Table 3 compares the weak scalability of the solver between running on the CPUs and on the accelerators respectively. The observed scalability on the CPUs is better. Some of the differences can be attributed to the fact that the accelerators cannot use the InfiniBand FDR interconnect directly, but instead rely on the host to perform the underlying communications, potentially increasing the latency and decreasing the bandwidth.

To achieve peak performance for the Xeon-Phi processor, one should run 4 threads per physical core. This allows for utilizing all of the available AVX2 vector units. As we see from Table 5 the performance degrades with the number of threads used. There are two reasons for this. First the performance of sparse

Table 4. Numerical solution of the problem with fractional Laplacian: error estimates and corresponding number of systems to be solved for different values of α.

α	0.5	0.75	0.25
Error	1.80E-7	1.01E-7	3.06E-7
N_{SYS}	91	120	120

Table 5. Fractional Laplacian: parallel times and efficiency for $\alpha = 0.25$.

DOFs	128^3		256^3	
Nodes	Time [s]	Efficiency [%]	Time [s]	Efficiency [%]
1	147		971	
2	65	113	461	105
4	26	101	241	101

Table 6. Fractional Laplacian: parallel times and efficiency for $\alpha = 0.5$.

DOFs	128^3		256^3	
Nodes	Time [s]	Efficiency [%]	Time [s]	Efficiency [%]
1	146		989	
2	59	124	455	109
4	37	98	244	101

Table 7. Fractional Laplacian: parallel times and efficiency for $\alpha = 0.75$.

DOFs	128^3		256^3	
Nodes	Time [s]	Efficiency [%]	Time [s]	Efficiency [%]
1	235		1623	
2	103	115	729	111
4	52	113	360	113

matrix and large vector operations are memory speed bound – efficient caching is hard to achieve. The second reason is that the Pentium derived cores can issue maximum of 2 instructions per cycle. The performance of modern C++ code suffers from low instruction per clock ratio.

The next set of parallel experiments is focused on the solver for fractional Laplace's equation. Here we consider fractional powers of $\alpha = 0.25, 0.5, 0.75$. The parameter $k = \frac{1}{3}$ is used in the quadrature formula (2). The corresponding approximation error estimates and numbers of systems with diagonally perturbed discrete Laplacians to be solved are given in Table 4. The same right-hand side as in (3) is used. Tables 5–7 show the parallel times and the scalability of the solver on up to 4 nodes. The following communication parameters were used:

$pph = 32$, $hpp = 1$, $ppa = 32$, $app = 2$. The results demonstrate a rather impressive scalability of the proposed parallel implementation approach.

5 Concluding Remarks

A parallel solver for the fractional Laplace's equation is implemented and tested on a heterogeneous system, demonstrating very promising parallel efficiency. The developed pioneering parallelization approach allows to use in parallel all levels of the heterogeneous architecture of the supercomputer Avitohol, including a number of nodes, each of them integrating CPUs and MIC accelerators. Results of extreme scale numerical tests are reported.We conclude also, that some further improvements of the parallel performance of the basic PCG MG solver on Xeon Phi accelerators is desired.

The major contribution of this study is the proposed general approach leading to efficient solution of fractional diffusion problems on a heterogeneous Intel Xeon Phi architecture, utilizing a given commonly available parallel AMG solver. We would emphasize that the results presented in the last three tables demonstrate the efficiency when the CPU-cores and MIC work together, that is they have been mixed in the related runs. In the particular case, `ml_MatrixFree` is used. Here, the important message is that the developed algorithm and software tools are directly portable if another faster parallel solver is available.

References

1. O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, 1996.
2. A. Bonito, J. Pasciak, *Numerical Approximation of Fractional Powers of Elliptic Operators*, Mathematics of Computation, 84 (2015), 2083–2110
3. R. Ciegis, V. Starikovicius, S. Margenov, R. Kriauzien, *Parallel Solvers for Fractional Power Diffusion Problems*, Concurrency Computat: Pract Exper (2017), e4216, https://doi.org/10.1002/cpe.4216
4. S. Harizanov, S. Margenov, *Positive Approximations of the Inverse of Fractional Powers of SPD M-Matrices*, submitted, posted as arXiv:1706.07620 (June 2017)
5. S. Harizanov, R. Lazarov, S. Margenov, P. Marinov, Y. Vutov, *Optimal Solvers for Linear Systems with Fractional Powers of Sparse SPD Matrices*, submitted, posted as arXiv:1612.04846v1 (December 2016)
6. L. Chen, R. Nochetto, O. Enrique, A.J. Salgado, *Multilevel Methods for Nonuniformly Elliptic Operators and Fractional Diffusion*, Mathematics of Computation, 85 (2016), 2583-2607
7. P.N. Vabishchevich, *Numerically Solving an Equation for Fractional Powers of Elliptic Operators*, Journal of Computational Physics, 282 (2015), 289-302
8. R. Lazarov, P Vabishchevich, *A Numerical Study of the Homogeneous Elliptic Equation with Fractional Order Boundary Conditions*. submitted, posted as arXiv:1702.06477v1 (February 2017)