BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **20**, No 6
Special Issue on New Developments in Scalable Computing
Sofia • 2020 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2020-0066

On Dynamic Parallelization of Multilevel Monte Carlo Algorithm

Nikolay Shegunov[1,2], *Oleg Iliev*[1,3]

[1]*Fraunhofer ITWM, Fraunhoferplatz 1, 67661 Kaiserslautern, Germany*
[2]*Sofia University, FMI, 1164 Sofia, 5 James Bourchier Blvd., Sofia, Bulgaria*
[3]*Institute of Mathematics, BAS, Acad. G. Bonchev St., 8, 1113 Sofia, Bulgaria*
E-mails: shegunov@itwm.fraunhofer.de oleg.iliev@itwm.fraunhofer.de

***Abstract*:** *MultiLevel Monte Carlo* (*MLMC*) *attracts great interest for numerical simulations of Stochastic Partial Differential Equations* (*SPDEs*), *due to its superiority over the standard Monte Carlo* (*MC*) *approach. MLMC combines in a proper manner many cheap fast simulations with few slow and expensive ones, the variance is reduced, and a significant speed up is achieved. Simulations with MC/MLMC consist of three main components: generating random fields, solving deterministic problem and reduction of the variance. Each part is subject to a different degree of parallelism. Compared to the classical MC, MLMC introduces "levels" on which the sampling is done. These levels have different computational cost, thus, efficiently utilizing the parallel resources becomes a non-trivial problem. The main focus of this paper is the parallelization of the MLMC Algorithm.*

***Keywords*:** *SPDE, MLMC, UQ, Parallelization, Flow in random porous media.*

1. Introduction

Monte Carlo, MC, methods are well known class of computational methods for Uncertainty Quantification (UQ). They rely on repeated random sampling to obtain numerical results. A major drawback of this approach is its slow convergence rate, which is proportional to the variance and inverse proportional to the square root of the number of samples. MultiLevel Monte Carlo (MLMC) methods are variance reduction methods combining many cheap samples with few expensive samples. For an overview on MLMC we refer to Giles [1], Graham et al. [2] and references therein. MLMC is particularly useful for applications related to flow in random porous media, in particular, for solving elliptic SPDE. This is the problem considered here. Despite its severe superiority to MC (see, e.g., Clife et al. [3], Blaheta, Béreš and Domesová [4], Mohring et al. [5]), MLMC is still computationally very expensive, and efficient parallelization is needed. The parallelization can be done at three levels: (i) parallelizing the solution for each sample (deterministic PDE), (ii) parallelizing the solution of all samples at one MLMC level, (iii) parallelizing at all or several MLMC levels simultaneously. See

also Drzisga et al. [6] for state-of-the-art parallelization for (i) and (ii), and for general discussion. Most of the papers in the literature discuss (i) and (ii). An optimal full parallelization requires solution of a NP optimization problem, and different heuristic approaches are used in different papers to approximate its solution.

In this paper we restrict ourselves to moderate number of processors and present a dynamic approach to the load distribution in the MLMC algorithm, simultaneously over several levels. The goal is to compute the mean flux through saturated porous media with prescribed pressure drop, and known distribution of the random coefficients. Earlier, in Mohring et al. [5], Iliev, Mohring and Shegunov [7] we have discussed approaches for defining coarse levels in MLMC, and in Zakharov et al. [8], we have studied a static parallelization done at each level separately.

The simulations, that useMLMC for SPDEs can be broken into three main parts: (a) generation of correlated random field that represents the uncertainty in the SPDEs; (b) solving deterministic PDE, for each realization of the random coefficients; (c) quantifying the uncertainty, using Multilevel Monte Carlo.

The rest of the paper is organized as follows. In the next section a Laplace equation with random coefficients is considered as a model problem. Next, the generation of the stochastic coefficients is considered, together with a short discussion on the used discretization, and a text recalling the mathematical formulation of the MLMC. The proposed parallelization strategy and the results from the computational experiments are presented in the third section.

2. Mathematical model and MLMC Algorithm

Formulation of the BVP for the SPDE. Consider an elliptic SPDE in unit cube in the domain $D = (0,1)^d,\ d = 2,3$, steady state single phase flow in random porous media,

$$-\nabla \cdot [k(x,\omega)\nabla p(x,\omega)] = 0 \text{ for } x \in D = (0,1)^d,\ \omega \in \Omega. \tag{1}$$

Proper boundary conditions are prescribed, see, e.g., Iliev, Mohring and Shegunov [7]. Both the coefficient $k(x,\omega)$and the solution $p(x,\omega)$ are subject to uncertainty, characterized by the random vector ω in a properly defined random space Ω. Quantity of interest here is the mean total flux at the outflow boundary:

$$E[Q(x,\omega)], \qquad \text{where } \ Q(x,\omega) := \int_{\{x=0\}\cap\partial D} k(x,\omega)\partial_n p(x,\omega)\, dx.$$

Although relatively simple, this problem illustrates well the challenges in the MLMC simulation. Solving Equation (1) is extremely challenging due to the huge dimensionality of the stochastic space. A common way to overcome this is to assume certain distribution for $k(x,\omega)$, and to consider finite set of samples from it. One model that has been studied extensively is a log-normal distribution for $k(x,\omega)$. The used covariance function is written as follows:

$$C(x,y) = \sigma^2 \exp(-||\, x-y\, ||_2/\lambda),$$

and satisfies

$$E[K(x,.)] = 0, E[K(x,.),K(y,.)] = c(x-y) = c(y-x),$$
$$\text{for } x,y \in D \text{ and } K(x,\omega) = \log\big(k(x,\omega)\big).$$

Here σ is the variance of the distribution and λ is the correlation length and the function $c: D \to R$ is one parameter stationary covariance function. To solve Equation (1) numerically, Monte Carlo type methods are often used. Thus, first permeability field has to be sampled (generated). Then the BVP for this realization of the permeability has to be solved. Finally, the uncertainty has to be quantified by calculating the empirical mean.

Random field generation. Generating permeability fields is an essential problem in solving SPDEs. The two most common approaches of generating random fields are Karhunen-Loeve expansion and Circulant embedding. Here a Circulant embedding algorithm is employed. For detailed description of our implementation we refer to M o h r i n g et al. [5].

Deterministic BVP problem. For a fixed random filed, aka permeability field, a standard numerical scheme can be used for solving the deterministic BVP. A standard cell centered finite volume is employed as spatial discretization since it has local conservation properties. The permeability can vary orders of magnitude over one realization; thus, the governing matrix has a large condition number. We use the Conjugate Gradient method preconditioned with Algebraic Multi Grid (AMG), as a liner solver provided by **Dune** library, outlined in B a s t i a n et al. [9], since it performs well for problems with large condition number.

UQ using MLMC. We shortly rewind the idea of MLMC. For more details we refer to C l i f f e et al. [3]. Let $\omega: \Omega \mapsto R^{M \times M}$ be a random vector over some probability space (in the 2D case), where *M* denotes the number of grid cells along one direction. Consider quantity of interest $Q_M(\omega)$, defined by some functional, depending on ω.

Assume that $E[Q_M]$ can be made arbitrary close to $E[Q]$ by choosing M sufficiently large (in our case, consider fine grid). Our goal is to approximate $E[Q]$ by $E[Q_M]$. This can be achieved by computing an estimator $\hat{Q}_M$, and quantifying its accuracy using the root mean square error $e(\hat{Q}_M) = (E[(\hat{Q}_M - E[Q])^2])^{1/2}$. The standard MC estimator for $E[Q_M]$ is defined as

$$\hat{Q}_{M,N}^{MC} = \frac{1}{N} \sum_{i}^{N} Q_M^i,$$

where $Q_M^i = Q_M(\omega\_i),\ i = 1, \dots, N$, are computed with independent samples of permeability *k*. Assume the cost to compute one $C(Q_M^i) = \mathrm{O}(M^\gamma)$ sample, where γ is positive. The mean square error can be expressed as

$$e(\hat{Q}_{M,N}^{MC})^2 = V[\hat{Q}_{M,N}^{MC}] + (E[\hat{Q}_{M,N}^{MC}] - E[Q])^2.$$

Since it can be assumed that $E\left[\hat{Q}_{M,N}^{MC}\right] = E[Q_M]$, when N is large enough and $V[\hat{Q}_{M,N}^{MC}] = N^{-1}V[Q_M]$, the mean square error becomes:

(2) $$e(\hat{Q}_{M,N}^{MC})^2 = N^{-1}V[Q_M] + (E[Q_M] - E[Q])^2.$$

The second term in Equation (2) comes from the discretization of the problem. Under the assumption that M is sufficiently large, it can be considered that $(E[Q_M] - E[Q])^2 \leq \varepsilon^2/2$ holds. Then choosing $N^{-1}V[Q_M] \leq \varepsilon^2/2$, gives error estimation $e(\hat{Q}_{M,N}^{MC}) \leq \varepsilon$. Neglecting the second term in (2), it is clear that, the error of MC Algorithm is inverse proportional to the square root of the number of samples

N, and proportional to the variance. A broad class of methods aim at reducing the variance in order to obtain better accuracy with the same efforts. Multilevel Monte Carlo method, MLMC belong to this class of methods. Its main idea is to properly combine many cheap computations on a "coarse" level with few expensive corrections simulations on "finer" levels. The efficiency of the particular MLMC method depends on how well the original variance is reproduced on the "coarse" level. More specifically the method is described as follows.

Let $\{M_l: l = 0 \dots L\} \in N$ be increasing sequence of numbers called levels, with corresponding quantities $\{Q_{M_l}\}_{l=0}^{L}$, and let $s \geq 2$ be coarsening factor, such that we have $M_l = sM_{l-1}$, for $l = 0 \dots L$. Defining $Y_l = Q_{M_l} - Q_{M_{l-1}}$ and setting $Y_0 = Q_{M_0}$, we can write the following telescopic expansion for E:

$$E[Q_M] = E\left[Q_{M_0}\right] + \sum_{l=1}^{L} E\left[Q_{M_l} - Q_{M_{l-1}}\right] = \sum_{l=0}^{L} E[Y_l]. \tag{3}$$

The expectation on the finest level is equal to the expectation on the coarsest level plus sum of corrections of the expectation on consecutive levels. Each term in Equation (3) is approximated using standard MC independent estimators, with N_l samples,

$$\widehat{Y_l} = N_l^{-1} \sum_{i}^{N_l} \left(Q_{M_l}^{(i)} - Q_{M_{l-1}}^{(i)}\right).$$

To obtain a stopping criteria and express the error in terms of samples, we use Lagrangian multipliers such that:

$$e(\hat{Q}_{M,N}^{\mathrm{ML}})^2 = \sum_{l=0}^{L} N_l^{-1} V[Y_l] + (E[Q_M] - E[Q])^2 \leq 2\varepsilon^2.$$

Let $v_l = V[Y_l]$, t_l the mean time computing difference Y_l once, and $T = \sum_{l=0}^{L} n_l t_l$ be the total time for the computation. Minimizing T under the above constraint and turning it to integer value gives us

$$n_l = \left\lceil \alpha \sqrt{\left(\frac{v_l}{t_l}\right)} \right\rceil \text{ with Lagrangian multiplier } \alpha = \frac{1}{\varepsilon^2} \sum_{l=0}^{L} \sqrt{(v_l/t_l)}. \tag{4}$$

To quantify the uncertainty in Equation (3), the levels are defined as a resolution of the spatial discretization, such that the number of square cells in $D = (0,1)^2$ and cubic cells in $D = (0,1)^3$ are exact power of 2. This means, that on the finer level we have 4 times more cells than coarser level for $D = (0,1)^2$ and 8 for $D = (0,1)^3$.

Permeability approximation on coarser levels. To represent the random field (permeability) on the coarser levels (grids), we consider two recursive constructions.

Simple averaging. Consider Fig. (1). One simple idea is to set the value in a cell on the coarser level (grid) by just taking the arithmetic average of four cells in two dimensional case, and eight cells in three dimensional case (see Fig. 1).

Renormalization. Here we calculate the permeability on the coarser levels (grids) in MLMC by renormalization. This technique has been widely used in the past (and is still intensively used by many groups) for upscaling hydraulic conductivity in heterogeneous media. For details we refer to W e n and G o m e z-H e r n á n d e z [10] and the references therein. Note that the effective hydraulic conductivity obtained as a result of the renormalization can be used to calculate an effective flux. In a nutshell,

the renormalization procedure used here is based on a recursive combination of harmonic, arithmetic and geometric averaging. For each coarse cell compounded by 2×2 finer cells, the permeability coefficient on a twice-coarser grid is calculated as $K_{1234} = \sqrt{K_{1234}^{a\times h} K_{1234}^{h\times a}}$, More details can be found in Iliev, Mohring and Shegunov [7]. The procedure is repeated recursively for each coarser level.

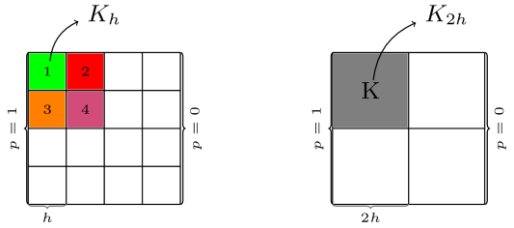


Fig. 1. Sketch on defining permeability on coarser levels

3. Computational experiments

This section is split into three parts. The first part provides new results confirming that the usage of permeability renormalization in building the coarse levels in MLMC performs better compared to the usage of just arithmetic averaging. In the second part, which is the main part of this paper, strategies for dynamic parallelization of the MLMC Algorithms are discussed. Finally, in the third part first results illustrating the performance of the parallelization algorithm are presented.

Permeability renormalization vs arithmetic averaging. Recall that ideally the coarsest level in MLMC should have a variance which is almost equal to the variance at the finest level, and the variance of the correction between finest and next coarse level should be as small as possible. In this case many (cheap) simulations will be performed on the coarsest level, and few (expensive) simulations will be performed at the finest level.

Table 1 summarizes simulation results for four-level MLMC for the case when $\sigma = 2.75, \lambda = 0.25$, and the prescribed tolerance is $\varepsilon = 1e^{-3}$. The mean flux computed with the plain MC is used as a reference. The numbers in the last four columns in the Table 1 are the numbers of samples required per level. It is directly observable that permeability renormalization transfers better the variance to the coarsest level and leads to a small variance at the finest level. Indeed, it requires only 144 samples at the finest level, contrary to 2211 samples required at the finest level in the case of arithmetic averaging for the permeability.

Table 1. Four-level MLMC, 960 processor cores, $\sigma = 2.75, \lambda = 0.25$, $\varepsilon = 1\times10^{-3}$

Method	E[Q]	$\frac{\lvert \boldsymbol{E}[\boldsymbol{Q}] - \boldsymbol{E}_{\mathbf{MC}}[\boldsymbol{Q}] \rvert}{\boldsymbol{E}_{\mathbf{MC}}[\boldsymbol{Q}]}$	Time, s	$\boldsymbol{Y_0}$	$\boldsymbol{Y_1}$	$\boldsymbol{Y_2}$	$\boldsymbol{Y_3}$
AVG	1.3411	5.9×10^{-4}	3696	1.5×10^{6}	9035	3760	2211
RENORM	1.3412	5.9×10^{-4}	3262	1.4×10^{6}	2674	1025	144

Results from simulations for a slightly harder problem, $\sigma = 2.75, \lambda = 0.3$, can be found in Table 2. Five-level Multilevel Monte Carlo algorithm and 3480 processors are used here. Similar results can be observed – permeability renormalization outperforms the arithmetic averaging.

Table 2. Five-level MLMC, 3840 processors cores, $\sigma = 2.75, \lambda = 0.3$, and $\varepsilon = 1\times10^{-3}$

Method	E[Q]	$\frac{\lvert \boldsymbol{E}[\boldsymbol{Q}] - \boldsymbol{E}_{\mathbf{MC}}[\boldsymbol{Q}] \rvert}{\boldsymbol{E}_{\mathbf{MC}}[\boldsymbol{Q}]}$	Time, s	$\boldsymbol{Y_0}$	$\boldsymbol{Y_1}$	$\boldsymbol{Y_2}$	$\boldsymbol{Y_3}$	Y_4
AVG	1.4317	5.5×10^{-4}	2273	2.4×10^{6}	11043	10646	4700	2273
RENORM	1.4316	5.5×10^{-4}	1156	2.3×10^{6}	9473	3608	1420	252

Parallel MLMC. Solving SPDE for Laplace equation, Equation (1), for specified stochastic parameters λ, σ, fixed fine grid resolution, and prescribed number of levels and MLMC tolerance, is based on the following algorithmic procedure: (i) based on former experience or on intuition, prescribe the initial number of samples for each of the levels; (ii) perform simulations with these predefined numbers of samples and compute an approximation to the variance at each level; (iii) use these approximate variances and Equation (4) to estimate the total number of samples needed per level; **(**iv) for each $\widehat{Y}_l$, generate the required additional number of random vectors, and for each of them solve the respective deterministic problem; (v) check for convergence with Equation (4), and compute final statistical moments if converged, or go to (iii) in the opposite case. Note that due to inaccuracy in evaluation of the variances at different levels after the initial stage, the total number of the needed samples per level, may be underestimated or overestimated. To minimize this risk, it is recommended to perform several estimation steps. Equation (4) implies that at each estimation step information from all levels has to be collected, that is, each estimate is a synchronization point of the parallel algorithm. After each "solve stage", a new estimated numbers of samples per level may be computed, and rebalancing of the available processors may be required. In such a dynamic rebalancing the number of processors assigned to the different levels may change, aiming to obtain better balancing for the newly estimated number of samples per level. The problem becomes even more involving, when the deterministic problems are large, and several processors has to be assigned for solving one deterministic problem.

Additionally, a problem can arise when the algorithm is close to converging. In this case, Equation (4) can give an estimate that just few samples need to be solved per a certain level, and this imposes challenges to balancing the work of the processors. To improve upon that problem, we introduce a measure for convergence "rate" between two "estimate" cycles, namely, $\delta = c - p$. Here c denotes the RMSE for the current cycle, and p denotes the RMSE for the previous one. It is clear that when we are very close to the desired tolerance, the difference δ will be very small. If a prescribed threshold for δ is reached, one can either terminate the algorithm, or can artificiality increase the estimated number of samples for certain levels.

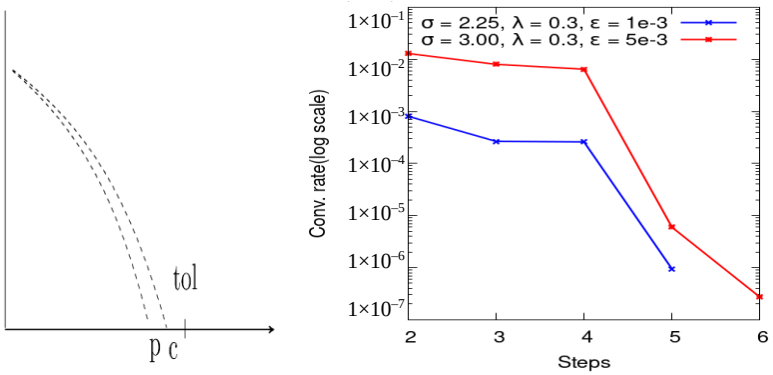


Fig. 2. Typical convergence of MLMC

Let us focus on one Estimate-Solve cycle of the algorithm, and without loss of generality assume that we have three-level MLMC Algorithm. Furthermore, suppose that we have already obtained statistical information from a previous cycle and have computed an estimate for the required number of samples per level using Equation (4).

Let us denote by $N_i,\ i = 0, 1, 2$, the number of required realizations (samples) per Monte Carlo estimator $\widehat{Y}_i$, where N_0 is the estimated number of samples for the estimator $\widehat{Y}_0$, corresponding to the coarsest level. Denote by p_i the number of processors allocated to $\widehat{Y}_i$, and by $p_{l_i}^g$ the respective group size of processors working on a single realization at this level. Finally, denote by t_i the respective time constants for solving a single problem once on a single processor on the respective level, and by p^{total} the total number of available processors. Then we can compute the total CPU time for the current "estimate-solve" cycle as

$$T_{\text{CPU}}^{\text{total}} = N_0 t_0 + N_1 t_1 + N_2 t_2.$$

Ideally the time which each processor should work is calculated as $T_{\text{CPU}}^{p} = \frac{T_{\text{CPU}}^{\text{total}}}{p^{\text{total}}}$.

Then dividing the CPU time needed for a $\widehat{Y}_i$ by T_{CPU}^{p}, we get a continuous value for the number of processors for each of the level estimators,

$$p_i^{\text{ideal}} = \frac{N_i t_i}{T_{\text{CPU}}^{p}} \quad \text{for } i = 0, 1, 2.$$

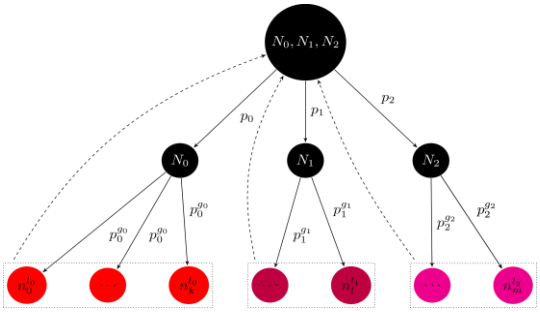


Fig. 3. Scheduling procedure for the parallel MLMC Algorithm

Let's further assume that we want to distribute all of the available processors to work simultaneously on all of the estimators Y_i. Unlike most of the papers, we do not parallelize level by level, we use the fact that MLMC provides top level coarse graining, and parallelization is done on all MLMC levels simultaneously.

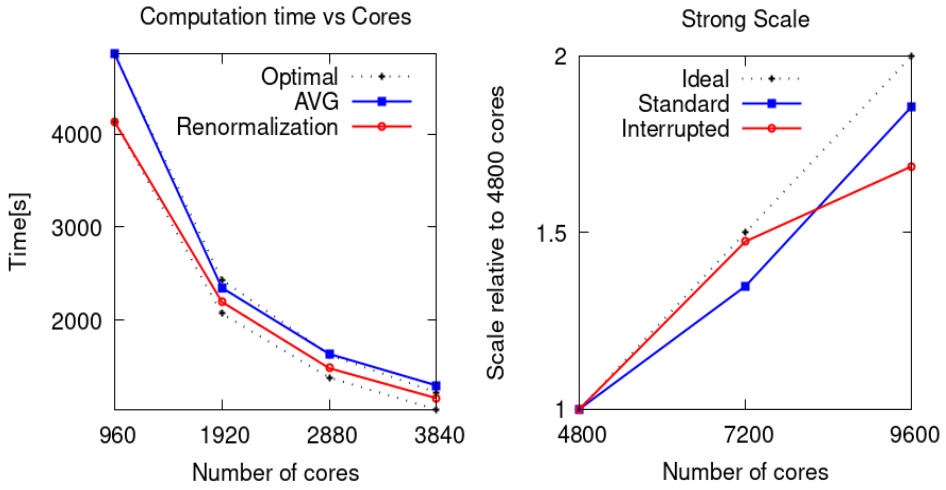
Then we can take

$$p_i = \lfloor p_i^{\text{ideal}} \rfloor, \text{ for } i = 0, 1, 2,$$

such that each $p_i \equiv 0 (\text{mod}\, p_{l_i}^g)$. This gives an integer number for the processors allocation per level. The unallocated processors can be left unused for this cycle, especially if this is a small number. Alternatively, one can try to further search for an optimal scheduling by constructing possible set of all upper and lower bounds by allocating the leftover processors to different estimators. In other words, we try to find an optimal integer processor distribution for the estimators between $\sum_{i=0}^{2} p_i$ and p^{total}. Till now we have only looked at the case where we distributed all of the available processors to work simultaneously on all of the level estimators. It might be difficult to find optimal strategy in this case because of strong imbalance of work between the estimators for the different levels. To find a reasonable strategy, we can consider all possible combinations of estimator groups. For example, it is possible to first schedule parallel computations for the estimator $\{Y_0\}$, and after that to schedule parallel computations simultaneously for $\{Y_1, Y_2\}$, and for the leftover from the coarsest level, if any. Note that this general approach includes the case of computing MLMC level by level. There are a few options to tackle the parallelism for each single estimator. A trivial way to distribute the samples to the available processors is to divide the work equally among them. This approach works well for many small samples with small computational time per sample. Another approach to distribute the work is by job dispatching using master-slave paradigm, similar to the job queue in Multi-Threading approach. In this way the problem with varying computational costs per sample is solved, however this will introduce huge number of small messages during computation. A third possibility of job distribution is a hybrid between the two. To improve the overhead due to integer rounding and varying computational time per sample in the following approach. Instead of waiting all the groups to finish the computations, the group that has finished its task first, can inform the other groups to stop computations, so that a rescheduling of the remaining work can be done. Thus, we have message exchange only if one group has finished its jobs. This is somewhat analogous to job-stealing in Multi-Threading. Further on, there are two possibilities for rescheduling: locally on a given estimator (level), or globally, across estimators for all levels, using Equation (4). Which one is more efficient, depends, on number of groups we have, how many processors etc. The results presented below are obtained using the global redistribution technique. The other approaches, as well as comparison with the scheduling strategy from D r z i s g a et al. [6], where level by level parallelization is done, is ongoing work and will be reported elsewhere.

Parallel MLMC Simulations. The computational times for $\sigma = 2.75, \lambda = 0.3$ with the two considered coarsening for the permeability are shown on Fig. 4a (left). In the simulations we use different size of the groups (number of processors allocated

per single deterministic problem) at different levels, namely $P_{l_0}^g = 1$, $P_{l_0}^g = 2$, $P_{l_0}^g = 3$, $P_{l_0}^g = 4$, $P_{l_0}^g = 5$. The different times and processor distribution per sample do not affect significantly the efficiency of the algorithm for the two considered coarsening.



(a) (b)

Fig. 4. Computational performance of permeability renormalization vs averaging (a); and scaling for large number of processor cores in the case of one scheduling per "solve-estimate" cycle and in the case of one or more enforced re-scheduling (b).

The performance of the parallel algorithm for large number of processor cores is shown on Fig. 4b (right). Renormalization is used in the coarsening. The computational times in the case when we may have several enforced estimates and rescheduling cycles (denoted as Yes, interrupted), and in the case when we wait for all the work scheduled for the current cycle to be completed before having a new estimate and scheduling (denoted as No, cycles not interrupted) are summarized in Table 3. In the case of interruption of a cycle (due to appearance of idle processors), updated estimates for the numbers of the needed samples per level are calculated using Equation (4) and the discussed optimization approach, and rescheduling of the available resources over all levels is carried out. This is a more efficient parallelization approach, as it is confirmed also by the numbers in Table 3.

Table 3. Five-Level MLMC Simulation on 7200 processor cores

Interrupted	$E[Q]$	Time, s	Median	Q_4
NO	1.7656	1251	1220	1416
YES	1.7655	955	947	1031

4. Summary

An approach for dynamic parallelization simultaneously for all or several levels of Multilevel Monte Carlo algorithm is presented aiming at obtaining better load distribution among the processor cores. In this way MLMC is used not only as a variance reduction method, but also is used to provide coarse grain parallelization. The presented simulation results demonstrate the efficiency of the parallelization approach.

***Acknowledgments*: The** authors gratefully acknowledge the provided computing time on SuperMUC at Leibniz Supercomputing Centre (**www.lrz.de**).

References

1. Giles, M. B. Multilevel Monte Carlo Methods. – Acta Numerica, Vol. **24**, 2015, pp. 259-328. DOI: 10.1017/S096249291500001X.
2. Graham, G., F. Y. Kuo, D. Nuyens, R. Scheichl, I. H. Sloan. Quasi-Monte Carlo Methods for Elliptic PDEs with Random Coefficients and Applications. – Journal of Computational Physics, Vol. **230**, 2011, No 10, pp. 3668-3694.
3. Cliffe, K. A., M. B. Giles, R. Scheichl, A. L. Teckentrup. Multilevel Monte Carlo Methods and Applications to Elliptic PDEs with Random Coefficients. – Computing and Visualization in Science, Vol. **14**, 2011, No 1. DOI: 10.1007/S00791-011-0160-X.
4. Blaheta, R., M. Béreš, S. Domesová. A Study of Stochastic FEM Method for Porous Media Flow Problem. – In: R. Bris, P. Dao, Eds. Proc. of International Conference Applied Mathematics in Engineering and Reliability. CRC Press, 2016, pp. 281-289.
5. Mohring, J., R. Milk, A. Ngo, O. Klein, O. Iliev, M. Ohlberger, P. Bastian. Uncertainty Quantification for Porous Media Flow Using Multilevel Monte Carlo. – In: Proc. of Int. Conf. Large-Scale Scientific Computing, Springer International Publishing, 2015, pp. 145-152.
6. Drzisga, D., B. Gmeiner, U. Rude, R. Scheichl. Scheduling Massively Parallel Multigrid for Multilevel Monte Carlo Methods. – SISC, Vol. **39**, 2017, No 5, S873-97.
7. Iliev, O., J. Mohring, N. Shegunov. Renormalization Based MLMC Method for Scalar Elliptic SPDE. – In: Proc. of International Conference on Large-Scale Scientific Computing, Springer, 2017, pp. 145-152.
8. Zakharov, P., O. Iliev, J. Mohring, N. Shegunov. Parallel Multilevel Monte Carlo Algorithms for Elliptic PDEs with Random Coefficients. – In: Lecture Notes in Computer Science. Vol. **11958**. 2020, p. 463.
9. Bastian, P., M. Blatt, A. Dedner, C. Engwer, R. Klofkorn, M. Ohlberger, O. Sander. A Generic Grid Interface for Parallel and Adaptive Scientific Computing. Part I: Abstract Framework. – Computing, Vol. **82**, 2008, No 2-3, pp. 103-119.
10. Wen, X. H., J. J. Gomez-Hernández. Upscaling Hydraulic Conductivities in Heterogeneous Media: An Overview. – J. Hydrol., Vol. **183**, 1996, No 1-2, pp. ix-xxxii.

Received: 25.09.2020; Second Version: 30.10.2020; Accepted: 4.11.2020