J. Parallel Distrib. Comput. 84 (2015) 24–36

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Intrinsic fault tolerance of multilevel Monte Carlo methods

Stefan Pauli [a,b,*], Peter Arbenz [a], Christoph Schwab [b]

[a] ETH Zürich, Computer Science Department, Universitätsstrasse 6, 8092 Zürich, Switzerland
[b] ETH Zürich, Seminar for Applied Mathematics, Rämistrasse 101, 8092 Zürich, Switzerland

HIGHLIGHTS

- Fault tolerant Monte Carlo/fault tolerant multilevel Monte Carlo.
- Intrinisic recilience to hardware failure at runtime.
- Exascale parallel computing.

ARTICLE INFO

Article history:
Received 18 August 2012
Received in revised form
9 April 2015
Accepted 10 July 2015
Available online 17 July 2015

Keywords:
Multilevel Monte Carlo
Fault tolerance
Failure resilience
Exascale parallel computing

ABSTRACT

Monte Carlo (MC) and multilevel Monte Carlo (MLMC) methods applied to solvers for Partial Differential Equations with random input data are proved to exhibit intrinsic failure resilience. Sufficient conditions are provided for non-recoverable loss of a random fraction of MC samples not to fatally damage the asymptotic accuracy versus work of a MC simulation. Specifically, the convergence behavior of MLMC methods on massively parallel hardware with runtime faults is analyzed mathematically and investigated computationally. Our mathematical model assumes node failures which occur uncorrelated of MC sampling and with general sample failure statistics on the different levels and which also assume absence of checkpointing, i.e., we assume irrecoverable sample failures with complete loss of data. Modifications of the MLMC with enhanced resilience are proposed. The theoretical results are obtained under general statistical models of CPU failure at runtime. Particular attention is paid to node failures with so-called Weibull failure models on massively parallel stochastic finite volume computational fluid dynamics simulations are discussed. We discuss the resilience of massively parallel stochastic Finite Volume computational fluid dynamics simulations.

1. Introduction

Monte Carlo (MC) methods estimate statistical moments of random variables (such as means or so-called "ensemble averages") by sample averages [6]. The goal can, for instance, be to determine the expected solution of a partial differential equation (PDE) with random initial or boundary conditions that follow some statistical law [12–14]. Then, each sample is the solution of the PDE for a random input (such as, in the context of hyperbolic systems of conservation laws, a particular initial/boundary condition). The statistical independence of the input data makes it possible to execute the simulations corresponding to each sample in parallel. The slow convergence of Monte Carlo methods ($M^{-1/2}$ for M draws of input data) entails large numbers of samples. This, in turn, implies good parallel scalability of MC methods to large numbers of processors. Mostly, the simulations take a similar amount of time such that a distribution among large numbers of processors with a balanced load is achieved quite easily. In a parallel setting a serious problem is to guarantee the statistical independence of the random input draws (see, e.g., [21]).

Multilevel Monte Carlo (MLMC) methods were recently proposed in [7,12] in order to improve the accuracy versus work. They can be used for efficient numerical simulations of stochastic ordinary or partial differential equations. Unlike MC methods where samples are only computed on one discretization, MLMC methods use a hierarchy of discretization levels hence computations are done on many different discretizations. Based on the expected solution computed on the coarsest discretization level, the expected difference from this level to the next finer one is added, until the finest discretization level is reached. In MLMC methods the expected difference from two consecutive discretization levels is estimated using the MC method. Hence, in this paper the difference of

* Corresponding author at: ETH Zürich, Computer Science Department, Universitätsstrasse 6, 8092 Zürich, Switzerland.
E-mail addresses: stefan.pauli@inf.ethz.ch (S. Pauli), arbenz@inf.ethz.ch (P. Arbenz), schwab@sam.math.ethz.ch (C. Schwab).

http://dx.doi.org/10.1016/j.jpdc.2015.07.005
0743-7315/

a realization computed on two consecutive discretization levels is referred to as a MLMC sample of a certain "level". Throughout this paper, a "level" does, therefore, not denote a discretization level but a level related to a MLMC sample. Note that the discretization levels need not be related to a grid hierarchy. MLMC can also be applied, e.g., to mesh-free Feynman–Kac problems [19,30]. It is by now known (see, e.g., [2,7,12–14,19]) that under rather general assumptions, MLMC methods converge faster than MC, in terms of overall computational work, i.e., cumulative execution time. The cumulative execution time and memory consumption of the computation of samples depend on the levels and differ considerably: the computation of a MLMC sample of a 'finer' level may require much more compute resources (execution time, memory space, number of cores) than of a sample of a 'coarser' level. The load of a MLMC simulation is therefore not as easy to balance [27] as in MC, as there are only few samples on the fine levels. Nevertheless, in the context of partial differential equations with random inputs, the MLMC method allows the approximation of ensemble averages of the solution with a complexity analogous to that necessary for one numerical solution of the deterministic problem on the finest mesh [2,12].

The present study is based on the following assumptions: (a) in large scale simulations on emerging, massively parallel computing platforms processor failures at runtime are inevitable [4,5], and occur, in fact, with increasing frequency as the number of processors increases, respectively the quality of processors decreases; (b) processor failures at runtime can, in general, not be predicted, but occur randomly and should therefore be modeled as stochastic processes; (c) processor failures at runtime are *not checkpointed* and *not recoverable*; (d) the algorithm of interest has *redundancy by design* in order to "survive" a certain number of (non-checkpointed) failure events with random arrivals.

Assumptions (c) and (d) exclude a large number of currently used standard algorithms, for which any loss of data entails abortion of execution. We mention only standard Gaussian Elimination with loss of one pivot element. Other algorithms may, however, tolerate partial loss of data at runtime. We think of iterative solvers of large, linear systems which may converge even if one or several iterates are "lost" due to hardware failures and so satisfy assumption (d); interestingly, assumption (b) implies that the *convergence results of deterministic algorithms for deterministic problems on random hardware will necessarily be probabilistic in nature*. Here, we consider the case when the algorithm under consideration is stochastic by design, such as Monte Carlo (MC) methods. As we argue in the present paper, MC methods, being *probabilistic in nature*, are intrinsically fault tolerant: as we prove, the loss of (a "subcritical" fraction of) information by failed samples does not render the whole simulation useless as is the case, e.g., in many matrix computations, such as Gaussian Elimination. MC samples which were lost due to node failures at runtime can be repeated since new, independent samples can be generated to replace the failed ones.

We provide a mathematical argument predicting that the convergence behavior of MC is not affected substantially if the failed samples are simply disregarded, provided there are "not too many" (made precise in the mathematical analysis, and corroborated in the numerical experiments) of these failures.

Specifically, in the present paper we analyze the performance of both MC and MLMC PDE solvers in the presence of hardware failures at runtime. In particular, we investigate the convergence behavior of these methods if processors fail according to a stochastic failure model; the presently developed mathematical analysis accommodates rather general failure models. Naturally, to arrive at a theory which is amenable to rigorous mathematical treatment, a number of simplifying assumptions had to be made. In particular, in our analysis we do not distinguish among different reasons of processor failure. So, we do not distinguish between node, program, network, or any other type of failure. We assume that the complete MC sample is lost if one of the (maybe multiple) processors fails that are used for its simulation. We disregard all samples affected by a failure and compute the results with the 'surviving' ones.

While in MC all samples are from the (single) finest level (or grid), MLMC gets its statistics also from samples corresponding to coarser grids. (The resolution of the finest level is determined by the required discretization error.) By using information on multiple levels MLMC needs much fewer samples on the finest level than ordinary MC to attain the same quality of answer. MLMC turns out to be much more efficient than MC. To get the optimal MLMC convergence rate (with respect to work), it is crucial to choose properly the numbers M_ℓ of samples on level ℓ.

In the presence of failures without checkpointing MC samples on all levels might be irrevocably lost. The larger (in the sense that they are defined on finer meshes) samples of the finer levels are more vulnerable than the (larger number of) smaller samples on the coarser levels. With very high failure rates it might not be feasible that sufficiently many samples survive on the finer levels. In general, the error components of faulty levels increase and the overall convergence rate is reduced. With a sufficiently high failure rate all samples on a particular level may get lost. In this case, the attainable error is bounded from below by the discretization error of that level.

Our main mathematical results are as follows: we prove convergence of MC and MLMC for the first moment (sample average) provided that sufficiently many samples survive on average. We compute the effect of failures according to existing failure models. Numerical experiments of MLMC for hyperbolic PDE's coupled with the Weibull failure model validate our theory. We further investigate the failure resilience of two and three dimensional time-dependent grid applications, like finite elements, finite differences, or finite volumes. These results are obtained by MLMC simulations treating the sample sizes M_ℓ as random variables.

We also discuss FT issues regarding MPI. In the present standard MPI-3.0 [15], failure of a single MPI process is fatal for the entire MLMC simulation. For our approach to work, MPI would have to be extended by a mechanism to survive losses of MPI processes at runtime, and continue with the remaining ones. Based on this paper the proposed fault tolerant MLMC was implemented [20,30] using the User Level Failure Mitigation (ULFM) [3], a fault tolerant version of MPI. We compare the theoretical error bound with the measured results from this implementation.

The paper is organized as follows: In Sections 2 and 3 we give error bounds for MC and MLMC, respectively, in the presence of a statistical loss of samples. In Section 4 we discuss the Weibull failure model. In Section 5 we conduct a number of numerical experiments to investigate how convergence is affected by failure. We consider two and three dimensional problems with different convergence rates of the PDE solver. We further discuss the practical applicability of our approach, and outline a possible procedure if the stochastic failure model is unknown.

2. MC with a random number of samples

We first introduce a fault tolerant MC (FT-MC) method. Starting from there the fault tolerant technique used in MLMC is derived.

We are interested in the expected value $\mathbb{E}[X]$ of a random variable (RV) X taking values in a Banach space B, on the probability space $(\Omega, \mathscr{A}, \mathbb{P})$, with sample space Ω, σ-algebra $\mathscr{A}$ and probability measure $\mathbb{P}$ [17]. If the 2nd moments of X exist the Monte Carlo method can be used to estimate $\mathbb{E}[X]$. Given a