Eidgenössische Technische Hochschule Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Intrinsic fault tolerance of multi level Monte Carlo methods

St. Pauli, P. Arbenz and Ch. Schwab

Research Report No. 2012-24
August 2012

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

Intrinsic Fault Tolerance of Multi Level Monte Carlo Methods

Stefan Pauli[a,b,1], Peter Arbenz[a], Christoph Schwab[b,2]

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

Abstract

Monte Carlo (MC) and Multilevel Monte Carlo (MLMC) methods applied to solvers for Partial Differential Equations with random input data are shown to exhibit intrinsic failure resilience. Sufficient conditions are provided for non-recoverable loss of a random fraction of samples not to fatally damage the asymptotic accuracy vs. work of an MC simulation. Specifically, the convergence behavior of MLMC methods on massively parallel hardware is analyzed mathematically and computationally, under general assumptions on the node failures and on the sample failure statistics on the different MC levels, in the 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. Specifically, node failures with the so-called Weibull failure models on massively parallel stochastic Finite Volume computational fluid dynamics simulations are discussed.

Keywords: Multilevel Monte Carlo, fault tolerance, failure resilience, exascale parallel computing

1. Introduction

Monte Carlo (MC) methods estimate statistical moments of random variables (such as means or so-called "ensemble averages") by sample averages, i.e. by repeated simulations of the quantities of interest that depend on random inputs [6]. The goal of the simulation 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]. In computing the sample averages, each random input (such as, in the context of hyperbolic systems of conservation laws, a particular initial/boundary condition) gives rise to a simulation that is independent of the other simulations. The statistical independence of input draws makes it possible to execute the simulations corresponding to each draw 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 the only serious problem is to guarantee the statistical independence of the random input draws (e.g. [17]).

Email addresses: stefan.pauli@inf.ethz.ch (Stefan Pauli), arbenz@inf.ethz.ch (Peter Arbenz), schwab@sam.math.ethz.ch (Christoph Schwab)

[1] The work of this author has been funded under ETH interdisciplinary research grant CH1-03 10-1.

[2] CS acknowledges partial support by the European Research Council under grant ERC AdG 247277-STAHDPDE.

Multilevel Monte Carlo (MLMC) methods were recently proposed in [7, 12] in order to improve the convergence verus work. These methods are more difficult to parallelize. MLMC methods have multiple discretization levels on each of which a Monte Carlo simulation is performed. In the end the results of all levels are summed to yield the desired quantity. MLMC methods converge faster than MC, in terms of work, i.e., execution time. The execution time and memory consumption of the computation of samples depend on the mesh levels of the discretization and differ considerably: the computation of an MC sample of a 'finer' level may require much more compute resources (execution time, memory space, number of cores) compared to a sample of a 'coarser' level. The load of an MLMC simulation is therefore not as easy to balance as in ordinary MC, as there are only few samples on the fine levels. Nevertheless, in the context of partial differential equations with random inputs, the MLMC has been proved in [2, 12] to allow the approximation of ensemble averages of the solution with accuracy versus complexity analogous to that necessary for one numerical solution of the deterministic problem (resp. of a "single path") on the finest mesh. The simultaneous use of different sample sizes on different mesh levels poses, however, challenges to parallelization with good load balancing (see, e.g. [23]).

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 [3, 4], 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 modelled 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 loss of data. We think of, for example, 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 on random hardware will be probabilistic in nature*. Here, we consider the case when the algorithm under consideration is stochastic by design, such as Monte Carlo (MC) methods. Being *probabilistic in nature*, MC methods are intrinsically fault tolerant: the loss of (a limited amount of) information by failed samples does not render the whole computation useless as is the case, e.g., in many matrix computations. Failed samples can be repeated or new independent samples can be generated to replace the failed ones. We show that here, the convergence behavior of MC is not affected substantially if the failed samples are simply disregarded, provided there are not too many of them.

In this 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 with a certain probability. In our analysis, we do not distinguish the reasons of 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 (wrt. work), it is crucial to choose properly the numbers M_ℓ of samples on level ℓ.

In the presence of failures samples on all levels are lost. The larger samples of the finer levels are more vulnerable than the smaller samples on the coarser levels. On finer levels large enough M_ℓ may not be sustainable. The error components of faulty levels increase and the overall convergence rate is reduced. In particular, with a sufficiently high failure rate a whole level may get lost such that the attainable error is bounded by the discretization error of that level.

We prove convergence of MC and MLMC for the first moment (mean) 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 one and three dimensional time-dependent grid applications, like finite elements, finite differences, or finite volumes. These results are obtained by MC simulations treating the sample sizes M_ℓ as random variables.

The paper is organized as follows: In Sections 2 and 3 we give error bound 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 one and three dimensional problems with different convergence rates of the PDE solver. We also discuss issues regarding MPI. In the present implementation, failure of one single MPI process is fatal for the entire MLMC simulation. For our method 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.

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 in the probability space $(\Omega, \mathscr{A}, \mathbb{P})$, with sample space Ω, σ-algebra $\mathscr{A}$ and probability measure $\mathbb{P}$ [16]. If the 2nd moments of X exist the Monte Carlo method can be used to estimate $\mathbb{E}[X]$. Given a fixed number M of independent, identically distributed samples X^i, $i = 1, 2, \ldots, M$, the MC approximation of $\mathbb{E}[X]$ is defined by

$$E_M[X] := \frac{1}{M} \sum_{i=1}^{M} X^i. \tag{1}$$

The mean square error in the estimator (1) is known to be (see, e.g., [12, 22])

$$\|\mathbb{E}[X] - E_M[X]\|^2_{L^2(\Omega;L^1(\mathbb{R}^d))} = \frac{1}{M} \|X\|^2_{L^2(\Omega;L^1(\mathbb{R}^d))}. \tag{2}$$

The "embarassingly parallel" evaluation of (1) across a possibly large number of nodes is a common approach to reduce wall clock time in MC simulations. In the present work, we highlight and capitalize on a second feature of the MC estimator (1): software or hardware failures at runtime may cause nodes to fail or even crash, such that some samples get lost. We will give sufficient conditions on the node failure statistics to prove that

1. *it is reasonable to continue the MC simulation without checkpointing,*
2. *that no recovery of samples from failed nodes is required,*
3. *that the statistical quality of the MC simulation is unaffected provided "node failures at runtime do not occur too frequently", and*
4. *that there is a certain critical node failure intensity above which the MC simulation does deterioriate, almost surely.*

Let us now be more specific about the mathematical model from which these assertions are deduced.

In the presence of system failures at runtime, the sample size M in (1) is not a fixed number anymore, but becomes itself a random variable $\hat{M}$. We denote the probability space for the failures by $(\Omega', \mathscr{A}', \mathbb{P}')$ and the respective expectation by $\mathbb{E}'[\cdot]$. We assume throughout the paper that the node failures at runtime occur with statistics that are independent of the realizations of X. This implies that the runtime to compute a solution of a realization of X is independent of e.g., the realization. In particular, we furthermore assume in the sequel that the *surviving* samples are statistically independent. The N-out-of-M strategy suggested by Li and Mascagni [10] has been designed in such a way that the random numbers generated by *any* N out of M random number streams *are* independent random numbers. The pseudo-random number generators in [11, 19] incorporate this strategy.

For fixed M the approximation $E_M[X]$ is a RV over Ω. For $\hat{M}$ considered as RV over Ω' this approximation becomes

$$E_{\hat{M}(\omega')}[X] := \frac{1}{\hat{M}(\omega')} \sum_{i=1}^{\hat{M}(\omega')} X^i(\omega).$$

Therefore, it is a RV over the product probability space $(\Omega \times \Omega', \mathscr{A} \otimes \mathscr{A}', \mathbb{P} \otimes \mathbb{P}')$. Since $(\Omega, \mathscr{A}, \mathbb{P})$ and since $(\Omega', \mathscr{A}', \mathbb{P}')$ are finite measure spaces, there exists (see, e.g., [22] and the references there) a unique product probability measure $\hat{\mathbb{P}} = \mathbb{P} \otimes \mathbb{P}'$ on $\mathscr{A} \otimes \mathscr{A}'$ such that

$$\hat{\mathbb{P}}(A \otimes A') = \mathbb{P}(A)\mathbb{P}'(A'), \qquad \forall A \in \mathscr{A}, A' \in \mathscr{A}' .$$

Theorem 2.1. *The MC error estimate with a random number of samples is given by*

$$\|\mathbb{E}[X] - E_{\hat{M}(\omega')}[X]\|^2_{L^2(\hat{\Omega}; L^1(\mathbb{R}^d); \hat{\mathbb{P}})} = \mathbb{E}'\left[\frac{1}{\hat{M}(\omega')}\right] \|X\|^2_{L^2(\Omega; L^1(\mathbb{R}^d))}. \tag{3}$$

Proof. We substitute the RV $\hat{M}(\omega')$ for M in equation (2) and integrate over $(\Omega', \mathscr{A}', \mathbb{P}')$ to obtain

$$\begin{aligned}\int_{\Omega'} \|\mathbb{E}[X] - E_{\hat{M}(\omega')}[X]\|^2_{L^2(\Omega; L^1(\mathbb{R}^d))} \mathbb{P}'(d\omega') &= \int_{\Omega'} \frac{1}{\hat{M}(\omega')} \|X\|^2_{L^2(\Omega; L^1(\mathbb{R}^d))} \mathbb{P}'(d\omega') \\ &= \mathbb{E}'\left[\frac{1}{\hat{M}(\omega')}\right] \|X\|^2_{L^2(\Omega; L^1(\mathbb{R}^d))}.\end{aligned}$$

Theorem I.3.17. in [21] then shows that

$$L^2(\Omega'; L^2(\Omega; L^1(\mathbb{R}^d))) \cong L^2(\Omega \times \Omega'; L^1(\mathbb{R}^d); \mathbb{P} \otimes \mathbb{P}'),$$

which leads to the claimed error estimate (3). □

Theorem 2.1 gives an error bound for the MC method with a random number of samples which can be used to deal with node failures. Knowing the failure distribution the term $\mathbb{E}'[1/\hat{M}(\omega')]$ can be computed and hence the error bound for the fault tolerant MC (FT-MC) method as well.

$\hat{M}(\omega') = 0$ means that no samples remain. If there is a positive probability that this happens, then $\mathbb{E}'[1/\hat{M}(\omega')]$ tends to infinity and the FT-MC error bound becomes meaningless. Without loss of generality we therefore ignore the case $\hat{M}(\omega') = 0$. In practice, this means that the MC estimation has to be restarted from scratch in the event that all samples are lost at runtime.

3. Abstract Multilevel Monte Carlo with sample losses

3.1. Review of the Multilevel Monte Carlo method

In this section we consider random variables X that are solutions of problems with random inputs that can be solved only approximately. Prominent examples are PDEs with random initial or boundary conditions as they arise in uncertainty quantification (UQ) in engineering applications. The solutions of these PDEs typically are obtained numerically by a discretization method like the finite element, finite difference, or finite volume method. We assume that we have available a hierarchy of discretizations that is indicated by a measure, e.g., the grid spacing.

In contrast to ordinary Monte Carlo methods, Multilevel Monte Carlo (MLMC) methods can exploit this hierarchy of discretizations. If implemented properly, MLMC provides estimates $E(X)$ of higher accuracy than MC for the same amount of work measured in floating point operations.

A hierarchy of discretizations is given for instance under the assumption that the computational domain is rectangular and is discretized by a hierarchy of regular grids with grid spacings (or "meshwidths")

$$h_0 > \cdots > h_L, \qquad h_i = 2h_{i+1}. \tag{4}$$

Such hierarchies are available to most simulations in engineering. Similar to [7], the difference between $X(\cdot, \omega)$ of the original problem and its discretization $X_h(\cdot, \omega)$ on the mesh of width h, i.e. the discretization error, is assumed to be

$$\|\mathbb{E}[X - X_{h_\ell}]\|_B = O(h_\ell^\alpha). \tag{5}$$

We also assume that the inequalities

$$\|X_{h_\ell} - X_{h_{\ell-1}}\|_{L^2(\Omega;B)} = O(h_{\ell-1}^\beta), \quad \ell > 0, \tag{6}$$

and

$$\|X_{h_0}\|_{L^2(\Omega;B)} = O(1) \tag{7}$$

hold. The first equality is a consequence of the bounded variation of X and of the consistency of the discretization scheme, given the almost sure regularity of the sample paths; the latter equality can always be achieved by a scaling of the data. Furthermore it is assumed that the work needed to compute X_{h_ℓ} satisfies

$$W_\ell = O(h_\ell^{-\gamma}) \,. \tag{8}$$

As explained e.g. in [7, 12–14], the MLMC method is based on the following telescopic sum,

$$\mathbb{E}[X_{h_L}] = \mathbb{E}[X_{h_0}] + \sum_{\ell=1}^{L} \mathbb{E}[X_{h_\ell} - X_{h_{\ell-1}}] \,, \tag{9}$$

that allows to estimate $\mathbb{E}[X]$ *levelwise*,

$$E[X_{h_L}] = E_{M_0}[X_{h_0}] + \sum_{\ell=1}^{L} E_{M_\ell}[X_{h_\ell} - X_{h_{\ell-1}}] \,. \tag{10}$$

On level ℓ we use an ordinary MC method with M_ℓ samples $(X^i_{h_\ell} - X^i_{h_{\ell-1}}), \quad i = 1, \ldots, M_\ell$, to approximate $\mathbb{E}[X_{h_\ell} - X_{h_{\ell-1}}]$. Note that in order to form these differences the realizations $X^i_{h_\ell}$ and $X^i_{h_{\ell-1}}$ have to be computed with the same ω. On the coarsest level we estimate $\mathbb{E}[X_{h_0}]$ with M_0 samples $X^i_{h_0}, \quad i = 1, \ldots, M_0,$. The MLMC error $\|\mathbb{E}[X] - E[X_{h_L}]\|_{L^2(\Omega;B)}$ is the norm of the difference of the true expectation $\mathbb{E}[X]$ in (9) and the MLMC estimate $E[X_{h_L}]$ in (10),

$$\begin{aligned}
\|\mathbb{E}[X] - E[X_{h_L}]\|_{L^2(\Omega;B)} &\leq \|\mathbb{E}[X] - \mathbb{E}[X_{h_L}]\|_B + \|\mathbb{E}[X_{h_L}] - E[X_{h_L}]\|_{L^2(\Omega;B)} \\
&\leq \|\mathbb{E}[X] - \mathbb{E}[X_{h_L}]\|_B + \|\mathbb{E}[X_{h_0}] - E_{M_0}[X_{h_0}]\|_{L^2(\Omega;B)} \\
&\qquad + \sum_{\ell=1}^{L} \|\mathbb{E}[X_{h_\ell} - X_{h_{\ell-1}}] - E_{M_\ell}[X_{h_\ell} - X_{h_{\ell-1}}]\|_{L^2(\Omega;B)} \\
&\overset{(2)}{\leq} \|\mathbb{E}[X] - \mathbb{E}[X_{h_L}]\|_B + M_0^{-1/2}\|X_{h_0}\|_{L^2(\Omega;B)} \\
&\qquad + \sum_{\ell=1}^{L} M_\ell^{-1/2}\|X_{h_\ell} - X_{h_{\ell-1}}\|_{L^2(\Omega;B)}.
\end{aligned} \tag{11}$$

With (6) and (7) this bound becomes

$$\|\mathbb{E}[X] - E[X_{h_L}]\|_{L^2(\Omega;B)} = O(h_L^\alpha) + M_0^{-1/2}O(1) + \sum_{\ell=1}^{L} M_\ell^{-1/2} O(h_{\ell-1}^\beta). \tag{12}$$

This result is valid for any numbers L and M_ℓ. However, to benefit from MLMC, a clever choice is crucial. One may balance the expected error terms on the right side of (12) [12] or minimize the computational work with respect to the expected error, see [7]. In either case, when expressed in terms of work, the convergence rates of MLMC are always as good as the ones from standard MC methods. Giles [7] shows that for $\alpha \geq 1/2$ and any $\epsilon < e^{-1}$ there is an L and values M_ℓ, $0 \leq \ell \leq L$, such that the MLMC (discretization and sampling) error is bounded by

$$\|\mathbb{E}[X] - E[X_{h_L}]\|_{L^2(\Omega;B)} < \epsilon.$$

with the total work for computing $E[X_{h_L}]$ being given by

$$\text{work} = \begin{cases} O(\epsilon^{-2}), & \beta > 1, \\ O(\epsilon^{-2}(\log \epsilon)^2), & \beta = 1, \\ O(\epsilon^{-2-(1-\beta)/\alpha}), & 0 < \beta < 1. \end{cases}$$

3.2. Sample losses in MLMC

In the previous section 2 we considered the MC method with a random number of samples, given the realistic assumption that at least one MC-sample survives. Within each level of the MLMC method a standard MC simulation is executed. This allows to reuse most parts of the FT-MC approach in the FT-MLMC method. In the FT-MLMC method, in contrast to the FT-MC

method, it is feasible to compute an MLMC estimate even in the case that all samples of one level $\ell \in [0, L]$ are lost. A level ℓ of mesh-refinement on which all MC samples are lost will be referred to as a *lost level*. In MLMC $\hat{M}_\ell(\omega') = 0$ on some level ℓ should not lead to an infinite error bound. Therefore, the MLMC error bound has to be modified such that it can handle lost levels.

3.3. MLMC error bound with lost levels

We derive an MLMC error bound under assumption that entire levels are lost. In *the discussion of this section, the indices of lost levels are not random.*

Let $\widetilde{\mathcal{L}} = \{0, ..., L\}$ be the set of all levels in an MLMC simulation. The set of all lost levels is denoted by $\mathcal{L}^* = \{\ell | \ell \text{ is lost}\}$. The set of all levels with at least one sample surviving is denoted by $\mathcal{L} = \widetilde{\mathcal{L}} \backslash \mathcal{L}^*$. The MLMC estimate (10) is modified such that lost levels are not taken into account,

$$E^{\mathcal{L}}[X_{h_L}] = \chi_{\mathcal{L}}(0) \cdot E_{M_0}[X_{h_0}] + \sum_{\ell \in \mathcal{L} \backslash \{0\}} E_{M_\ell}[X_{h_\ell} - X_{h_{\ell-1}}], \tag{13}$$

the error bound

$$\|\mathbb{E}[X] - E^{\mathcal{L}}[X_{h_L}]\|_{L^2(\Omega;B)}$$

is analyzed.

By the triangle inequality the error bound yields

$$\|\mathbb{E}[X] - E^{\mathcal{L}}[X_{h_L}]\|_{L^2(\Omega;B)} \leq \|\mathbb{E}[X] - E^{\widetilde{\mathcal{L}}}[X_{h_L}]\|_{L^2(\Omega;B)} + \|E^{\widetilde{\mathcal{L}}}[X_{h_L}] - E^{\mathcal{L}}[X_{h_L}]\|_{L^2(\Omega;B)}.$$

The first is the standard MLMC error in (11). The second term equals

$$\begin{aligned}
&\|E_{M_0}[X_{h_0}] + \sum_{\ell=1}^{L} E_{M_\ell}[X_{h_\ell} - X_{h_{\ell-1}}] \\
&\qquad - \left(\chi_{\mathcal{L}}(0) \cdot E_{M_0}[X_{h_0}] + \sum_{\ell \in \mathcal{L} \backslash \{0\}} E_{M_\ell}[X_{h_\ell} - X_{h_{\ell-1}}] \right) \|_{L^2(\Omega;B)} \\
&= \|\chi_{\mathcal{L}^*}(0) \cdot E_{M_0}[X_{h_0}] + \sum_{\ell \in \mathcal{L}^* \backslash \{0\}} E_{M_\ell}[X_{h_\ell} - X_{h_{\ell-1}}]\|_{L^2(\Omega;B)} \\
&\leq \chi_{\mathcal{L}^*}(0) \cdot E_{M_0}[\|X_{h_0}\|_{L^2(\Omega;B)}] + \sum_{\ell \in \mathcal{L}^* \backslash \{0\}} E_{M_\ell}[\|X_{h_\ell} - X_{h_{\ell-1}}\|_{L^2(\Omega;B)}] \\
&= \chi_{\mathcal{L}^*}(0) \cdot \|X_{h_0}\|_{L^2(\Omega;B)} + \sum_{\ell \in \mathcal{L}^* \backslash \{0\}} \|X_{h_\ell} - X_{h_{\ell-1}}\|_{L^2(\Omega;B)}.
\end{aligned}$$

This inequality admits to bound the MLMC error where some levels are lost.

Theorem 3.1. *The MLMC error with lost levels is bounded by*

$$\begin{aligned}
\|\mathbb{E}[X] - E^{\mathcal{L}}[X_{h_L}]\|_{L^2(\Omega;B)} \leq \|\mathbb{E}[X] - E^{\widetilde{\mathcal{L}}}[X_{h_L}]\|_{L^2(\Omega;B)} \\
+ \chi_{\mathcal{L}^*}(0) \cdot \|X_{h_0}\|_{L^2(\Omega;B)} + \sum_{\ell \in \mathcal{L}^* \backslash \{0\}} \|X_{h_\ell} - X_{h_{\ell-1}}\|_{L^2(\Omega;B)},
\end{aligned}$$

or, with assumptions (5), (6) and (7),

$$\|\mathbb{E}[X] - E^{\mathcal{L}}[X_{h_L}]\|_{L^2(\Omega;B)} \leq O(h_L^\alpha) + (M_0^{-1/2} + \chi_{\mathcal{L}^*}(0))O(1) + \sum_{\ell=1}^{L} (M_\ell^{-1/2} + \chi_{\mathcal{L}^*}(\ell))O(h_{\ell-1}^\beta) \,.$$

Losing a level increases the MLMC error bound in Theorem 3.1 substantially. This applies particularly for the low levels where costs of losing them are highest, since losing level ℓ leads to an error $O(h_{\ell-1}^{\beta})$. Remark that higher levels than the lost one will not improve the order of the error anymore.

3.4. Description of the failure space and its random variables

We work under the "all or nothing paradigm", i.e. a sample is either correctly computed or irrecoverably lost when nodes fail at runtime. We do not distinguish between node, program, network, or any other cause of failure at runtime. We discard all samples affected by a failure, and compute the result with the remaining ones. Then M_ℓ the number of samples per level is not a fixed number anymore, but a random number. A level where all samples are lost will be referred to as a "lost level". One is free to choose any other criterion to exclude a level, for example if more than 90% of the samples are lost. For simplicity we only consider the initial definition in this paper. In the MLMC estimate in equation (13) lost levels are neglected in the computation. Therefore, the event "level ℓ is used" resp. "level ℓ is lost" is a random event.

In accordance with Section 2 the probability space for the random solution X is denoted as $(\Omega, \mathscr{A}, \mathbb{P})$. The probability space to model runtime failures is denoted as $(\Omega', \mathscr{A}', \mathbb{P}')$. Evidently, the number of MC samples per discretization level $\hat{M}_\ell$ depends on the runtime failures.

All RVs belonging to the space $(\Omega', \mathscr{A}', \mathbb{P}')$ are defined as follows:

Definition 3.1. (RV on $(\Omega', \mathscr{A}', \mathbb{P}')$) *The number of samples per level $\hat{M}_\ell$ is a RV in the probability space $(\Omega', \mathscr{A}', \mathbb{P}')$ and depends on the failures in the computer, i.e.,*

$$\hat{M}_\ell : (\Omega', \mathscr{A}', \mathbb{P}') \to (\mathbb{N}_0, 2^{\mathbb{N}_0}).$$

The set of all surviving levels in the MLMC estimate is a random set

$$\widehat{\mathcal{L}} : (\Omega', \mathscr{A}', \mathbb{P}') \to (2^{\widetilde{\mathcal{L}}}, 2^{\widetilde{\mathcal{L}}})$$

which is defined by

$$\widehat{\mathcal{L}}(\omega') = \{\ell | \hat{M}_\ell(\omega') > 0\},$$

the random set of all levels ℓ in which at least one draw of the discretization at level ℓ has survived. Obviously, $\widehat{\mathcal{L}}(\omega')$ is a measurable map and, hence, a RV obtaining values in $2^{\widetilde{\mathcal{L}}}$. The RV $\hat{M}_\ell^*$ is defined as

$$\hat{M}_\ell^* : (\Omega', \mathscr{A}', \mathbb{P}') \to (\mathbb{N}, 2^{\mathbb{N}}),$$

and depends on $\hat{M}_\ell$, $\chi_{\widehat{\mathcal{L}}}(\ell)$ and on the fixed $a \in \mathbb{N}$

$$\hat{M}_\ell^* = \begin{cases} \hat{M}_\ell, & \text{if } \chi_{\widehat{\mathcal{L}}}(\ell) = 1, \\ a, & \text{otherwise}, \end{cases}$$

where values of a are freely chosen (e.g. $a \to \infty$), since a is only applied if the level is lost.

3.5. Fault Tolerant MLMC error bound with random failures

We derive an error bound for an MLMC sample average estimate which is computed based on a random numbers of samples which models the MLMC in the presence of failures at runtime. Specifically, the following fault tolerant MLMC (FT-MLMC) estimator is used:

$$\hat{E}^{\widehat{\mathcal{L}}}[X_{h_L}] = \chi_{\widehat{\mathcal{L}}(\omega')}(0) \cdot E_{M_0}[X_{h_0}] + \sum_{\ell \in \widehat{\mathcal{L}}(\omega') \setminus \{0\}} E_{\hat{M}^*_\ell(\omega')}[X_{h_\ell} - X_{h_{\ell-1}}] \, .$$

Theorem 3.2. *The FT-MLMC error under influence of failure is bounded by*

$$\begin{aligned}\|\mathbb{E}[X] - \hat{E}^{\widehat{\mathcal{L}}}[X_{h_L}]\|_{L^1(\Omega';L^2(\Omega;B))} \leq \int_{\Omega'} \|\mathbb{E}[X] - E^{\widetilde{\mathcal{L}}}[X_{h_L}]\|_{L^2(\Omega;B)} \mathbb{P}'(d\omega') + \mathbb{E}'[1 - \chi_{\widehat{\mathcal{L}}}(0)]\|X_{h_0}\|_{L^2(\Omega;B)} \\ + \sum_{\ell=1}^{L} \mathbb{E}'[1 - \chi_{\widehat{\mathcal{L}}}(\ell)]\|X_{h_\ell} - X_{h_{\ell-1}}\|_{L^2(\Omega;B)} \, .\end{aligned}$$

In the error bound for $\|\mathbb{E}[X] - E^{\widetilde{\mathcal{L}}}[X_{h_L}]\|_{L^2(\Omega;B)}$ the number of samples per levels is given by $\hat{M}^*_\ell(\omega')$. This is the only dependence on ω' in this error bound.

Or given the assumptions (5), (6) and (7) and by choosing $a \to \infty$ in M^*_ℓ

$$\begin{aligned}&\|\mathbb{E}[X] - \hat{E}^{\widehat{\mathcal{L}}}[X_{h_L}]\|_{L^1(\Omega';L^2(\Omega;B))} \\ &\quad \leq O(h_L^\alpha) + \mathbb{E}'[\hat{M}_0^{-1/2}\chi_{\widehat{\mathcal{L}}}(0) + (1 - \chi_{\widehat{\mathcal{L}}}(0))]O(1) + \sum_{\ell=1}^{L} \mathbb{E}'[\hat{M}_\ell^{-1/2}\chi_{\widehat{\mathcal{L}}}(\ell) + (1 - \chi_{\widehat{\mathcal{L}}}(\ell))]O(h_{\ell-1}^\beta) \, .\end{aligned}$$

Proof. By Theorem 3.1 we have

$$\begin{aligned}\|\mathbb{E}[X] - \hat{E}^{\widehat{\mathcal{L}}}[X_{h_L}]\|_{L^2(\Omega;B)} \leq \|\mathbb{E}[X] - E^{\widetilde{\mathcal{L}}}[X_{h_L}]\|_{L^2(\Omega;B)} \\ + (1 - \chi_{\widehat{\mathcal{L}}}(0))\|X_{h_0}\|_{L^2(\Omega;B)} + \sum_{\ell=1}^{L}(1 - \chi_{\widehat{\mathcal{L}}}(\ell))\|X_{h_\ell} - X_{h_{\ell-1}}\|_{L^2(\Omega;B)}.\end{aligned}$$

Both sides are integrated over the probability space $(\Omega', \mathscr{A}', \mathbb{P}')$ leading to

$$\begin{aligned}&\int_{\Omega'} \|\mathbb{E}[X] - \hat{E}^{\widehat{\mathcal{L}}}[X_{h_L}]\|_{L^2(\Omega;B)} \mathbb{P}'(d\omega') \\ &\qquad \leq \int_{\Omega'} (\|\mathbb{E}[X] - E^{\widetilde{\mathcal{L}}}[X_{h_L}]\|_{L^2(\Omega;B)} \\ &\qquad\qquad + (1 - \chi_{\widehat{\mathcal{L}}}(0))\|X_{h_0}\|_{L^2(\Omega;B)} + \sum_{\ell=1}^{L}(1 - \chi_{\widehat{\mathcal{L}}}(\ell))\|X_{h_\ell} - X_{h_{\ell-1}}\|_{L^2(\Omega;B)})\mathbb{P}'(d\omega').\end{aligned}$$

With the linearity of the expected value operator and by factorization the error estimate becomes

$$\begin{aligned}&\int_{\Omega'} \|\mathbb{E}[X] - \hat{E}^{\widehat{\mathcal{L}}}[X_{h_L}]\|_{L^2(\Omega;B)} \mathbb{P}'(d\omega') \\ &\quad \leq \int_{\Omega'} \|\mathbb{E}[X] - E^{\widetilde{\mathcal{L}}}[X_{h_L}]\|_{L^2(\Omega;B)} \mathbb{P}'(d\omega') \\ &\qquad + \int_{\Omega'} (1 - \chi_{\widehat{\mathcal{L}}}(0))\mathbb{P}'(d\omega')\|X_{h_0}\|_{L^2(\Omega;B)} + \sum_{\ell=1}^{L} \int_{\Omega'} (1 - \chi_{\widehat{\mathcal{L}}}(\ell))\mathbb{P}'(d\omega')\|X_{h_\ell} - X_{h_{\ell-1}}\|_{L^2(\Omega;B)} \, .\end{aligned}$$

In the MLMC estimate $E^{\widetilde{\mathcal{L}}}[X_{h_L}]$ the number of samples per levels is a parameter. For all levels with at least one sample surviving this number is given by $\hat{M}_\ell(\omega')$. For lost levels there are no restrictions on the number of samples in $E^{\widetilde{\mathcal{L}}}[X_{h_L}]$. The random variable $\hat{M}^*_\ell(\omega')$ represents this setting and is therefore used as the number of samples per level in the MLMC estimate $E^{\widetilde{\mathcal{L}}}[X_{h_L}]$. □

We derived a general fault tolerant MLMC method. Regarding the implementation some aspects should be considered. A (single level) MC method is used to estimate $\mathbb{E}[X_{h_\ell} - X_{h_{\ell-1}}]$ by $E_{\hat{M}^*_\ell(\omega')}[X_{h_\ell} - X_{h_{\ell-1}}] = \sum_{i=1}^{\hat{M}^*_\ell(\omega')}(X^i_{h_\ell} - X^i_{h_{\ell-1}})$, where $X^i_{h_\ell}$ and $X^i_{h_{\ell-1}}$ approximate the same realization X^i with two different discretization parameters h_ℓ and $h_{\ell-1}$. This implies a strong statistical correlation between $X^i_{h_\ell}$ and $X^i_{h_{\ell-1}}$ (in general the same random numbers are used in $X^i_{h_\ell}$ and $X^i_{h_{\ell-1}}$). Failures influence the random number of samples per level $\hat{M}^*_\ell(\omega')$. It is emphasized that a sample on a level always consists of the difference between the two computed solutions $(X^i_{h_\ell} - X^i_{h_{\ell-1}})$. In other words whenever a $X^i_{h_{\ell-1}}$ is lost due to a failure, it is required that the corresponding $X^i_{h_\ell}$ is disregarded as well, and vice versa. We propose therefore to do all computations for a sample $X^i_{h_\ell} - X^i_{h_{\ell-1}}$ on a single unit. A failure in this unit leads automatically to a loss of both $X^i_{h_\ell}$ and $X^i_{h_{\ell-1}}$. Only entirely computed samples $X^i_{h_\ell} - X^i_{h_{\ell-1}}$ are accumulated or communicated to other units. This procedure has the slight disadvantage that two different discretizations have to be computed on the same unit.

4. Statistical model of node failure at runtime

System failures can interfere with the computation of the MLMC estimate. They can be due to errors in software, hardware or network but also due to environmental effects. These failures are manifestations of various types of errors which can be roughly classified as permanent, transient, or silent [4]. Permanent errors do not disappear, whereas transient errors are short term errors. Silent errors are undetected (permanent or transient) errors, and hence they may lead to undetected erroneous results. Other classifications into hard and soft errors are conceivable and for instance described in [8].

In the previous section we have extended the MLMC theory to cover random numbers of samples. This allows to disregard all samples affected by detected errors (permanent or transient, soft or hard). Our fault tolerant MLMC method simply ignores these lost samples and tries to make the best out of the surviving ones. Silent (undetected) errors however are not covered by this theory.

To specify the error bound of the FT-MLMC method the statistics of failures leading to sample losses has to be known. For new HPC systems first empirical statistical failure studies [15, 20] are available. Schroeder and Gibson [20] derived a realistic failure model that has been adopted in the present paper. The authors studied the failures which occurred over 9 years in more than 20 different systems at Los Alamos National Laboratory. Their raw data [25] contains an entry for any failure that required the attention of a system administrator. The authors conclude that the time interval between two failures of an individual nodes, as well as for the entire system is well fitted by the *Weibull distribution* with the Weibull shape parameter k between 0.7 and 0.8. The probability density function of the Weibull random variable x is given by (e.g. [16])

$$f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k}, & \text{if } x \geq 0, \\ 0, & \text{if } x < 0, \end{cases}$$

where $\lambda > 0$ is the scale parameter and $k > 0$ is the shape parameter. Schroeder and Gibson also found the failure rates to be roughly proportional to the number of processors in the system. Therefore, we use the Weibull distribution to model the time between two consecutive failures on one node. We assume that node failures are statistically independent.

4.1. Time to next failure in a renewal process

When a simulation on a computer is started, we do in general not know when this computer failed last. Thus, we are not only interested in the time between two failures but also in the distribution of the time from the start of the computation until the first failure on a node. This time is modeled with an ordinary renewal process (RP). An ordinary RP is a sequence of i.i.d. nonnegative RVs $Y_1, Y_2, \ldots$ In our case these RVs are the time intervals between two failures, hence they are Weibull distributed. After a failure the failed component is renewed within a negligible time. Therefore, the time to the next failure is again Weibull distributed. This ordinary RP is started at $t = 0$. We are interested in the so called forward recurrence time till the next failure. F_t is the time from t up to and including the moment of the next failure. The MLMC computation is started at time t. Rinne [18] states that, as long as t is finite, F_t can only be evaluated numerically. In the case $t \to \infty$ however [18, eq. (4.41a)] provides the analytic distribution function of F_∞.

A procedure to draw realizations of the forward recurrence time F_∞ is given in [24]:

1. Generate the random variable S from $S = \Gamma(1 + \frac{1}{k}, \lambda^k)$, where λ and k are the Weibull scale and shape parameter, respectively.
2. Compute the random variable V as $V = S^{1/k}$.
3. Generate the random variable F_∞ from $F_\infty = \text{Uniform}(0, V)$.

4.2. Estimate the number of samples per level with MC

A realization of the number $\hat{M}_\ell$ of samples on level ℓ is drawn by drawing realization of the first failure F_∞ on a node where samples of level ℓ are computed. The term $\mathbb{E}'[\hat{M}_\ell^{-1/2}\chi_{\widehat{\mathcal{L}}}(\ell) + (1 - \chi_{\widehat{\mathcal{L}}}(\ell))]$ is then estimated by MC, i.e., by averaging over the different realizations of $\hat{M}_\ell$.

4.3. Calibration of parameters

The Weibull scale and shape parameters λ and k are used as parameters in the drawing of forward recurrence times F_∞. They depend on the machinery the FT-MLMC method runs on. In [18] several methods are presented to estimate both Weibull parameters

In our analysis we use the parameter given in [20], $k \approx 0.7$ and $\lambda \approx 7.6 \cdot 10^5$ for a single node. Note that $k \approx 0.7$ is explicitly given in [20] for an individual node. λ is estimated from [20, Fig. 6(b)]. The time between failures, such that the cumulative distribution function is equal to 0.5, is given approximatively by $x \approx 4.5 \cdot 10^5$s. Therefore, λ can be approximated by $0.5 \approx 1 - e^{-(4.5 \cdot 10^5/\lambda)^k}$. This leads to $\lambda \approx 7.6 \cdot 10^5$.

These parameters lead to a mean time between failure (MTBF) of approximately 11 days for a typical node. We use nodes with 128 cores, as in [20] a node has 80–128 cores. Whenever a node is hit by a failure, results of all involved 128 cores are assumed to be irrevocably lost.

5. Numerical experiments

All MLMC methods (which satisfy assumptions (5)–(8)) are suited for the FT-MLMC method. We assess the quality of our FT-MLMC error bounds by means of the grid-based finite volume code ALSVIDUQ [1] for solving hyperbolic systems of conservation laws. We set $\alpha = \beta = s$, see eqs. (5)–(6). With this choice, the work to compute a sample on level ℓ is

$$W_\ell = 2^{d+1} W_{\ell-1} = 2^{(d+1)\ell} W_0, \qquad \ell > 0, \tag{14}$$

where the exponent originates from d space and 1 time dimension. Note that samples on low levels require a small fraction of execution time compared with samples on high levels. The same holds for memory space; memory usage of a grid-based PDE solver is given by

$$\mathrm{mem}_\ell = 2^d \mathrm{mem}_{\ell-1} = 2^{d\ell} \mathrm{mem}_0, \qquad \ell > 0. \tag{15}$$

As suggested in [12] for $s < (d+1)/2$, we set M_ℓ, the number of samples on level ℓ, to be

$$M_\ell = 2^{-2s} M_{\ell-1} = 2^{2(L-\ell)s} M_L, \qquad \ell > 0. \tag{16}$$

On the finest level, we chose a moderate number of samples, e.g., $M_L = 20$.

We parallelize our simulation by levels and across levels. A core deals with samples of only one level.

There is an intermediate level b whose samples are executed on one core. This level is arbitrary to some extent. It is determined typically based on memory consumption or execution times. On levels $\ell > b$, a single sample is executed in parallel on multiple cores. The number of cores is determined by the memory requirement of the sample, as given in equation (15). On levels $\ell \leq b$ samples are executed sequentially. So, the number N_ℓ of cores invested for a sample on level ℓ is

$$N_\ell = \lceil 2^{d(\ell-b)} \rceil, \qquad \ell > 0. \tag{17}$$

We collect samples in tasks of equal execution times. Tasks are the units of work that are submitted to the compute nodes. A sample of the finest level forms a task. Tasks of lower levels consist of multiple samples. On very coarse levels, all respective samples are included in a single task that may have a shorter execution time than (most) other tasks. Tasks of the same level are always computed on independent nodes. A node may compute several tasks from different levels.

In Fig. 1 a 3D example is given with 4 levels, $b = 1$, $M_L = 4$, and $s = 1/2$. The MLMC estimate is computed in parallel on 293 cores. The 4 samples of level L are each computed in a task of 64 cores. Hence, 256 cores are involved in the computation of this level. The number of cores in a task is one on level $b = 1$. This number increases by 8 towards the finer levels, recording (17). On level 0, one core deals with a single task that comprises all 32 samples of this level. The execution time of this task is approximately half of the others, causing some small load imbalance.

5.1. Lowering risk of level loss

A high probability of losing entire levels increases the FT-MLMC error bound in Theorem 3.2 substantially. This applies particularly for the coarse levels where the impact of a loss is highest. Two different ways of computing the samples are investigated. Both lower the risk of losing coarse levels:

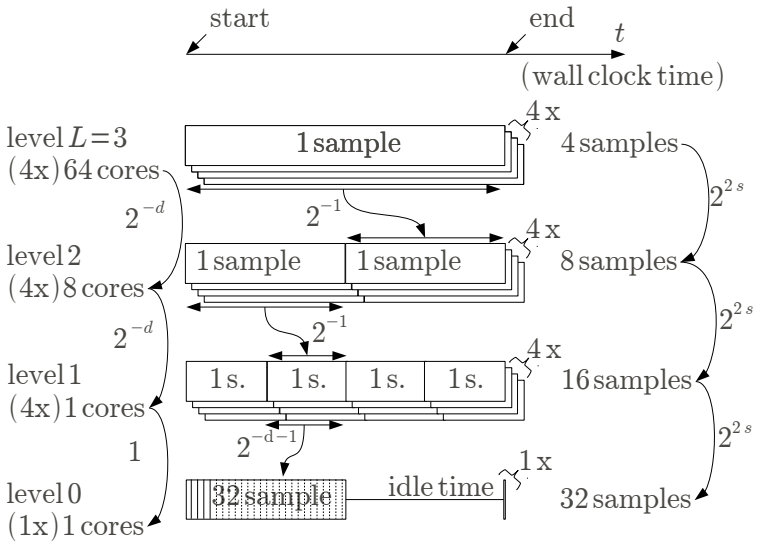


Figure 1: Computing an MLMC estimate with parameters $L = 3$, $M_L = 4$, $d = 3$, $s = 1/2$ and only one core per task on levels 0 and 1.

- "late save/4 tasks": The samples of a level are split into at least four tasks each of which is computed on independent nodes. The samples are stored only after the completion of the whole task. So, according to our assumptions, a failure at run-time of a task leads to the total loss of all its samples. In the example shown in Fig. 1 only level 0 is affected as all other levels already run on at least four cores, see Fig. 2. This strategy reduces the risk of

Figure 2: The "late save/4 tasks" strategy requests at least 4 tasks per level. In the example of Fig. 1 this leads to a large number of very small tasks on level 0.

losing a level 0 by reducing the execution time of a task as well as by using multiple tasks. The execution time of the now 4 tasks is reduced by 4 compared to the original one which increases load imbalance.

- "immediate save": Immediately after the completion of a sample, it is safely stored, and hence will not get lost anymore.

The two strategies have a different behavior in case of failure. In the "immediate save" strategy, a larger part of the data on a node survives the failure of the node, leading to higher failure resilience. This advantage is offset by higher communication overhead. In the "late save" strategy the statistical quality of the remaining samples is equivalent to the N-out-of-M strategy suggested by Li and Mascagni [10]. How to get statistical independence of the remaining samples in the "immediate save" strategy is to our knowledge an open question. Then only the first entries (up to the failure) of a random number stream are used. Also the resilience of massive parallel RNG (such as WELL [9]) in the presence of partial loss of streams remains to be addressed.

5.2. Assessment of the FT-MLMC error bound

The error bound is compared with the measured error of a FT-MLMC method. We chose a finite volume method (FT-MLMC-FVM) that solves the 1D Euler equations of gas dynamics. Following [13, 23], the 1D Euler equations of gas dynamics are given by

$$\begin{cases} \rho_t + \operatorname{div}(\rho \mathbf{u}) = 0, \\ (\rho \mathbf{u})_t + \operatorname{div}(\rho \mathbf{u} \otimes \mathbf{u} + p\mathbf{ID}) = 0, \\ E_t + \operatorname{div}((E+p)\mathbf{u}) = 0, \end{cases}$$

in $D = (0,1)$, with outflow boundary conditions, where ρ is the density and $\mathbf{u}$ the velocity vector. The pressure p and the total energy E are related by the ideal gas equation of state

$$E := \frac{p}{\gamma - 1} + \frac{1}{2}\rho|\mathbf{u}|^2,$$

with γ the ratio of specific heats. The RV of interest is $\mathbf{X}(x,t,\omega) = \{\rho(x,t,\omega), u(x,t,\omega), p(x,t,\omega)\}$, computed at $t = .5$. A random initial shock with uncertain location is used as initial condition

$$\mathbf{X}_0(x,\omega) = \{\rho_0(x,\omega), u_0(x,\omega), p_0(x,\omega)\} = \begin{cases} \{3.0, 0.0, 3.0\} & \text{if} \quad x < Y(\omega), \\ \{1.0, 0.0, 1.0\} & \text{if} \quad x > Y(\omega). \end{cases}$$

In [13] an MLMC Finite Volume Method (MLMC-FVM) approach is proposed to estimate $\mathbb{E}[\mathbf{X}(\cdot,t)]$. An MLMC-FVM error bound is proved for scalar solutions in [12] and assumed for systems in [13, 14] by

$$\|\mathbb{E}[\mathbf{X}(\cdot,t)] - E^{\hat{\mathcal{L}}}[\mathbf{X}(\cdot,t)]\|_{L^2(\Omega;L^1(\mathbb{R}^d))} \leq C_1 h_L^s + C_2 \hat{M}_0^{*-\frac{1}{2}} + C_3 \left\{ \sum_{\ell=1}^{L} \hat{M}_\ell^{*-\frac{1}{2}} h_{\ell-1}^s \right\}. \tag{18}$$

This term corresponds to the error bound (12). As suggested in [12] we set $C_1 = C_3$ and $C_2 = C_3 h_0^s$. The convergence rate

$$\|\mathbb{E}[\mathbf{X}(\cdot,t)] - E^{\mathcal{L}}[\mathbf{X}(\cdot,t)]\|_{L^2(\Omega;L^1(\mathbb{R}^d))} \lesssim W^{-s/(d+1)} \cdot \log(W), \quad \text{if } s < (d+1)/2$$

is empirically shown in [13]. Here, $W = \sum W_\ell M_\ell$ is the total work. Samples on level ℓ are bounded by [12–14]

$$\|\mathbf{X}_\ell(\cdot,t) - \mathbf{X}_{\ell-1}(\cdot,t)\|_{L^2(\Omega;L^1(\mathbb{R}^d))} \leq C_4 h_{\ell-1}^s \,. \tag{19}$$

This bound is related to assumption (6). Inequalities (18) and (19) imply the validity of Theorems 3.1 and 3.2 for the FT-MLMC-FVM method applied to the 1D Euler equation of gas dynamics. Further examples, in particular in higher dimensions, can be found in [12–14].

Our finite volume code ALSVIDUQ [1] is sequentially executed on a Computer with 8 GB RAM and with an Intel(R) Core(TM) i5 processor at 2.67 GHz. As a reference, the work W_ℓ for $s = 1/2$ on the level ℓ corresponding to 512 grid-points was 0.043 s.

The FT-MLMC results are compared with an MLMC reference solution with 12 levels and $s = 1$. The finest level had a resolution of 32768 cells. The number of samples on this level was 16.

In the present numerical study, node failures were simulated. The numbers of surviving samples $\hat{M}_\ell$ for each level were drawn from the renewal process introduced in Section 4.1. The simulation

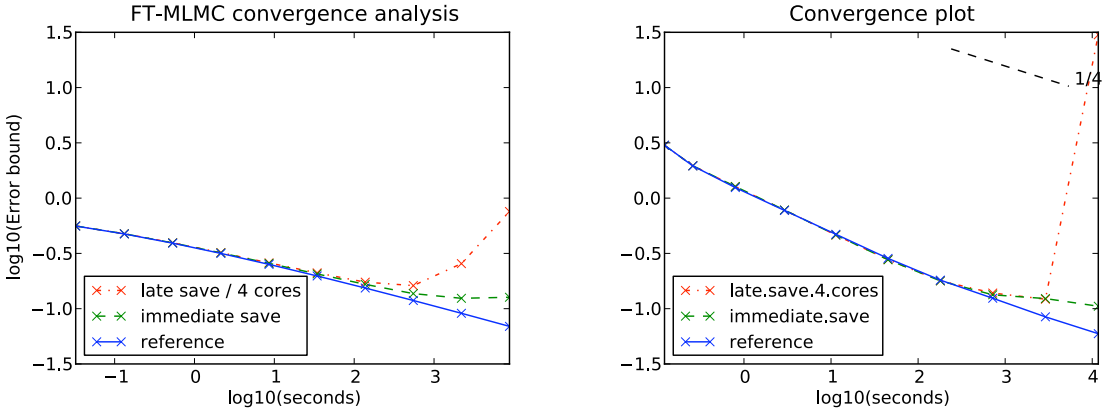
on level ℓ was then executed with $\hat{M}_\ell$ realizations. It gos with out saying that we do not consider our simulator to fail. An assumption verified by repeatedly computing the result.

In order to save computing time we increased the error rate, i.e., the Weibull scale parameter λ, by the factor $5 \cdot 10^6$. Only then failures are frequent enough to be observed with relatively low computation times.

Fig. 3 (a) shows the FT-MLMC-FVM error bound while Fig. 3 (b) shows the measured error with the parameters described in Table 1. Note that longer simulations imply more levels. We

M_L	20
d; s	1; 1/2
# cells on level 0 ($\propto 1/h_0$)	4
W_7 (time for one sample)	0.043s
simulations with levels	$L = 4, 5, \ldots, 13$
one core per task	$b = 4$
Weibull parameters λ; k	$\frac{7.6 \cdot 10^5}{5 \cdot 10^6}$; 0.7

Table 1: The parameters used in Fig. 3.



(a) Error bound (in Theorem 3.2).

(b) Measured error.

Figure 3: 1D Euler equation of gas dynamics with uncertain initial data solved by FT-MLMC-FVM.

are (mostly) interested in the point where the FT-MLMC "late save/4 tasks" and the "immediate save" graph start to deviate from the fault-free MLMC "reference" graph. From there on the performance of the FT-MLMC method drops significantly. This point is clearly similar in both the error bound as well as in the actual error shown in Fig. 3.

Other quantities of lower importance do not match well for several reasons. First, the constants C_1, C_2, and C_3 in the error bound are unknown or can only be very conservatively estimated. The weight of these constants influences the weight of the different error contributions. Second, we are comparing two different things, an error bound with an error. Already the standard fault-free MLMC "reference" shows a considerable difference between the error bound and the error. It seems that the "reference" shows pre-asymptotic convergence behaviour for a run-time below 100 s. After this point the theoretical convergence rate of 1/4 (this slop is shown in the upper right corner) appears to hold. Furthermore, the computed and measured total execution time (number of cores

$\times$ execution time) in the y-axis differ.

The "late save/4 tasks" error does not only stop decreasing but grows again for large computation times, in Fig. 3 around 10^3sec.. At first this may surprise. As the number of levels increases the runtime of all tasks increases. This allows to compute more and more samples of a fixed level in the same task. Hence the failure probability of this task increases. In the "late save/4 tasks" mode failures lead to the loss of all samples of a task. Therefore the probability of level losses not only increase for the high levels, but even for medium and low levels.

5.3. Analysis of FT-MLMC error bounds

In a standard MLMC implementation with MPI-2.1 process failures are fatal. A failure terminates the entire computation. We show that it is possible to compute much bigger problems with the FT-MLMC with the original MLMC convergence rate still being maintained. We assume that MPI has some failure resilience as implemented for instance in FT-MPI [5]. In particular we assume that the MPI processes that are not affected by a failure, can continue working and communicating among each other.

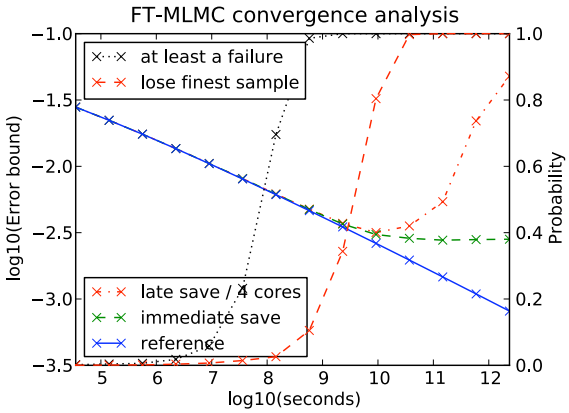
Besides the FT-MLMC error bound, we investigate two additional quantities. First, the probability that a standard MLMC implementation with MPI-2.1 terminates without issuing a result, labeled as "MPI-2.1 terminates". Here, we allow tasks of the same level to be executed on the same node, as in a standard MLMC implementation. The second quantity is the probability that a individual sample on the finest level gets lost. The two quantities are labeled "MPI-2.1 terminates" and "lose finest sample", respectively.

	(a)	(b)	(c)
M_L	20	20	20
d; s	1; 1/2	3; 1/2	3; 1
# cells on level 0 ($\propto 1/h_0$)	512	$8 \times 8 \times 8$	$8 \times 8 \times 8$
W_0 (time for one sample)	0.043s	0.01s	0.1s
simulations with levels	$L = 7, 8, \cdots, 20$	$L = 4, 5, \cdots, 11$	$L = 3, 4, \cdots, 10$
one core per task	$b = 6$	$b = 5$	$b = 4$
Weibull parameters λ; k	$7.6 \cdot 10^5$; 0.7	$7.6 \cdot 10^5$; 0.7	$7.6 \cdot 10^5$; 0.7

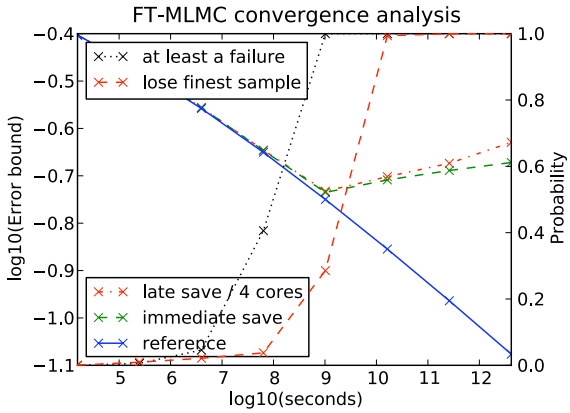
Table 2: The parameters used in Fig. 4 (a) - (c).

Three error bounds with different parameters are analyzed, see Table 2. The three error bounds are shown in Fig. 4. Sub-fig. (a) corresponds to the 1D FT-MLMC-FVM case discussed in the previous subsection. (b) and (c) correspond to a 3D FT-MLMC-FVM, derived from the settings described in [13].

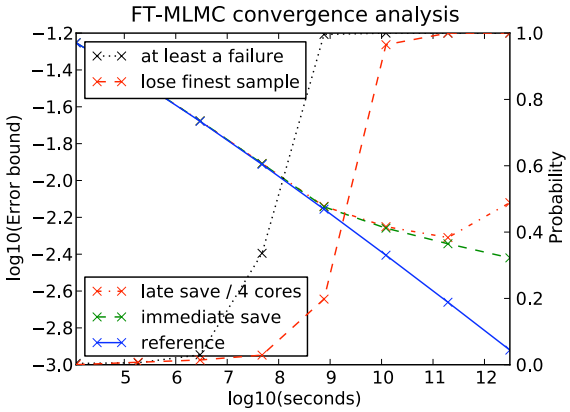
The standard MLMC implementation works satisfactory as long as it fails only rarely. In Fig. 4 less than 10% fail as long as the computation takes less than 10^7sec. Computations with longer run-times benefit from a FT-MLMC method. Close to a run-time of 10^9sec. the FT-MLMC method experiences many sample losses, that the performance drops significantly. This is apparent by the increasing disparity between the two FT-MLMC graphs "late save/4 tasks" and "immediate save" with the fault-free MLMC "reference" graph. Hence, the FT-MLMC method allows to solve problems which use two orders of magnitude more run-time compare to the larges problem solved with a standard MLMC implementation. The number of cores and the run-time per core of the first and the last computation done with the FT-MLMC method is shown in Table 3.



(a) 1D-FT-MLMC-FVM with parameters from Table 2.



(b) 3D-FT-MLMC-FVM with parameters from Table 2.



(c) 3D-FT-MLMC-FVM with parameters from Table 2.

Figure 4: Behavior of the FT-MLMC-FVM with different parameters under Weibull distributed failures.

	first		last
(a)	$2.6 \cdot 10^3$ cores at $1.4 \cdot 10^4$ s (3.9 h)	–	$2.0 \cdot 10^4$ cores at $1.0 \cdot 10^5$ s (31 h)
(b)	$1.5 \cdot 10^3$ cores at $4.4 \cdot 10^4$ s (12 h)	–	$1.2 \cdot 10^4$ cores at $8.7 \cdot 10^4$ s (24 h)
(c)	$1.7 \cdot 10^3$ cores at $2.8 \cdot 10^4$ s (7 h)	–	$1.4 \cdot 10^4$ cores at $4.6 \cdot 10^4$ s (15 h)

Table 3: The FT-MLMC method in Fig. 4 is useful for computations between 10^7sec. and 10^9sec.. The number of cores and the run-time per core is shown for the first and the last computation done with FT-MLMC.

In Fig. 4 it is further observed that the performance of the FT-MLMC method drops significantly as soon as the probability of losing the finest samples approaches values close to one. This is expected, since then much of the computational work is used by the attempt to compute samples, of which in the end only very few to none survive. This leads to a high error to work ratio, or in other words to a poor FT-MLMC performance.

In practice one should start using the FT-MLMC method whenever a standard MLMC implementation with MPI crashes too frequently. There is no need to know the exact failure statistics

(e.g. Weibull) to get a rough approximation of when this will happen. This behavior may as well be experienced, estimated according to expert knowledge, or estimated with simpler models such as mean time between failures (MTBF). The same holds for the approximation of the "lose finest sample" probability, an indicator for the drop in the FT-MLMC performance.

6. Conclusions and future work

We introduced and analyzed a checkpoint-free and fault-tolerant Multi Level Monte Carlo strategy, termed FT-MLMC. The approach is based on disregarding all samples affected by a node failure at run-time. It is assumed that MPI is extended such that processes unaffected by a failure can continue working and communicating. The MLMC estimate is computed with the remaining samples. By incorporating general statistical models of sample losses into the mathematical failure model and into the error bound of the FT-MLMC method a new MLMC error bound conditional on prescribed node-failure statistics is derived.

The principal conclusion of the present analysis is that *up to a certain rate of node failures per sample the FT-MLMC error bound and, hence, the performance of the FT-MLMC, is provably of the same type as that of the standard, fault-free MLMC*. We therefore conclude that *in this range of failure rates*, there is no need for additional fault tolerance strategies such as, e.g., checkpoint/restart or re-computation of lost samples.

Due to the increasing likelihood of node-failures at run-time in emerging massive parallel hardware, obviation of checkpointing may afford substantial increases in parallel efficiency and scalability. In our analysis, we paid particular attention to the case when failures are Weibull distributed. We considered, exemplarily, the case that the samples were computed with the Finite Volume Method (FVM). We emphasize, however, that neither the Weibull distribution nor the FVM are essential for the principal conclusions about the performance of the FT-MLMC method.

We showed by a number of examples that the FT-MLMC method compared to the standard MLMC method can compute much bigger and much more (approx. 100×) time-consuming problems, compared to the standard MLMC method.

The proposed approach of simply discarding samples affected by a node failure at run-time has the additional benefit that failures do not extend the run-time as for instance checkpoint/restart, or the re-computation of the lost samples does. This leads to a fixed run-time, even if failures occur.

The above analysis completely disregards silent-errors. In a further work we may cover some silent-errors by detecting them by statistical means [10].

Further work also includes investigations of the resilience of parallel random number generators (such as WELL [9]). This includes the preservation of statistical quality with partial loss of streams.

[1] *ALSVID-UQ*. avalable from http://mlmc.origo.ethz.ch/, Aug. 2012.

[2] A. Barth, Ch. Schwab, and N. Zollinger. Multi-level Monte Carlo finite element method for elliptic pdes with stochastic coefficients. *Numer. Math.*, 119(1):123–161, 2011.

[3] F. Cappello. Fault tolerance in petascale/exascale systems: Current knowledge, challenges and research opportunities. *Int. J. High Perform. Comput. Appl.*, 23(3):212–226, 2009.

[4] F. Cappello, A. Geist, B. Gropp, L. Kale, B. Kramer, and M. Snir. Toward exascale resilience. *Int. J. High Perform. Comput. Appl.*, 23(4):374–388, 2009.

[5] G. Fagg and J. Dongarra. FT-MPI: Fault tolerant MPI, supporting dynamic applications in a dynamic world. In J. Dongarra, P. Kacsuk, and N. Podhorszki, editors, *Recent Advances in Parallel Virtual Machine and Message Passing Interface (EuroPVM/MPI 2000)*, pages 346–353, Berlin, 2000. Springer. (Lecture Notes in Computer Science, 1908).

[6] G. S. Fishman. *Monte Carlo: Concepts, Algorithms, and Applications*. Springer, New York, 5th edition, 2003.
[7] M. B. Giles. Multilevel Monte Carlo path simulation. *Operations Research*, 56(3):607–617, 2008.
[8] M. Hoemman and M. Heroux. Fault-tolerant iterative methods via selective reliability. Technical Report SAND2011-3915 C, Sandia National Laboratories, Albuquerque NM, 2011.
[9] P. L'Ecuyer and F. Panneton. Fast random number generators based on linear recurrences modulo 2: overview and comparison. In M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joine, editors, *Proceedings of the 2005 Winter Simulation Conference*, pages 110–119, 2005.
[10] Y. Li and M. Mascagni. Analysis of large-scale grid-based Monte Carlo applications. *Int. J. High Perform. Comput. Appl.*, 17:369–382, 2003.
[11] M. Mascagni, D. Ceperley, and A. Srinivasan. SPRNG: A scalable library for pseudorandom number generation. *ACM Trans. Math. Softw.*, 26:436–461, 2000.
[12] S. Mishra and Ch. Schwab. Sparse tensor multi-level Monte Carlo finite volume methods for hyperbolic conservation laws with random initial data. *Math. Comp.*, 81(280):1979–2018, 2012.
[13] S. Mishra, Ch. Schwab, and J. Šukys. Multi-level Monte Carlo finite volume methods for nonlinear systems of conservation laws in multi-dimensions. *J. Comput. Phys.*, 231(8):3365–3388, 2012.
[14] S. Mishra, Ch. Schwab, and J. Šukys. Multi-level Monte Carlo finite volume methods for shallow water equations with uncertain topography in multi-dimensions. *SIAM Journal of Scientific Computing, (submitted)*, 2011.
[15] D. Nurmi, J. Brevik, and R. Wolski. Modeling machine availability in enterprise and wide-area distributed computing environments. In J. Cunha and P. Medeiros, editors, *Euro-Par 2005. Parallel Processing*, volume 3648 of *Lecture Notes in Computer Science*, pages 432–441. Springer, Berlin, 2005. doi:10.1007/11549468_50.
[16] A. Papoulis and S.U. Pillai. *Probability, random variables, and stochastic processes*. McGraw-Hill electrical and electronic engineering series. McGraw-Hill, 2002.
[17] W. P. Petersen and P. Arbenz. *Introduction to Parallel Computing*. Oxford University Press, Oxford, 2004.
[18] H. Rinne. *The Weibull Distribution: A Handbook*. CRC Press, 2009.
[19] J. K. Salmon, M. A. Moraes, R. O. Dror, and D. E. Shaw. Parallel random numbers: as easy as 1, 2, 3. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC'11, pages 16:1–16:12, New York, NY, USA, 2011. ACM.
[20] B. Schroeder and G. A. Gibson. A large-scale study of failures in high-performance computing systems. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 249–258, Washington, DC, USA, 2006. IEEE Computer Society.
[21] Ch. Schwab. Numerical analysis of stochastic partial differential equations. Lecture notes, 2010.
[22] Ch. Schwab and C. J. Gittelson. Sparse tensor discretizations of high-dimensional parametric and stochastic PDEs. *Acta Numer.*, 20:291–467, 2011.
[23] J. Šukys, S. Mishra, and Ch. Schwab. Static load balancing for multi-level Monte Carlo finite volume solvers. In R. Wyrzykowski et al., editors, *Parallel Processing and Applied Mathematics (PPAM 2011)*, pages 245–254, Berlin, 2012. Springer. (Lecture Notes in Computer Science, 7203).
[24] Y. Zhao. *Parametric inference from window censored renewal process data*. PhD thesis, The Ohio State University, Columbus, Ohio, 2006.
[25] The raw data and more information is available at the two URLs http://www.pdl.cmu.edu/FailureData/ and http://www.lanl.gov/projects/computerscience/data/.

Research Reports

No.	Authors/Title
12-24	*St. Pauli, P. Arbenz and Ch. Schwab* Intrinsic fault tolerance of multi level Monte Carlo methods
12-23	*V.H. Hoang, Ch. Schwab and A.M. Stuart* Sparse MCMC gpc Finite Element Methods for Bayesian Inverse Problems
12-22	*A. Chkifa, A. Cohen and Ch. Schwab* High-dimensional adaptive sparse polynomial interpolation and applications to parametric PDEs
12-21	*V. Nistor and Ch. Schwab* High order Galerkin approximations for parametric second order elliptic partial differential equations
12-20	*X. Claeys, R. Hiptmair, and C. Jerez-Hanckes* Multi-trace boundary integral equations
12-19	*Šukys, Ch. Schwab and S. Mishra* Multi-level Monte Carlo finite difference and finite volume methods for stochastic linear hyperbolic systems
12-18	*Ch. Schwab* QMC Galerkin discretization of parametric operator equations
12-17	*N.H. Risebro, Ch. Schwab and F. Weber* Multilevel Monte-Carlo front tracking for random scalar conservation laws
12-16	*R. Andreev and Ch. Tobler* Multilevel preconditioning and low rank tensor iteration for space-time simultaneous discretizations of parabolic PDEs
12-15	*V. Gradinaru and G.A. Hagedorn* A timesplitting for the semiclassical Schrödinger equation
12-14	*Ph. Grohs and G. Kutyniok* Parabolic molecules
12-13	*V. Kazeev, O. Reichmann and Ch. Schwab* Low-rank tensor structure of linear diffusion operators in the TT and QTT formats
12-12	*F. Müller, P. Jenny and D.W. Meyer* Multilevel Monte Carlo for two phase flow and transport in random heterogeneous porous media