**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

**Revised: November 2013**

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 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 vs. work of an 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.

*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 [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 the only serious problem is to guarantee the statistical independence of the random input draws (e.g. [19]).

*Corresponding author
*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 by the 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 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 methodes 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 required finest discretization level is reached. In MLMC methods the expected difference from two consecutive discretization levels is computed using the MC method. Hence, in this paper the difference of a realization computed on two consecutive discretization levels is referred to as a MLMC sample of a certain "level". A "level" does, therefore, in this paper, not denote a discretization level but a level related to a MLMC sample. It is by now known (see, eg., [2, 7, 12–14]) that under rather general assumptions, MLMC methods converge faster than MC, in terms of overall computational work, i.e., cumulated execution time. The cumulated execution time and memory consumption of the computation of samples depends on the levels and differ considerably: the computation of an MLMC sample of a 'finer' level may require much more compute resources (execution time, memory space, number of cores) than sample of a 'coarser' level. The load of an MLMC simulation is therefore not as easy to balance [25] 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 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 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 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 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 in the present paper, 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 Eliminiation. 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 accomodates 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 (wrt. work), it is crucial to choose properly the numbers $M_\ell$ of samples on level $\ell$.

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 $M_\ell$ 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 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. Based on this paper the proposed fault tolerant MLMC was already implemented [18] using the User Level Failure Mitigation (ULFM) [3], a fault tolerant version of MPI. We compare the 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.

## 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 $\Omega$, $\sigma$-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 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, 24])

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

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, 21] incorporate this strategy.

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

$$E_{\hat{M}}[X] := \frac{1}{\hat{M}} \sum_{i=1}^{\hat{M}} X^i,$$

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., [24] 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*

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

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

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

Theorem I.3.17. in [23] 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). □

Once a particular statistical distribution for the node failures has been adopted (and calibrated to the hardware platform of interest), the term $\mathbb{E}'[\hat{M}^{-1/2}]$ can be computed and hence using Theorem 2.1, the a priori error bound for the fault tolerant MC (FT-MC) method as well. The intuition behind Theorem 2.1 is that all failures are accounted for in the error bound of the FT-MC. The number of failures occurring in a FT-MC simulation varies according to a known failure distribution, nevertheless the a priori error bound in Theorem 2.1 remains valid, regardless of the number of failures. Knowing the failure distribution allows to compute an error bound before the number of failures and the number of surviving samples is known.

The condition $\hat{M}(\omega') = 0$ means that no samples remain. If the probability of this event is positive, then obviously $\mathbb{E}'[\hat{M}^{-1/2}]$ tends to infinity and the FT-MC error bound becomes meaningless. *We therefore assume that* $\mathbb{P}'(\{\hat{M} = 0\}) = 0$ *in the ensuing analysis.* 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 the derivation of Giles [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 converge with order $\alpha$

$$\|\mathbb{E}[X - X_{h_\ell}]\|_B = O(h_\ell^\alpha), \quad \alpha > 0, \tag{5}$$

where $\|\cdot\|_B$ denotes a norm in the Banach space $B$. We also assume a convergence order $\beta$ for

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

and assume that

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

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 $X_{h_\ell}$ satisfies

$$W_\ell = O(h_\ell^{-\gamma}) \,, \quad \gamma > 0 \,. \tag{8}$$

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 $\ell$ we use an ordinary MC method with $M_\ell$ samples $(X^i_{h_\ell} - X^i_{h_{\ell-1}})$, $i = 1, \ldots, M_\ell$, to approximate $\mathbb{E}[X_{h_\ell} - X_{h_{\ell-1}}]$. Note, that in this paper a sample on level $\ell$ comprises a difference between the same realization $X^i$ computed on two consecutive discretization levels $h_\ell$ and $h_{\ell-1}$. In order to form this difference the realizations $X^i_{h_\ell}$ and $X^i_{h_{\ell-1}}$ have to be computed with the same $\omega$. On the coarsest level we estimate $\mathbb{E}[X_{h_0}]$ with $M_0$ samples $X^i_{h_0}$, $i = 1, \ldots, M_0$,. 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),

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

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 $M_\ell$. 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 $M_\ell$, $0 \leq \ell \leq L$, 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 $\ell$ of mesh-refinement on which all MC samples are lost will be referred to as a *lost level*. In MLMC $\hat{M}_\ell = 0$ on some level $\ell$ 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 $M_\ell \geq 0$, and hence $M_\ell = 0$ might appear. We shall refer to this case as an "entirely lost level". In *the discussion of this section, the number of samples $M_\ell$ are not random.*

The MLMC estimate (10) is modified such that lost levels ($M_\ell = 0$) are not taken into account,

$$E[X_{h_L}] = E_{M_0}[X_{h_0}] + \sum_{\ell=1}^{L} E_{M_\ell}[X_{h_\ell} - X_{h_{\ell-1}}], \quad E_{M_\ell}[\cdot] = 0 \text{ if } \hat{M}_\ell = 0. \tag{13}$$

The error bound

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

is analyzed.

In the following derivation we use a bound for a MC simulation with $M = 0$ when nothing is computed ($E_M[X] = 0$, $M = 0$):

$$\|\mathbb{E}[X] - E_0[X]\|_{L^2(\Omega;B)} = \|\mathbb{E}[X]\|_{L^2(\Omega;B)} = \sqrt{\|\mathbb{E}[X]\|_B^2} \leq \sqrt{\mathbb{E}[\|X\|_B^2]} = \|X\|_{L^2(\Omega;B)} \tag{14}$$

Following the derivation (11) provides a error bound for the MLMC method where levels might be lost ($M_\ell = 0$)

$$\begin{aligned}
\|\mathbb{E}[X] - E[X_{h_L}]\|_{L^2(\Omega;B)} &\leq \|\mathbb{E}[X] - \mathbb{E}[X_{h_L}]\|_B + \|\mathbb{E}[X_{h_L}] - E[X_{h_L}]\|_{L^2(\Omega;B)} \\
&\leq \|\mathbb{E}[X] - \mathbb{E}[X_{h_L}]\|_B + \|\mathbb{E}[X_{h_0}] - E_{M_0}[X_{h_0}]\|_{L^2(\Omega;B)} \\
&\quad + \sum_{\ell=1}^{L} \|\mathbb{E}[X_{h_\ell} - X_{h_{\ell-1}}] - E_{M_\ell}[X_{h_\ell} - X_{h_{\ell-1}}]\|_{L^2(\Omega;B)} \\
&\overset{(2),(14)}{\leq} \|\mathbb{E}[X] - \mathbb{E}[X_{h_L}]\|_B + \min(1, M_0^{-1/2})\|X_{h_0}\|_{L^2(\Omega;B)} \\
&\quad + \sum_{\ell=1}^{L} \min(1, M_\ell^{-1/2})\|X_{h_\ell} - X_{h_{\ell-1}}\|_{L^2(\Omega;B)} \,.
\end{aligned} \tag{15}$$

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

**Theorem 3.1.** *The Multilevel Monte Carlo error with lost levels (i.e. when $M_\ell \geq 0$) is bounded by*

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

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

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

Losing a level ($M_\ell = 0$) 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 $\ell$ 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. Fault Tolerant MLMC error bound with random failures*

As in the fault tolerant MC method 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 $M_\ell$ the number of samples per level is not a fixed number anymore, but a random number $\hat{M}_\ell$.

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 level $\hat{M}_\ell$ depends on the runtime failures. Hence $\hat{M}_\ell$ is a measurable mapping

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

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:

$$\hat{E}[X_{h_L}] = E_{\hat{M}_0}[X_{h_0}] + \sum_{\ell=1}^{L} E_{\hat{M}_\ell}[X_{h_\ell} - X_{h_{\ell-1}}] \,, \quad E_{\hat{M}_\ell}[\cdot] = 0 \text{ if } \hat{M}_\ell = 0.$$

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

$$\begin{aligned}\|\mathbb{E}[X] - \hat{E}[X_{h_L}]\|_{L^1(\Omega';L^2(\Omega;B))} \leq \|\mathbb{E}[X] - \mathbb{E}[X_{h_L}]\|_B + \mathbb{E}'[\min(1, \hat{M}_0^{-1/2})]\|X_{h_0}\|_{L^2(\Omega;B)} \\ + \sum_{\ell=1}^{L} \mathbb{E}'[\min(1, \hat{M}_\ell^{-1/2})]\|X_{h_\ell} - X_{h_{\ell-1}}\|_{L^2(\Omega;B)},\end{aligned}$$

*or given the assumptions* (5), (6) *and* (7)

$$\|\mathbb{E}[X] - \hat{E}[X_{h_L}]\|_{L^1(\Omega';L^2(\Omega;B))} \leq O(h_L^\alpha) + \mathbb{E}'[\min(1, \hat{M}_0^{-1/2})]O(1) + \sum_{\ell=1}^{L} \mathbb{E}'[\min(1, \hat{M}_\ell^{-1/2})]O(h_{\ell-1}^\beta) \,.$$

*Proof.* By Theorem 3.1 we have

$$\begin{aligned}\|\mathbb{E}[X] - \hat{E}[X_{h_L}]\|_{L^2(\Omega;B)} \leq \|\mathbb{E}[X] - \mathbb{E}[X_{h_L}]\|_B + \min(1, \hat{M}_0^{-1/2})\|X_{h_0}\|_{L^2(\Omega;B)} \\ + \sum_{\ell=1}^{L} \min(1, \hat{M}_\ell^{-1/2})\|X_{h_\ell} - X_{h_{\ell-1}}\|_{L^2(\Omega;B)},\end{aligned}$$

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

$$\begin{aligned}&\int_{\Omega'} \|\mathbb{E}[X] - \hat{E}[X_{h_L}]\|_{L^2(\Omega;B)}\mathbb{P}'(d\omega') \\ &\qquad \leq \int_{\Omega'} \|\mathbb{E}[X] - \mathbb{E}[X_{h_L}]\|_B + \min(1, \hat{M}_0^{-1/2})\|X_{h_0}\|_{L^2(\Omega;B)} \\ &\qquad\qquad + \sum_{\ell=1}^{L} \min(1, \hat{M}_\ell^{-1/2})\|X_{h_\ell} - X_{h_{\ell-1}}\|_{L^2(\Omega;B)}\mathbb{P}'(d\omega') \,.\end{aligned}$$

With the linearity of the mathematical expectation the error estimate becomes

$$\begin{aligned}&\int_{\Omega'} \|\mathbb{E}[X] - \hat{E}[X_{h_L}]\|_{L^2(\Omega;B)}\mathbb{P}'(d\omega') \\ &\qquad \leq \int_{\Omega'} \|\mathbb{E}[X] - \mathbb{E}[X_{h_L}]\|_B\mathbb{P}'(d\omega') + \int_{\Omega'} \min(1, \hat{M}_0^{-1/2})\mathbb{P}'(d\omega')\|X_{h_0}\|_{L^2(\Omega;B)} \\ &\qquad\qquad + \sum_{\ell=1}^{L} \int_{\Omega'} \min(1, \hat{M}_\ell^{-1/2})\mathbb{P}'(d\omega')\|X_{h_\ell} - X_{h_{\ell-1}}\|_{L^2(\Omega;B)} \,.\end{aligned}$$

□

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 $E_{\hat{M}_\ell}[X_{h_\ell} - X_{h_{\ell-1}}] = \sum_{i=1}^{\hat{M}_\ell}(X^i_{h_\ell} - X^i_{h_{\ell-1}})$, where $X^i_{h_\ell}$ and $X^i_{h_{\ell-1}}$ approximate the same realization $X^i$ with two different discretization parameters $h_\ell$ 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 $\hat{M}_\ell$. 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 [5]. 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. At present only rather rough failure models are available and the development of more detailed, and realistic models is needed, in our opinion. For new HPC systems first empirical statistical failure studies [16, 22] are available. Schroeder and Gibson [22] derived a somewhat 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 [27] 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 node, 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. [17])

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

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. This model assumes that node failures are statistically independent. This is one of the rather rough approximations made. The data of Schroeder and Gibson [22] also gives evidence of a correlation between the failure rate and the type and intensity of the workload running on a machine, as the failure rate is considerably lower during the night and the weekend. However, we had to idealize, since more accurate node failure data and models are hard to come by, at least at presence. Nevertheless we point out that our theory accomodates also other, more sophisticated parametric failure models, as long as they satisfy the general assumptions in the present paper, and we hope that failure models with better quantitative accuracy will become available in the near future.

### *4.1. Time to next failure in a renewal process*

When a new simulation on a computer is started, generally we do not know the last time of failure of this computer. Thus, when applying the failure model from Section 4, 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 the model applied these RVs are the time intervals between two failures, hence they are Weibull distributed. After a failure the failed component is assumed to be 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 [20] states that, as long as $t$ is finite, $F_t$ can only be evaluated numerically. In the case $t \to \infty$ however [20, eq. (4.41a)] provides the analytic distribution function of $F_\infty$.

A procedure to draw realizations of the forward recurrence time $F_\infty$ is given in [26]:

1. Generate the random variable S from $S = \Gamma(1 + \frac{1}{k}, \lambda^k)$, where $\lambda$ 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 $F_\infty$ from $F_\infty = \text{Uniform}(0, V)$.

### *4.2. Estimate the number of samples per level with MC*

One instance of the number of samples $\hat{M}_\ell$ on level $\ell$ is drawn by generating a realization of the first failure $F_\infty$ on a node where samples of level $\ell$ are computed. The term $\mathbb{E}'[\min(1, \hat{M}_\ell^{-1/2})]$ is then estimated by MC, i.e., by averaging over different realizations of $\hat{M}_\ell$.

### *4.3. Calibration of parameters*

The two Weibull scale and shape parameters $\lambda$ and $k$ are used in the above drawing of the forward recurrence times $F_\infty$. They depend on the machinery the FT-MLMC method runs on. In [20] several methods are presented to estimate both Weibull parameters.

In our analysis we use the parameters given in [22], $k \approx 0.7$ and $\lambda \approx 7.6 \cdot 10^5$ for a single node. Note that $k \approx 0.7$ is explicitly given in [22] for an individual node. $\lambda$ is estimated from [22, 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, $\lambda$ can be approximated by $0.5 \approx 1 - e^{-(4.5 \cdot 10^5/\lambda)^k}$, yielding $\lambda \approx 7.6 \cdot 10^5$.

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 [22] 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 satisfying 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 ALSVID-UQ [1] for solving hyperbolic systems of conservation laws. We set $\alpha = \beta = s$ in eqs. (5)–(6). With this choice, the work to compute a sample $X^i_{h_\ell} - X^i_{h_{\ell-1}}$ on level $\ell$ is

$$W_\ell = 2^{d+1} W_{\ell-1} = 2^{(d+1)\ell} W_0, \qquad \ell \geq 0, \tag{16}$$

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

$$\mathrm{mem}_\ell = 2^d \mathrm{mem}_{\ell-1} = 2^{d\ell} \mathrm{mem}_0, \qquad \ell \geq 0. \tag{17}$$

As suggested in [12] for $s < (d+1)/2$, the number of samples $M_\ell$ on level $\ell$, is set to be

$$M_\ell = 2^{-2s} M_{\ell-1} = 2^{2(L-\ell)s} M_L, \qquad \ell \geq 0. \tag{18}$$

On the finest level, we chose a moderate number of samples, e.g., $M_L = 8$.

We parallelize our simulation level-wise and across levels. A core deals with samples of only one level, i.e, it computes solutions on two resolutions used for a sample $X^i_{h_\ell} - X^i_{h_{\ell-1}}$ on level $\ell$.

In our implementation we choose an intermediate level $b$ on which individual samples are executed on a single core. This level is somewhat arbitrary. It is determined typically based on memory requirements or execution times of a sample. The subdomains are chosen large enough such that the communication over subdomain-interfaces plays a miner role. 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 (17). On levels $\ell \leq b$ samples are executed sequentially. So, the number $N_\ell$ of cores invested for a sample on level $\ell$ is

$$N_\ell = \lceil 2^{d(\ell-b)} \rceil, \qquad \ell \geq 0. \tag{19}$$

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 coarser 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. Any single node may compute 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. On level $b = 1$ the number of cores in a task is one. This number increases by 8 towards the finer levels, according to (19). 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 that of the others, causing a small load imbalance.

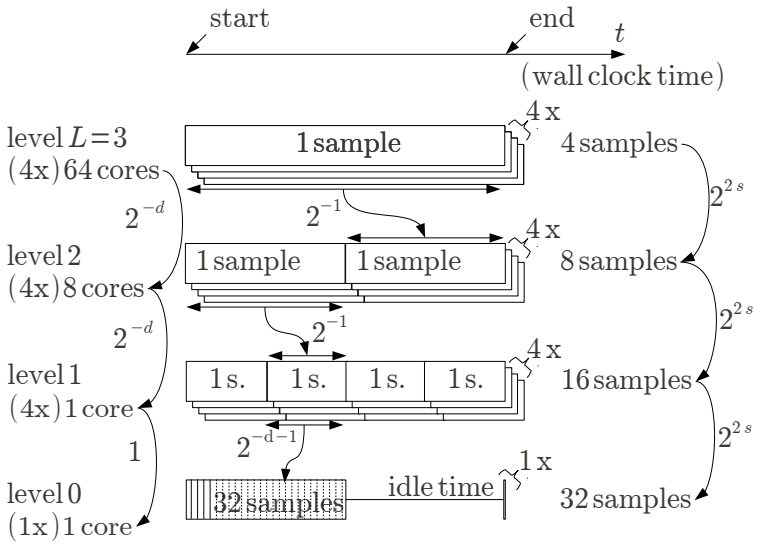



Figure 1: Computing an MLMC estimate with parameters $L = 3$, $M_L = 4$, $d = 3$, $s = 1/2$ and only one core per task on levels 0 and 1.

As mentioned in Section 3.4 it is crucial that samples $(X^i_{h_\ell} - X^i_{h_{\ell-1}})$ are lost as a whole, i.e., it must never happen that one part (either $X^i_{h_\ell}$ or $X^i_{h_{\ell-1}}$) survives and is used in the end result while the other part is lost. This is achieved by computing a whole sample $(X^i_{h_\ell} - X^i_{h_{\ell-1}})$ on cores belonging to the same node. In case of node failure the whole sample is lost. Large samples however use many cores such that multiple nodes have to be used to compute them. In case of node failure, the processes running on unaffected cores have to realize that they should stop computing their part of the sample. How this is solved is implementation dependent. In our FT-MLMC implementation [18] we used User Level Failure Mitigation (ULFM) [3], a fault tolerant extension of MPI. With ULFM a MPI process trying to communicate with a failed one is notified about the failure. This process can then prohibit any further communication on the communicator used for the computation of the affected sample. This insures that the information of failures is received by all involved MPI processes. Further details are available in [18].

### *5.1. Lowering risk of level loss*

A high probability of losing all samples on one level increases the FT-MLMC error bound in Theorem 3.2 substantially. This applies particularly to 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 all samples on coarse levels:

- "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 runtime 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 losing all samples on 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.


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.

- "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. Euler equation of gas dynamics (New a section on its own)*

We show results for the finite volume method (FT-MLMC-FVM) that solves the Euler equations of gas dynamics. Following [13, 25], the $d$-dimensional 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 $D = (0,1)^d$, with outflow boundary conditions, where $\rho$ is the density and $\mathbf{u}$ the velocity vector. The pressure $p$ and the total energy $E$ are related by the equation of state of an ideal gas, ie.

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

with $\gamma$ 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 $t = .6$ s. In [13] an MLMC Finite Volume Method (MLMC-FVM) approach is proposed to estimate $\mathbb{E}[\mathbf{X}(\cdot,t)]$. A MLMC-FVM error bound is shown for scalar solutions in [12] and assumed to hold for nonlinear hyperbolic systems of conservation laws in [13, 14] by

$$\|\mathbb{E}[\mathbf{X}(\cdot,t)] - E^{\hat{\mathcal{L}}}[\mathbf{X}(\cdot,t)]\|_{L^2(\Omega;L^1(\mathbb{R}^d))} \leq C_1 h_L^s + C_2 \hat{M}_0^{*-\frac{1}{2}} + C_3 \left\{ \sum_{\ell=1}^{L} \hat{M}_\ell^{*-\frac{1}{2}} h_{\ell-1}^s \right\}, \tag{20}$$

which corresponds to the error bound (12). We set the constants as $10 \cdot C_1 = C_2 = 10 \cdot C_3 = .5$. The convergence rate

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

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

$$\|\mathbf{X}_\ell(\cdot,t) - \mathbf{X}_{\ell-1}(\cdot,t)\|_{L^2(\Omega;L^1(\mathbb{R}^d))} \leq C_4 h_{\ell-1}^s. \tag{21}$$

This bound is related to assumption (6). The bounds (20) and (21) imply the validity of Theorems 3.1 and 3.2 for the FT-MLMC-FVM method.

*5.3. Assessment of the FT-MLMC error bound*

The error bound is compared with the measured error of a FT-MLMC implementation. In [18] we report on the implementation of a MPI-parallelized FT-MLMC method in ALSVID-UQ [1]. In this implementation we used the User Level Failure Mitigation (ULFM) [3], a fault tolerant extension of MPI. A failure generator is used to simulate MPI process failures. For this purpose a timed asynchronous interrupt is set, which kills the MPI process using the exit system call. This assures that failures can happen at any time, during the computation, the communication, or while ULFM is recovering from previous failures. Our implementation in [18] demonstrates that the FT-MLMC algorithm proposed in this paper can be successfully implemented.

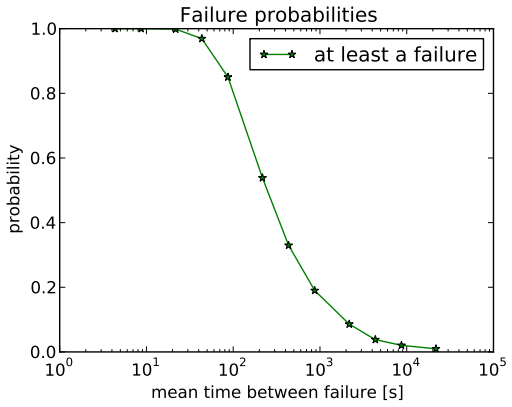
In [18] we computed the FT-MLMC results on Brutus, a large compute cluster at ETH Zurich, where we used one node with four 12-core AMD Opteron 6174 CPUs and 64 GB of RAM. Due to incompatibility of the ULFM developer implementation with the batch system of Brutus it is currently not possible to run the code on multiple nodes. We used a modified version of the failure model in Section 4 where failures do not affect a whole node but only a process. Otherwise the failure model remains the same.

The exact parameters used for the simulation are described in [18] and are summarized in Table 1. All measurements are averages over multiple FT-MLMC runs. In the current paper we used 100 runs for $6\,\mathrm{s} < \mathrm{MTBF} < 100\,\mathrm{s}$, otherwise 30 runs where used. The error is computed using a MLMC reference solution $E[X_{\mathrm{ref}}]$ with $L = 8$, $M_L = 8$ and $h_L = 2^{-11}$, the absolute FT-MLMC error is measured as $\sqrt{E_{30;\,100}[\|E[X_{h_L}] - E[X_{\mathrm{ref}}]\|_{L^1}^2]}$. In contrast to the "immediate save" strategy described in Section 5.1 we implemented an "intermediate save" strategy, where the intermediate results of a process are sent to other processes computing samples of the same level, up to four times during the computation. Additionally in both the "intermediate save" and the "late save" strategy the samples of a level are split into at least two tasks, each of which is computed on independent cores.

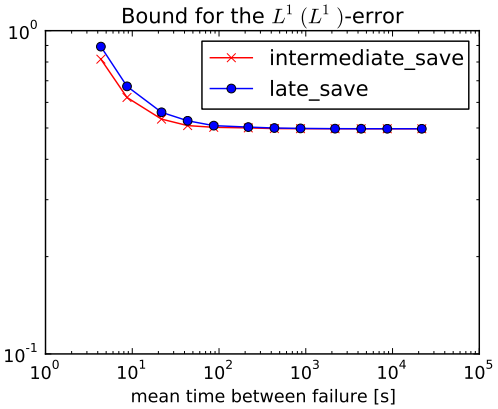
| | |
|---|---|
| $M_L$ | 2 |
| $d$; $s$ | 2; 1/2 |
| # cells on level 0 ($\propto 1/h_0$) | $2^4$ |
| intermediate save | 4 times, using 2 tasks |
| late save | using 2 tasks |
| $W_5$ (time for one sample) | $16 \times 94\,\mathrm{s}{=}1504\,\mathrm{s}$ |
| simulations with levels | $L = 5$ |
| one core per task | $b = 3$ |
| Weibull parameters $\lambda$; $k$ | variable; 0.5 |

Table 1: The parameters used in Fig. 3.

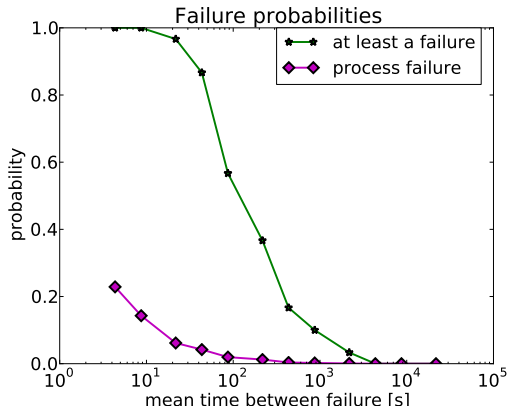
In Fig. 3 we compare the measurements from a FT-MLMC implementation with the In Fig. 3 we compare the measurements from a FT-MLMC implementation with the derived FT-MLMC error bound. The same FT-MLMC problem was simulated with different mean time between failures (MTBF), by varying the Weibull scale parameter $\lambda$ of the failure model. In order to save computing time rather large failure rates (small MTBFs) where used. We determine the *entry failure rate*, the failure rate from which on fault tolerance is useful. This point is reached when at least 10% of all FT-MLMC runs encounter a failure, and hence 10% of all non fault tolerant implementations (using standard MPI-3.0) would terminate without a result. This is measured by the "at least a



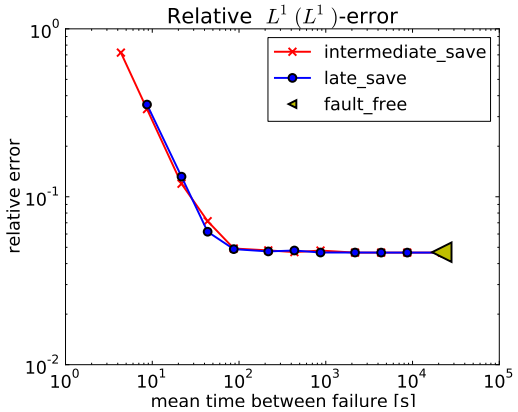

(a) Failure probabilities of the 2D FT-MLMC bound.




(b) 2D FT-MLMC error bound.




(c) Measured failure probabilities of the 2D FT-MLMC implementation [18].




(d) Measured relative errors of the 2D FT-MLMC implementation [18].

Figure 3: Behavior of the 2D FT-MLMC-FVM with different parameters under Weibull distributed failures.

failure" probability, shown in Fig. 3(a) for the theory derived in this paper and in Fig. 3(c) for the measured FT-MLMC implementation [18]. In both subfigures the entry failure rate is around MTBF=$10^3$ s for the given problem. Fig. 3(c) additionally presents the measured "process failure" probability, which shows that in average 20% of all started processes fail at MTBF$\approx$ 6 s. However, at the entry failure rate (MTBF=$10^3$ s) the process failure probability is still small.

In Fig. 3(b) and 3(d) the FT-MLMC error bound and the measured relative FT-MLMC error respectively are shown for the "intermediate save" and the "late save" strategies. Additionally we present the measured reference error from the fault free ALSVID-UQ [1] in Fig. 3(d). We measure the *critical failure rate*, the failure rate from which on the FT-MLMC method does not perform well anymore. In the FT-MLMC implementation, shown in Fig. 3(d), the critical failure rate is at MTBF $\approx$ 40 s. In the FT-MLMC error bound, shown in Fig. 3(b), the critical failure rate is similar. There the "intermediate save" strategy provides a small benefit over the "late save" strategy, whereas in the measurement this benefit is negligible. At MTBF $\approx$ 40 s the "intermediate save" strategy performs even slightly better compared to the "late save" strategy. This artifact appears since we are only averaging over 100 FT-MLMC runs.

It is observed in both the FT-MLMC implementation as well as in the FT-MLMC error bound that the error only increases slightly between the entry failure rate and the critical failure rate. In our opinion, this is a fundamental insight, as one could also expect a "gradual" loss of performance, rather than a range of failure intensities where only a minor degradation is to be observed. Only after the critical failure rate is reached, the error strongly increases and the numerical quality of the simulation results is lost.

The rate of failure explosion after the critical failure rate does not match well when comparing the error bound and the measured relative error. One reason is that the constants $C_1$, $C_2$, and $C_3$ in the error bound are unknown or can only be estimated very conservatively. In reality many additional constants are implicitly contained in the three constants $C_1$, $C_2$, and $C_3$. As in $\mathbb{E}'[\min(1, \hat{M}_0^{-1/2})]$ with certain implicitly problem dependent constants $C'$ and $C''$. It would be more precise to instead write, with $\mathbf{1}_A$ denoting the indicator function of the set $A$

$$\mathbb{E}'[C'\mathbf{1}_{M_\ell=0} + C''\hat{M}_0^{-1/2}\mathbf{1}_{M_\ell>0})] \,.$$

This can be seen in the derivation of the bound with lost levels, in eq. (15), where the part $\hat{M}_0^{-1/2}$ comes from the two inequalities (2) and (14) which imply two unknown constants $C' \geq 1$ and $C'' \geq 1$.

This constant $C'$ describes the error in case a level is lost while $C''$ describes the error when samples are computed. Levels are mostly lost for small MTBFs. Hence choosing a large constant $C'$ would mainly influence the error bound for small MTBFs, hence leading to a steeper rate of failure explosion after the critical failure rate for the error bound. Several other implied parameters could be used to tune the error bound towards the measured error. However, without tuning the parameters we get a error bound which shows good qualitative agreement with the measured error.

Also keep in mind, that we are comparing two different things, a possibly pessimistic asymptotic error bound with an actual discretiation, resp. sampling error.

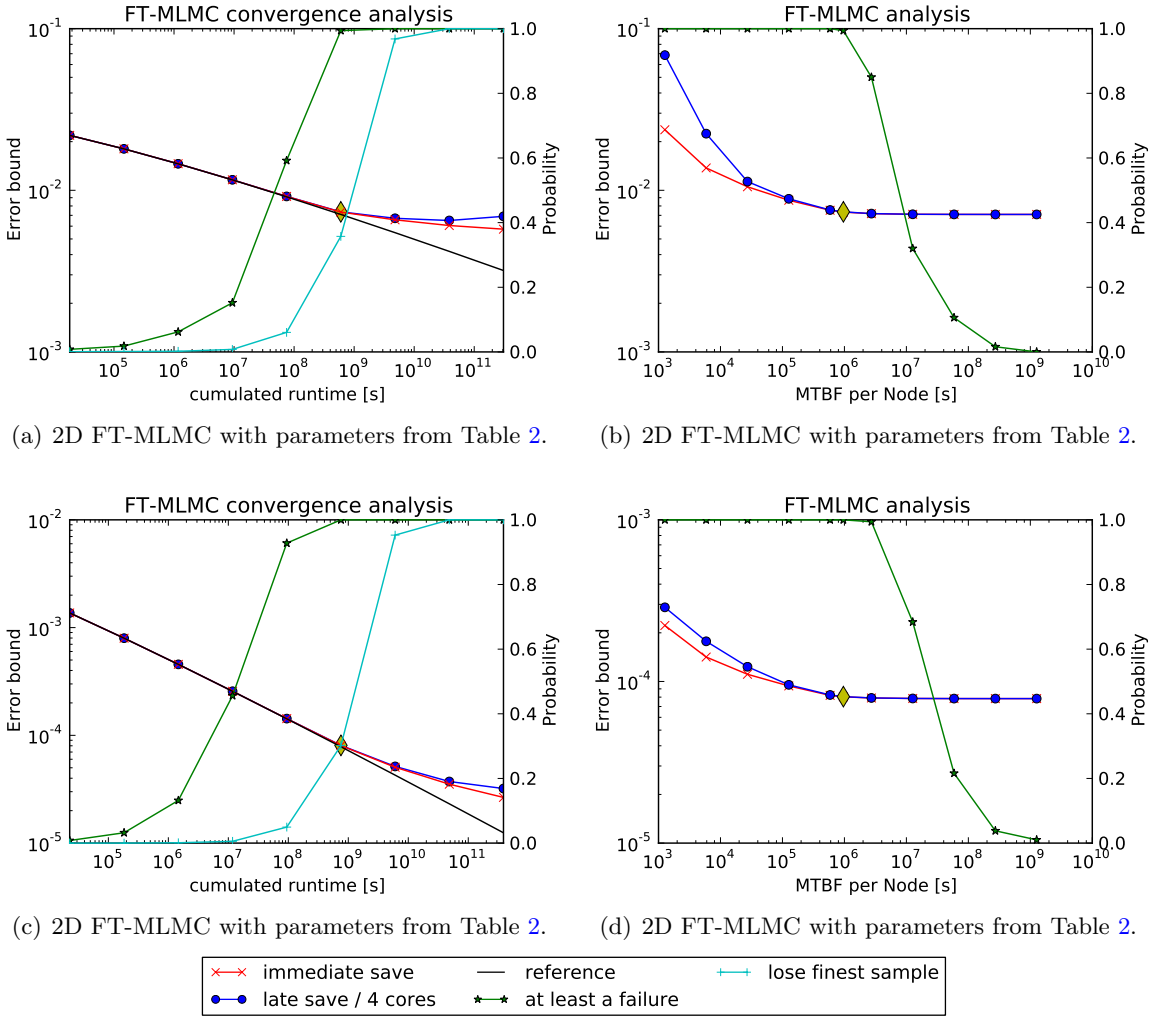
### *5.4. Analysis of FT-MLMC error bounds*

Four error bounds with different parameters are analyzed, see Table 2. The error bounds from the 2 dimensional problem are shown in Fig. 4 and the three dimensional once in Fig. 5. Two plots are shown for each case. In the left the problem solved is scaled, but the failure rate (Weibull parameter $\lambda$) remains the same and vice versa in the right one. The run is exactly the same in both plots, and its results are in either plot marked with a yellow rhombus.

| | Fig. 4(a,b) | Fig. 4(c,d) | Fig. 5(a,b) | Fig. 5(c,d) |
|---|---|---|---|---|
| $M_L$ | 8 | 8 | 8 | 8 |
| $d$; $s$ | 2; 1/2 | 2; 1 | 3; 1/2 | 3; 1 |
| # cells on level 0 ($\propto 1/h_0$) | $32 \times 32$ | $16 \times 16$ | $8 \times 8 \times 8$ | $4 \times 4 \times 4$ |
| $W_0$ (time for one sample) | 0.05 s | 0.04 s | 0.01 s | 0.007 s |
| simulations with levels | $L = 5, \ldots, 13$ | $L = 5, \ldots, 13$ | $L = 4, \ldots, 11$ | $L = 4, \ldots, 11$ |
| one core per task | $b = 4$ | $b = 3$ | $b = 4$ | $b = 3$ |
| Weibull parameters $\lambda$; $k$ | $7.6 \cdot 10^5$; 0.7 | $7.6 \cdot 10^5$; 0.7 | $7.6 \cdot 10^5$; 0.7 | $7.6 \cdot 10^5$; 0.7 |

Table 2: The parameters used in Fig. 4 and 5 (a)–(c).

We determine the entry failure rate and the critical failure rate in all the measurements of Figs. 4(b,d) and 5(b,d). In all these cases the entry failure rate is at around $\lambda = 5 \cdot 10^7$ and the




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

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

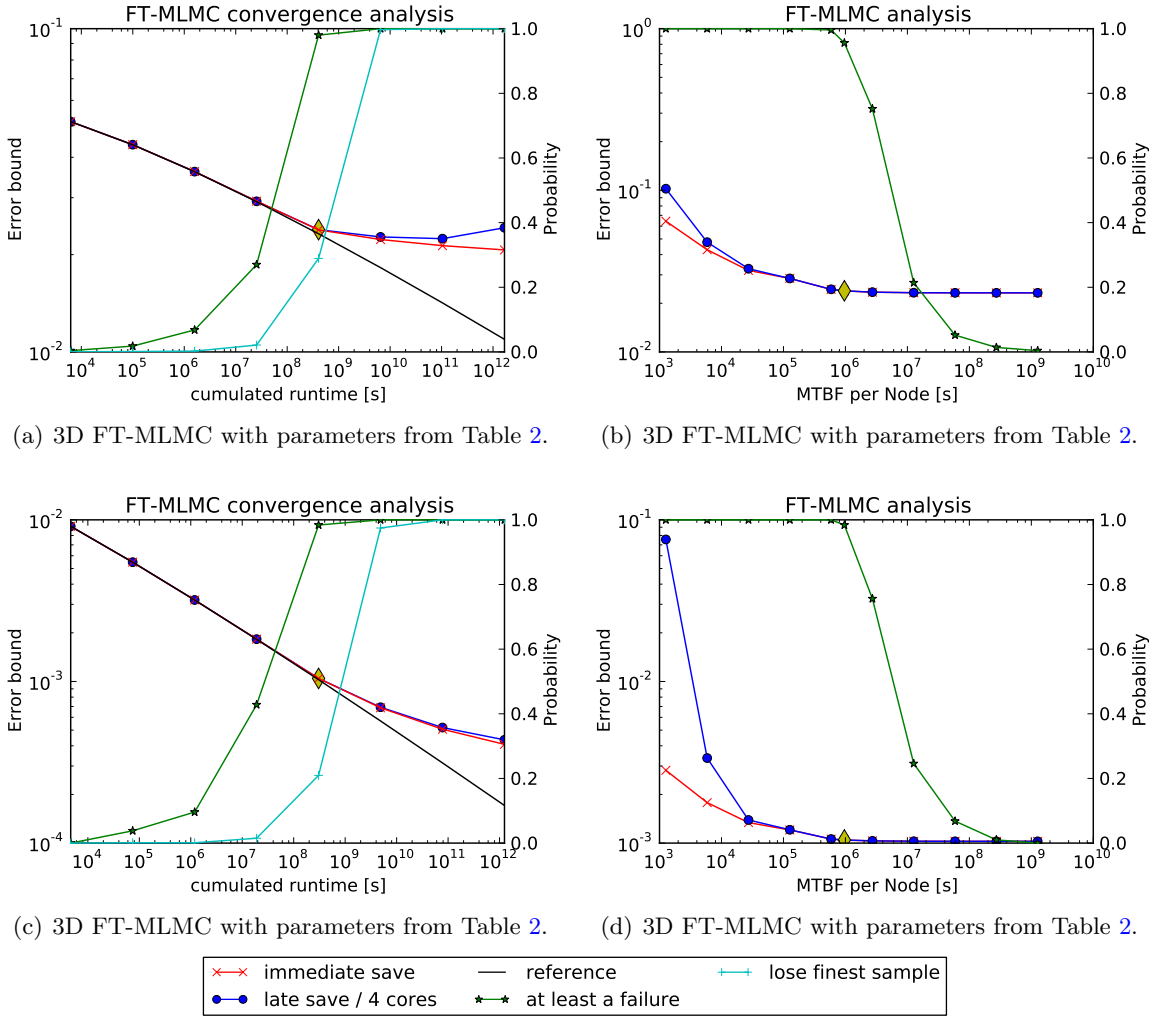
(c) 2D FT-MLMC with parameters from Table 2.

(d) 2D FT-MLMC with parameters from Table 2.

Figure 4: Behavior of the 2D FT-MLMC with different parameters under Weibull distributed failures.

critical failure rate at around $\lambda = 10^5$. Again we observe that between the entry failure rate and the critical failure rate the error bound only increases slightly. Hence, this desirable property is found also in large runs, and in multiple dimensional runs. Only after the critical failure rate the error explodes.

Similar to the entry failure rate, the failure rate from which on fault tolerance is useful, there is a *entry problem size*, the problem size from which on fault tolerance is useful, when operating on a given computer. Again we define this point as reached when 10% of all FT-MLMC runs encounter a failure. The entry problem size is at a total execution time of around $5 \cdot 10^6$ s for Figs. 4(a) and 5(a,c). In Fig. 4(c) the entry problem size is around $5 \cdot 10^5$ s. Similar to the critical failure rate, we use the term of *critical problem size* in Figs. 4(a,c) and 5(a,c). This problem size is reached in our measurements at around $10^9$ s. Also in these figures we observe that the convergence behavior of FT-MLMC is only negligibly affected by failures between the *entry problem size* and the critical problem size. After the critical problem size however, the FT-MLMC method experiences many




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

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

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

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

Figure 5: Behavior of the 3D FT-MLMC with different parameters under Weibull distributed failures.

sample losses, such 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, when using the failure distribution of Section 4 the FT-MLMC method allows to solve problems which use more than two orders of magnitude more runtime compared to the largest problem solved with a standard MLMC implementation.

The "late save/4 tasks" error does not only stop decreasing but grows again after the critical problem size, Fig. 4(a,c) and 5(a). 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.

In Fig. 4(a,c) and 5(a,c) it is further observed that the performance of the FT-MLMC method drops significantly as soon as the probability of losing all samples on the finest level approaches

values close to one. This can be 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.

*5.5. Practical applicability*

To employ the theoretical error bounds derived in this paper, it is essential to have an accurate, parametric statistical node failure model of the compute platform used. Once such a model has been verified and calibrated to the particular hardware of interest, the computational work could be redistributed among nodes and cores in order to minimize the overall impact of node failures on the accuracy of the computation. Finally, based on a statistical node failure model of the type considered here, a FT-MLMC error bound can be computed, which allows to infer precise statements on the contributions to the overall error stemming from failures, and on whether the problem of interest can be solved using FT-MLMC or plain MLMC on the given system.

The weak point of the presently proposed approach is that *accurate and detailed, validated statistical failure models and the data to calibrate them*

are currently not available. Hopefully, the situation will improve over time, as the awareness of faults grows. For the time being we opted to use failure models that allow to determine rough estimates of the entry failure rate, critical failure rate, entry problem size, and critical problem size.

We expect that in practice the entry problem size and the entry failure rate are not computed with a known failure distribution, but rather we expect that users will be bothered when failures terminate the application prematurely in at least 10% of the runs, which coincides with the mentioned quantities. Only then fault tolerance appears to become an issue for the user (this figure might, however, be problem and application dependent). It is however not practical for the user to decide when the performance of FT-MLMC deteriorates, i.e., the critical problem size and the critical failure rate cannot be determined *without some knowledge* of the failure distribution. In our measurements we observed that there is a range between the entry failure rate and the critical failure rate, and a range between the entry problem size and the critical problem size where the performance of FT-MLMC appears to be unaffected by faults. In our problems, obviating checkpointing in this observed fault tolerance extends the range of applicability by two orders of magnitude: on a given computer the FT-MLMC method is applicable to problems that take around 100 times longer compared to the "plain" fault-free MLMC method, or, equivalently, a given problem can be solved on a computer with around 100 times smaller MTBF thanks to fault tolerance.

We are aware that this estimated range may be specific to our assumptions, and may change when other failure distributions and problems are considered. We are however confident that for a number of applications, the potential range of improvement on Monte Carlo methods is large.

Our failure model does not take in to account that random node failures may be correlated among different nodes in practice. This has adverse effects on the error bound. For large samples, running on multiple nodes, correlated node failures could be beneficial in the following sense: as soon as one node of a sample failed, the computation of this sample is stopped anyway, hence it does not matter if additional nodes fail simultaneously. For most problems which occur in engineering practice, and in particular for the Finite Volume simulations considered in the present paper, MC Samples on low levels easily fit on a single node. In the "late save/4 tasks" strategy at least 4 cores on different nodes are used to compute samples of this level. Node failure correlation might increase the risk of losing all 4 involved nodes simultaneously, which has a clearly negative effect

on the error bound. But we do not see any parameter which would ruin the nice property that the error only increases slightly between the entry failure rate and the critical failure rate, and between the entry problem size and the critical problem size and that there is a considerable large range where FT-MLMC is applicable.

The measurement of the number of failures in one single FT-MLMC run on a computer with unknown failure distribution does not provide any statistical information: *it is not possible to estimate the range of applicability of the FT-MLMC method with a single run.* The same holds for attempts to estimate the constants in the a priori error estimate from data obtained with one single run.

## 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*, in MC and MLMC PDE simulations 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 compared the derived FT-MLMC error bound with measured errors of a fault tolerant implementation [18]. We showed by a number of examples that the FT-MLMC method compared to the standard MLMC method can solve, on a given hardware platform, considerably larger and more time-consuming problems as compared to the standard fault-free 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 does for instance checkpoint/restart, or the re-computation of lost samples. This leads to a bounded run-time, even if failures occur.

We conclude with the need for further research to develop better and more realistic stochastic and/or deterministic failure models for massively parallel (in particular, exascale) computing platforms. It would be very useful to know the failure distribution of the computer one is using. In the best case a failure model would be provided by hardware vendors, ideally with failure intensity parameters. Another line of research is the development of effective self-calibrating statistical failure models which, when run on a given computing platform, would "probe" a given hardware for failures and accumulate estimates of failure parameters at runtime.

The presently proposed models and their analysis completely disregard silent-errors. In a further work we may cover some silent-errors by detecting them by statistical means [10].

We further emphasize that the presently proposed failure models are uncorrelated across the hardware platform: in the presently proposed models, failures occur independently of relative position of the node within the compute platform. There is, however, evidence that the 'ageing' of hardware as well as the time of day and the day of the week can affect fault arrival intensity parameters, and it is plausible that spatial correlation (e.g. due to overheating) can play a role as well.

Further work also includes investigations of the resilience of parallel random number generators. This includes the preservation of statistical quality with partial loss of streams. Once realistic and, ideally, self-calibrating fault models are in place, *adaptive processor load distribution at runtime* to minimize impacts of faults on numerical simulation quality would be a major milestone.

## Acknowledgment

The authors thank Jonas Šukys for the support with ALSVID-UQ, and the anonymous reviewers for their valuable comments and suggestions that improved the quality of the paper.

## References

[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] W. Bland, A. Bouteiller, T. Herault, J. Hursey, G. Bosilca, and J. Dongarra. An evaluation of user-level failure mitigation support in MPI. In J. L. Träff, S. Benkner, and J. Dongarra, editors, *Recent Advances in the Message Passing Interface*, pages 193–203. Springer, 2012. (Lecture Notes in Computer Science, 7490).

[4] 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.

[5] 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.

[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 J. Sci. Comput., 34(6), pp. B761–B784*, 2012.

[15] MPI Forum. MPI: A Message-Passing Interface Standard. Version 3.0, October 2013. available at: http://www.mpi-forum.org (Oct. 2013).

[16] 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.

[17] A. Papoulis and S.U. Pillai. *Probability, random variables, and stochastic processes*. McGraw-Hill electrical and electronic engineering series. McGraw-Hill, 2002.

[18] S. Pauli, M. Kohler, and P. Arbenz. A fault tolerant implementation of Multi-Level Monte Carlo methods. *Proceedings of the international conference on parallel computing (ParCo 2013), Munich, Germany*, 2013.

[19] W. P. Petersen and P. Arbenz. *Introduction to Parallel Computing*. Oxford University Press, Oxford, 2004.

[20] H. Rinne. *The Weibull Distribution: A Handbook*. CRC Press, 2009.

[21] 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.

[22] 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.

[23] Ch. Schwab. Numerical analysis of stochastic partial differential equations. Lecture notes, 2010.

[24] Ch. Schwab and C. J. Gittelson. Sparse tensor discretizations of high-dimensional parametric and stochastic PDEs. *Acta Numer.*, 20:291–467, 2011.

[25] 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).

[26] Y. Zhao. *Parametric inference from window censored renewal process data*. PhD thesis, The Ohio State University, Columbus, Ohio, 2006.

[27] 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*<br>Intrinsic fault tolerance of multi level Monte Carlo methods |
| 12-23 | *V.H. Hoang, Ch. Schwab and A.M. Stuart*<br>Sparse MCMC gpc Finite Element Methods for Bayesian Inverse Problems |
| 12-22 | *A. Chkifa, A. Cohen and Ch. Schwab*<br>High-dimensional adaptive sparse polynomial interpolation and applications to parametric PDEs |
| 12-21 | *V. Nistor and Ch. Schwab*<br>High order Galerkin approximations for parametric second order elliptic partial differential equations |
| 12-20 | *X. Claeys, R. Hiptmair, and C. Jerez-Hanckes*<br>Multi-trace boundary integral equations |
| 12-19 | *Šukys, Ch. Schwab and S. Mishra*<br>Multi-level Monte Carlo finite difference and finite volume methods for stochastic linear hyperbolic systems |
| 12-18 | *Ch. Schwab*<br>QMC Galerkin discretization of parametric operator equations |
| 12-17 | *N.H. Risebro, Ch. Schwab and F. Weber*<br>Multilevel Monte-Carlo front tracking for random scalar conservation laws |
| 12-16 | *R. Andreev and Ch. Tobler*<br>Multilevel preconditioning and low rank tensor iteration for space-time simultaneous discretizations of parabolic PDEs |
| 12-15 | *V. Gradinaru and G.A. Hagedorn*<br>A timesplitting for the semiclassical Schrödinger equation |
| 12-14 | *Ph. Grohs and G. Kutyniok*<br>Parabolic molecules |
| 12-13 | *V. Kazeev, O. Reichmann and Ch. Schwab*<br>Low-rank tensor structure of linear diffusion operators in the TT and QTT formats |
| 12-12 | *F. Müller, P. Jenny and D.W. Meyer*<br>Multilevel Monte Carlo for two phase flow and transport in random heterogeneous porous media |