ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Seminar for Applied Mathematics SAM

Adaptive load balancing for massively parallel multi-level Monte Carlo solvers

J. Sukys

Research Report No. 2013-21
July 2013

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

Funding: This work is performed under ETH interdisciplinary research grant CH1-03 10-1 and CSCS production project grant ID S366.

Adaptive load balancing for massively parallel multi-level Monte Carlo solvers

Jonas Šukys

ETH Zürich, Switzerland,
jonas.sukys@sam.math.ethz.ch.

Abstract. The Multi-Level Monte Carlo (MLMC) algorithm was shown to be a robust and fast solver for uncertainty quantification in the solutions of multi-dimensional systems of stochastic conservation laws. A novel *static* load balancing procedure is already developed to ensure scalability of the MLMC algorithm on massively parallel hardware up to 40 000 cores. However, for random fluxes or random initial data with large variances, the time step of the explicit time stepping scheme becomes also random due to the *random* CFL stability restriction. Such *sample path dependent* complexity of the underlying deterministic solver renders the aforementioned *static* load balancing very inefficient. We introduce an improved, *adaptive* load balancing procedure which is based on two key ingredients: 1) pre-computation of the time step size for *each* realization, 2) distribution of the obtained loads using the greedy algorithm to workers (core groups) with non-identical speed of execution. Numerical experiments in multi-dimensions showing strong scaling of our implementation are presented.

Keywords: uncertainty quantification, conservation laws, multi-level Monte Carlo, FVM, load balancing, greedy algorithms, linear scaling.

Acknowledgments. This work is performed under ETH interdisciplinary research grant CH1-03 10-1 and CSCS production project grant ID S366.

1 Introduction

A number of problems in physics and engineering are modeled in terms of systems of conservation laws:

$$\begin{cases} \mathbf{U}_t(\mathbf{x},t) + \operatorname{div}(\mathbf{F}(\mathbf{U})) = \mathbf{S}(\mathbf{x},\mathbf{U}), \\ \mathbf{U}(\mathbf{x},0) = \mathbf{U}_0(\mathbf{x}), \end{cases} \quad \forall (\mathbf{x},t) \in \mathbb{R}^d \times \mathbb{R}_+. \tag{1}$$

Here, $\mathbf{U} : \mathbb{R}^d \to \mathbb{R}^m$ denotes the vector of conserved variables, $\mathbf{F} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^{m\times d}$ is the collection of directional flux vectors and $\mathbf{S} : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}^m$ is the source term. The partial differential equation is augmented with initial data $\mathbf{U}_0$.

Examples for conservation laws include the shallow water equations of oceanography, the Euler equations of gas dynamics, the Magnetohydrodynamics (MHD) equations of plasma physics, the wave equation and others.

As the equations are non-linear, analytic solution formulas are only available in very special situations. Consequently, numerical schemes such as finite volume methods [4] are required for the study of systems of conservation laws.

Existing numerical methods for approximating (1) require initial data $\mathbf{U}_0$, source $\mathbf{S}$ and flux function $\mathbf{F}$ as input. However, in most practical situations, it is not possible to measure these inputs precisely. Such uncertainty in inputs propagates to the solution, leading to the *stochastic* system of conservation laws:

$$\begin{cases} \mathbf{U}(\mathbf{x},t,\omega)_t + \operatorname{div}(\mathbf{F}(\mathbf{U},\omega)) = \mathbf{S}(\mathbf{x},\omega), \\ \qquad\qquad\qquad \mathbf{U}(\mathbf{x},0,\omega) = \mathbf{U}_0(\mathbf{x},\omega), \end{cases} \quad \mathbf{x}\in\mathbb{R}^d, \quad t>0, \quad \forall \omega \in \Omega. \tag{2}$$

where $(\Omega, \mathcal{F}, \mathbb{P})$ is a complete probability space, the initial data $\mathbf{U}_0$ and the source term $\mathbf{S}$ are random fields [5,6], and the flux $\mathbf{F}$ is a Ω-uniformly Lipschitz random function [5]. The solution is also realized as a random field; its statistical moments (e.g. expectation $\mathbb{E}[\mathbf{U}]$ and variance $\mathbb{V}[\mathbf{U}]$) are the quantities of interest. An estimate of the expectation can be obtained by the Monte Carlo finite volume method (MC-FVM) [5], i.e. by computing the sample mean (ensemble average) of M solutions $\mathbf{U}_{\mathcal{T}}^{i,n}$, each of them approximated using FVM method [4] on a mesh $\mathcal{T}$ with mesh width Δx:

$$E_M[\mathbf{U}_{\mathcal{T}}^n] := \frac{1}{M}\sum_{i=1}^{M} \mathbf{U}_{\mathcal{T}}^{i,n}, \quad M\in\mathbb{N}. \tag{3}$$

MC-FVM estimate $E_M[\mathbf{U}_{\mathcal{T}}^n]$ with $M = \mathcal{O}(\Delta x^{-s/2})$ was proven [6,5,7] to converge to the $\mathbb{E}[\mathbf{U}]$, where s denotes the convergence rate of the FVM solver. The error of MC-FVM was shown to scale asymptotically as $(\text{Work})^{-s/(d+1+2s)}$, making MC-FVM method computationally not feasible when high accuracy is needed.

The *multi-level* Monte Carlo finite volume method (MLMC-FVM) was recently proposed in [6,5]. The key idea behind MLMC-FVM is to simultaneously draw MC samples on a hierarchy of nested grids. There are four main steps:

1. **Nested meshes:** Consider *nested* triangulations $\{\mathcal{T}_\ell\}_{\ell=0}^{\infty}$ of the spatial domain with corresponding mesh widths Δx_ℓ that satisfy $\Delta x_\ell = \mathcal{O}(2^{-\ell}\Delta x_0)$, where Δx_0 - mesh width of the coarsest resolution at the lowest level $\ell = 0$. An example of such hierarchy with the first 3 levels is provided in Figure 1.
2. **Sample:** For each level of resolution $\ell \in \mathbb{N}_0$, we draw M_ℓ independent identically distributed (i.i.d) samples $\{\mathbf{U}_{0,\ell}^i, \mathbf{S}_\ell^i, \mathbf{F}_\ell^i\}$ with $i = 1, 2, \ldots, M_\ell$ from the random fields $\mathbf{U}_0, \mathbf{S}, \mathbf{F}$ and approximate $\mathbf{U}_{0,\ell}^i$ and $\mathbf{S}_\ell^i$ by cell averages.
3. **Solve:** For each resolution level ℓ and each realization $\{\mathbf{U}_{0,\ell}^i, \mathbf{S}_{0,\ell}^i, \mathbf{F}_\ell^i\}$, the underlying balance law (1) is solved by the finite volume method [4] with mesh width Δx_ℓ; denote solutions by $\mathbf{U}_{\mathcal{T}_\ell}^{i,n}$ at the time t^n and mesh level ℓ.

4. **Estimate solution statistics:** Fix the highest level $L \in \mathbb{N}_0$. We estimate the expectation of the solution (random field) with the following estimator:

$$E^L[\mathbf{U}(\cdot, t^n)] := \sum_{\ell=0}^{L} E_{M_\ell}[\mathbf{U}^n_{\mathcal{T}_\ell} - \mathbf{U}^n_{\mathcal{T}_{\ell-1}}], \tag{4}$$

with E_{M_ℓ} being the MC estimator defined in (3) for the level ℓ. Higher statistical moments can be approximated analogously (see, e.g., [6,5]).

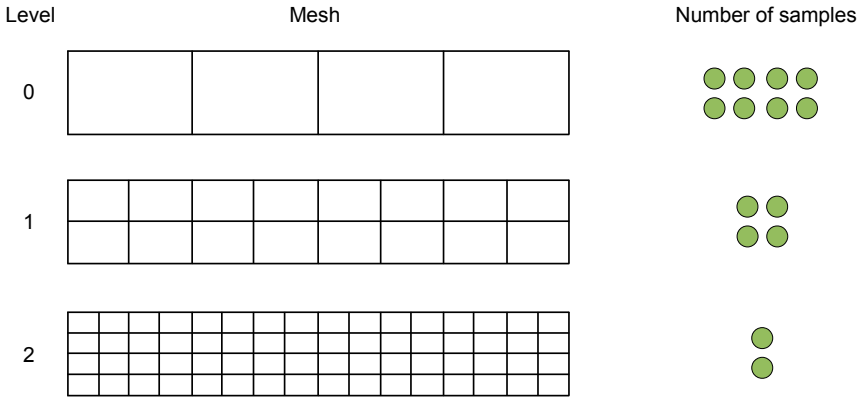

Fig. 1. Example of the first three levels ($L = 2$) of the hierarchy of nested grids for the two dimensional case. Example for the number of samples M_ℓ is provided according to (5) for $s = 1/2$ with the number of samples on the finest mesh level set to $M_L = 2$.

In order to equilibrate statistical and spatio-temporal discretization errors in (4), the following number of samples on each mesh level ℓ is needed [6,7]:

$$M_\ell = M_L 2^{2(L-\ell)s}, \quad M_L \in \mathbb{N}. \tag{5}$$

Notice that most of MC samples are computed on the coarsest mesh level $\ell = 0$, and only a small fixed number M_L of samples is needed on the finest mesh $\ell = L$, see Figure 1. The error vs. work estimate for MLMC-FVM is given by [5,6],

$$\text{error} \lesssim (\text{Work})^{-s/(d+1)} \log(\text{Work}). \tag{6}$$

The above estimate shows that MLMC-FVM is superior to MC-FVM. In particular, at the relative error level of 1%, MLMC-FVM was shown to be approximately two orders of magnitude faster than MC-FVM [6,7,5].

MLMC-FVM is *non-intrusive* as any standard FVM solver can be used in step 3. Furthermore, MLMC-FVM is amenable to *efficient parallelization*, which

is the main topic of this paper. In Sect. 2 *static* load balancing is reviewed and its limits are discussed as a motivation for a novel *adaptive* load balancing, which is introduced in Sect. 4 and whose parallel scaling and efficiency analysis is provided in Sect. 5. Finally, discussion for improvements is given in Sect. 6.

2 Scalable parallel implementation of MLMC-FVM

We use 3 levels of parallelization: across mesh levels, across MC samples and using domain decomposition (DDM) for FVM solver, see example in Figure 2.

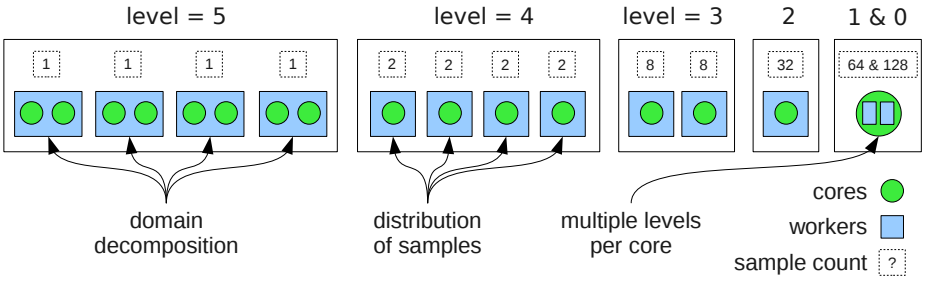

Fig. 2. Parallelization over mesh levels, MC samples and using domain decomposition.

In [8] all required ingredients for parallelization were introduced and analyzed: parallel robust pseudo random number generation (`WELL512a` RNG was used), numerically stable parallel "online" variance computation algorithms, domain decomposition method within each FVM solver and load balancing, which distributes computational work of multiple concurrent solve steps *evenly* among the available cores. All of the aforementioned algorithms, except for the load balancing, are *general* and can be applied to any stochastic system of conservation laws (2).

The goal of this paper is to investigate the limits of the *static* (compile-time) load balancing introduced in [8] and to design a novel *adaptive* (run-time) load balancing which would be efficient for a much broader range of stochastic systems of conservation laws (2), for instance, where the flux function $\mathbf{F}$ is random.

In what follows, we assume *a homogeneous computing environment* meaning that all cores are assumed to have identical CPUs and RAM per node, and equal bandwidth and latency to all other cores.

3 Static load balancing

Static load balancing as presented in [8] distributes samples among all cores at compile-time using the a-priori estimates on the computational work of any

sample on a given mesh resolution $\mathcal{T}$ and time horizon T. Such load balancing appeared to be very efficient for stochastic systems of conservation laws (2) with *deterministic fluxes* $\mathbf{F}$ and stochastic initial data $\mathbf{U}_0$ with *small* variance $\mathbb{V}[\mathbf{U}_0]$, i.e. $\mathbb{V}[\mathbf{U}_0]$ is much smaller than the mean value $\mathbb{E}[\mathbf{U}_0]$ of the field.

In particular, we have verified strong scaling (fixed discretization and sampling parameters while increasing #cores) of our implementation of *static* load balancing in the parallel (using MPI [9]) code ALSVID-UQ [1] up to 40 000 cores at high efficiency, see Figure 3. Labels "MLMC" and "MLMC2" indicate $s = 1/2$ and $s = 1$ in (5), respectively. The runtime of all simulations is measured by the *wall clock time*, accessible by `MPI_Wtime()` routine [9]. We define efficiency by

$$\text{efficiency} := 1 - \frac{(\text{total clock time of all MPI routines and idling})}{(\#\text{cores}) \times (\text{wall clock time})}. \tag{7}$$

Simulations were executed on Cray XE6 (see [10]) with 1496 AMD Interlagos 2 x 16-core 64-bit CPUs (2.1 GHz), 32 GB DDR3 memory per node, 10.4 GB/s Gemini 3D torus interconnect with a theoretical peak performance of 402 TFlops.

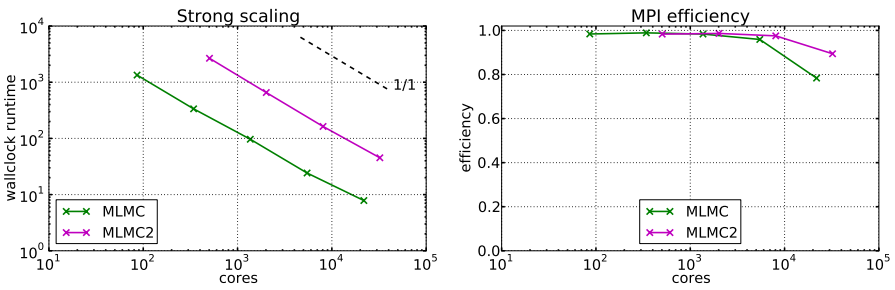

Fig. 3. Strong scaling of *static* load balancing up to 40 000 cores.

3.1 Estimates for the computational work of the FVM solver

Static load balancing as presented in [8] is explicitly based on two (not very strictly defined) properties of the computational work $\text{Work}_{\mathcal{T}}$ required to solve (1) for a given *realization* of random input data on a given mesh $\mathcal{T}$:

1. *accurate* relative (w.r.t. another mesh $\mathcal{T}'$) estimates for $\text{Work}_{\mathcal{T}}$ are available
2. for a *fixed* mesh $\mathcal{T}$, estimate $\text{Work}_{\mathcal{T}}$ is almost the *same* for *all* realizations

Accurate estimates were derived in [8]. For a given mesh $\mathcal{T}$ with mesh width Δx and total number of cells $N = \#\mathcal{T}$, the required computational work for one *time step* (numerical flux approximations) of *one* sample was computed to be

$$\text{Work}_{\mathcal{T}}^{\text{step}} = \text{Work}^{\text{step}}(\Delta x) = \mathcal{O}(N) = K \Delta x^{-d}, \tag{8}$$

where constant K depends on FVM that is used, but does *not* depend on mesh width Δx. In most explicit FVM schemes [4], lower order terms $\mathcal{O}(\Delta x^{-d+1})$ in (8) are negligible, even on a very coarse mesh. To ensure the stability of the FVM scheme, a CFL condition [4] is imposed on the time step size $\Delta t := t^{n+1} - t^n$,

$$\Delta t = \frac{C_{\mathrm{CFL}}}{\lambda}\Delta x, \quad 0 < C_{\mathrm{CFL}} \leq 1, \quad \lambda > 0, \tag{9}$$

where the so-called CFL number C_{CFL} does not depend on Δx and λ is the absolute value of the maximal wave speed [4]. Hence, the computational work $\mathrm{Work}_{\mathcal{T}}^{\mathrm{det}}$ for *one* complete *deterministic* solve using the FVM method on the triangulation $\mathcal{T}$ with mesh width Δx is given by multiplying the work for one step (8) by the total number of time steps Δt^{-1} for the time horizon $T > 0$,

$$\mathrm{Work}_{\mathcal{T}}^{\mathrm{det}} = \mathrm{Work}_{\mathcal{T}}^{\mathrm{step}} \cdot \frac{T}{\Delta t} = K\Delta x^{-d}\lambda\frac{T}{C_{\mathrm{CFL}}\Delta x} = \frac{KT}{C_{\mathrm{CFL}}}\lambda\Delta x^{-(d+1)}. \tag{10}$$

3.2 Limits of the static load balancing

For *deterministic fluxes* $\mathbf{F}$ and stochastic initial data $\mathbf{U}_0$ with *small* variance $\mathbb{V}[\mathbf{U}_0]$, the maximum wave speed λ does *not* vary significantly among all MC samples and hence the second property in subsection 3.1 holds. However, if, for instance, the flux $\mathbf{F}$ is *random*, the maximum wave speed λ can *strongly depend* on the particular realization of $\mathbf{F}$. As an example, we consider wave equation in the random spatially inhomogeneous d-dimensional domain $\mathbf{D} \subset \mathbb{R}^d$, which can be written in a form of a *linear system* of $d+1$ *first order* conservation laws [7],

$$\begin{cases} p_t(\mathbf{x}, t, \omega) - \nabla \cdot (c(\mathbf{x}, \omega)\mathbf{u}(\mathbf{x}, t, \omega)) = 0, \\ \mathbf{u}_t(\mathbf{x}, \omega) - \nabla p(\mathbf{x}, \omega) = 0, \end{cases} \quad \mathbf{x} \in \mathbf{D},\ t > 0,\ \omega \in \Omega, \tag{11}$$

with *deterministic* initial data $p(\mathbf{x}, 0) \in C^\infty(\mathbf{D})$, $\mathbf{u}_0(\mathbf{x}, 0) \in (C^\infty(\mathbf{D}))^d$ and *random* coefficient $c \in L^0(\Omega, L^\infty(\mathbf{D}))$ with $\mathbb{P}[c(\mathbf{x}, \omega) > 0, \forall \mathbf{x} \in \mathbf{D}] = 1$.

As the system (11) is *linear* and random coefficient c is independent of t, the maximum wave speed λ does not depend on t, but *explicitly depends* [7] on c,

$$\lambda(\omega) = \max_{\mathbf{x} \in \mathbf{D}} \sqrt{c(\mathbf{x}, \omega)}. \tag{12}$$

Depending on c, the variance of $\lambda(\omega)$ can be very large. As an example, consider domain $\mathbf{D} = [0, 3]^2$ and the wave speed c given by its Karhunen-Loève expansion,

$$\log c(\mathbf{x}, \omega) = \log \bar{c}(\mathbf{x}) + \sum_{\mathbf{m} \in \mathbb{N}_0^2 \setminus \{0\}}^{\infty} \sqrt{\alpha_{\mathbf{m}}}\Psi_{\mathbf{m}}(\mathbf{x})Y_{\mathbf{m}}(\omega), \tag{13}$$

with eigenvalues $\alpha_{\mathbf{m}}$, eigenfunctions $\Psi_{\mathbf{m}}(\mathbf{x})$, and the mean field $\bar{c}(\mathbf{x})$ set to

$$\alpha_{\mathbf{m}} = |\mathbf{m}_1 + \mathbf{m}_2|^{-2.5}, \quad \Psi_{\mathbf{m}}(\mathbf{x}) = \sin(\mathbf{m}_1\pi\mathbf{x}_2)\sin(\mathbf{m}_2\pi\mathbf{x}_1), \quad \bar{c}(\mathbf{x}) \equiv 0.1,$$

and with *independent standard normal* random variables $Y_m \sim \mathcal{N}[0,1]$.

Then, $c, c^{-1} \notin L^\infty(\Omega, L^\infty(\mathbf{D}))$, i.e. there is *positive* probability such that $\lambda(\omega)$ attains any *arbitrary large* or *arbitrary small* value.

As the system (11) is *linear*, $\lambda(\omega)$ does *not* depend on the initial condition $p_0, \mathbf{u}_0$. However, for the sake of completeness, we provide our choices:

$$p_0 = \exp\left(-12.5 \left\|\mathbf{x} - (1.5, 1.5)^\top\right\|_2^2\right), \quad \mathbf{u}_0 \equiv 0.$$

We would like to note, that according to Proposition 1 and Theorems 2 and 5 in [7], solutions to (11) with (13) are well-defined, and have finite mean and variance, which are well approximated by the MLMC-FVM method.

For such class of problems, the work estimates (10) are no longer valid; more precisely, the computational work required for one sample (realization) on a given mesh $\mathcal{T}$ is a random variable, directly proportional to $\lambda(\omega)$,

$$\mathrm{Work}_{\mathcal{T}}^{\mathrm{rand}}(\omega) = \frac{KT}{C_{\mathrm{CFL}}} \lambda(\omega) \Delta x^{-(d+1)}. \tag{14}$$

Proceeding with our analysis, we consider the *expected* computational work,

$$\mathbb{E}[\mathrm{Work}_{\mathcal{T}}] = \mathbb{E}[\mathrm{Work}_{\mathcal{T}}^{\mathrm{rand}}(\omega)] = \frac{KT}{C_{\mathrm{CFL}}} \mathbb{E}[\lambda(\omega)] \Delta x^{-(d+1)}, \tag{15}$$

which is *finite*, as long as the expected value of maximal wave speed λ is finite. The direct consequence of this is that the *static* load balancing from [8], at least *on average*, is expected to scale. Furthermore, in [7], MLMC-FVM algorithm is analyzed in the case of (14) and the resulting complexity of error vs. *expected* amount of computational work is proven to be analogous to (6),

$$\mathrm{error} \lesssim (\mathbb{E}[\mathrm{Work}])^{-s/(d+1)} \log(\mathbb{E}[\mathrm{Work}]). \tag{16}$$

However, due to non-uniform sizes of samples, the *efficiency* of the balancing is expected to drop significantly for each individual run of the MLMC-FVM algorithm. This can be clearly seen in Figure 4, where the scaling analysis of *static* load balancing in MLMC-FVM for the wave equation (11) with material coefficient given by (13) is performed. As expected, the algorithm *scales* linearly with the number of cores, but the efficiency is consistently low.

The example above is not the only case where λ is random with large relative variance $\mathbb{V}[\lambda]/\mathbb{E}[\lambda]$. All systems (linear and nonlinear) of conservation laws exhibit analogous phenomenon if the flux function $\mathbf{F}$ is *random* with large variance, see [5] for more examples. Another class of problems is with *non-linear* fluxes, where λ *depends* not only on $\mathbf{F}$ but also on $\mathbf{U}$ (hence also on $\mathbf{U}_0$) and potentially has large variance if $\mathbf{U}_0$ has large variance.

To improve the inefficiency of load balancing, samples need to be *redistributed* among cores, taking into account *sample-dependent* estimates (14) for the computational work. This, however, can not be done *statically* at the time of compilation, since $\mathrm{Work}_{\mathcal{T}}^{\mathrm{rand}}(\omega)$ depends on the particular realization. To this end, we introduce an *adaptive* load balancing, where samples are distributed during runtime, i.e. after computing $\lambda(\omega)$ for *each* required realization, but *before* actually starting the FVM time stepping of *any* sample (hence, *not* dynamic balancing).

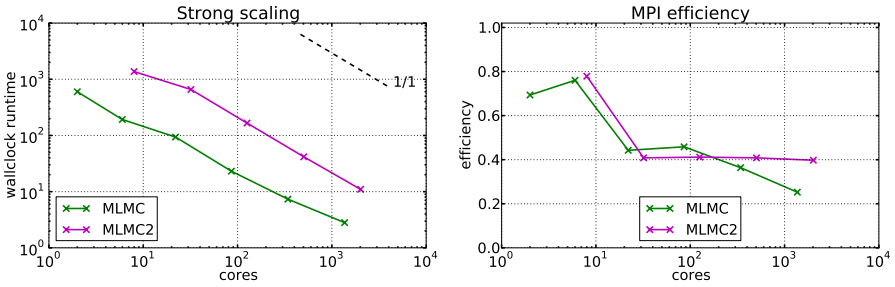

Fig. 4. Inefficient strong scaling of *static* load balancing in case of *random* maximum wave speeds $\lambda(\omega)$ with large relative variance $\mathbb{V}[\lambda]/\mathbb{E}[\lambda]$ resulting from (13).

4 Adaptive load balancing

We assume to have a "pool" $\mathcal{G}$ of cores (processing units), consisting of groups $\mathcal{G}_m$ (of arbitrary size) of cores indexed by "multi level" $m = L, L-1, \ldots, m_0 \geq 0$, which are themselves divided into *equal* groups $\mathcal{G}_m^s$ of cores indexed by "sampler" $s = 1, \ldots, P_m$. The number of cores in a given sampler $\mathcal{G}_m^s$ is independent on s and denoted by D_m. An example of such pool with $L = 5$, $m_0 = 1$, $\{P_m\} = \{1, 1, 2, 4, 8\}$, $\{D_m\} = \{1, 1, 1, 1, 2\}$ is depicted in Figure 2. We assume, that *any* of the MC samples from *any* mesh level ℓ can be efficiently computed on *any* sampler $\mathcal{G}_m^s$ in the pool, in serial or by using domain decomposition if $D_m > 1$. By efficient computation we assume strong scaling of the domain decomposition.

4.1 Computation and distribution of loads

Define Load_ℓ^i to be the normalized (constants are neglected) required computation time for the i-th *difference* of samples between mesh levels ℓ and $\ell - 1$,

$$\text{Load}_\ell^i = \lambda_\ell^i \left(\Delta x_\ell^{-(d+1)} + \Delta x_{\ell-1}^{-(d+1)} \right), \qquad \ell = 0, \ldots, L, \quad i = 1, \ldots, M_\ell, \quad (17)$$

where all λ_ℓ^i need to be computed, possibly in parallel. Since computations of λ_ℓ^i are *much* cheaper compared to the full FVM, they are performed on a *single* largest sampler $\mathcal{G}_L^1$ consisting of D_L cores, and then broadcast to *every* core, i.e. every core has values for all λ_ℓ^i. The resulting amount of data to be communicated is minuscule compared with the amount of data of the needed for the FVM solver.

The goal of the load balancing is to distribute all samples with required computational time Load_ℓ^i to samplers $\mathcal{G}_m^s$. Greedy algorithm for *identical* samplers has been analyzed in [3] and was proven to be a 4/3-approximation, i.e. the makespan (maximum run-time among all workers) is at most 4/3 times larger than the *optimal* (minimal) makespan. If loads are *not* ordered, then greedy algorithm is only a 2-approximation [3]. Here we present a generalization of the greedy algorithm for samplers with *non-identical speed* of execution. The main

idea of the algorithm is the *recursive* assignment of the *largest* available Load_ℓ^i to the sampler $\mathcal{G}_m^s$ for which the total run-time $\mathcal{R}_m^s$ *including* Load_ℓ^i is *minimized*. The pseudo code of the *adaptive* load balancing is provided as Algorithm 1, where the notation $\mathrm{Load}_\ell^i \in \mathcal{G}_m^s$ means that i-th difference of samples between mesh resolution levels ℓ and $\ell-1$ is assigned to be computed on sampler $\mathcal{G}_m^s$.

Algorithm 1 Greedy load balancing (with non-identical speeds of execution)

$\mathcal{L} = \{\mathrm{Load}_\ell^i : \ell = 0, \ldots, L, i = 1, \ldots, M_\ell\}$
while $\mathcal{L} \neq \varnothing$ **do**
  $\mathrm{Load}_\ell^i = \max \mathcal{L}$
  $\mathcal{G}_m^s = \arg\min_{\mathcal{G}_m^s} \left(\mathcal{R}(\mathcal{G}_m^s) + \mathrm{Load}_\ell^i/D_m\right), \quad \mathcal{R}(\mathcal{G}_m^s) = \sum \{\mathrm{Load}/D_m : \mathrm{Load} \in \mathcal{G}_m^s\}$
  $\mathcal{G}_m^s = \mathcal{G}_m^s \cup \mathrm{Load}_\ell^i$
  $\mathcal{L} = \mathcal{L} \backslash \mathrm{Load}_\ell^i$
end while

Note, that if samplers have *identical* speeds of execution, i.e. D_m are all equal, then the above algorithm 1 reduces to the standard greedy algorithm.

If loads are *not* ordered (replace "$\max \mathcal{L}$" by "*any* load from $\mathcal{L}$"), then algorithm 1 is only a $(1 + D_{\max}/D_{\min})$-approximation (analogous proof as in [3]). Hence, if samplers $\mathcal{G}_m^s$ have very *heterogeneous* speeds of execution $1/D_m$, algorithm 1 may provide a *much* longer makespan, compared to the optimal. However, if we assume that loads are *ordered* and are as heterogeneous as samplers,

$$\frac{\mathrm{Load}_{\max}}{\mathrm{Load}_{\min}} := \frac{\max_{\ell,i} \mathrm{Load}_\ell^i}{\min_{\ell,i} \mathrm{Load}_\ell^i} \geq \frac{D_{\max}}{D_{\min}}, \tag{18}$$

then algorithm 1 is a 2-approximation. We present this result as a theorem.

Theorem 1. *If* (18) *holds and the last load of the bottle-neck sampler is bounded by* $(D_{\min}/D_{\max}) \cdot \mathrm{Load}_{\max}$*, then algorithm 1 is a* 2*-approximation.*

Proof. Let $R(\mathcal{G}^*)$ be the run-time of the bottle-neck sampler $\mathcal{G}^*$ and Load^* be the last sample assigned to $\mathcal{G}^*$. Then, according to distribution procedure,

$$R(\mathcal{G}^*) \leq R(\mathcal{G}_m^s) + \mathrm{Load}^*/D_m, \quad \forall m = m_0, \ldots, L, \quad s = 1, \ldots, P_m.$$

Summing the above inequality over all samplers $\mathcal{G}_m^s$, we obtain a bound

$$R(\mathcal{G}^*) - \frac{1}{\#\{\mathcal{G}_m^s\}} \sum_{m,s} \frac{\mathrm{Load}^*}{D_m} \leq \frac{1}{\#\{\mathcal{G}_m^s\}} \sum_{m,s} R(\mathcal{G}_m^s) \leq \mathcal{R}^o,$$

where $\mathcal{R}^o$ is the optimal timespan, which is certainly *not* smaller than the average of all runtimes $R(\mathcal{G}_m^s)$. Next, we use (18) and $\mathrm{Load}^* \leq \mathrm{Load}_{\max} D_{\min}/D_{\max}$,

$$\frac{1}{\#\{\mathcal{G}_m^s\}} \sum_{m,s} \frac{\mathrm{Load}^*}{D_m} \leq \frac{1}{\#\{\mathcal{G}_m^s\}} \sum_{m,s} \frac{D_{\min}}{D_{\max}} \frac{\mathrm{Load}_{\max}}{D_{\min}} \leq \frac{\mathrm{Load}_{\max}}{D_{\max}} \leq \mathcal{R}^o.$$

Combining both bounds, the desired inequality $R(\mathcal{G}^*) \leq 2\mathcal{R}^o$ is obtained. □

In case of MLMC-FVM, the assumption (18) is often satisfied, since loads Load_ℓ^i scale asymptotically as $\text{Work}_{\mathcal{T}_\ell} = \mathcal{O}(2^{(d+1)\ell})$ due to (14), and the speeds of execution D_m using domain decomposition scale only as $\#\mathcal{T}_m$, i.e. $D_m = \mathcal{O}(2^{dm})$.

4.2 Implementation remarks

Once the loads have been distributed to samplers $\mathcal{G}_m^s$, the parallel execution of FVM solves and the final assembly of the MLMC-FVM estimator remained analogous as in [8], i.e. Message Passing Interface (MPI) was chosen, making heavy use of the appropriate local MPI inter-communicators [9]. The *new* part for the *adaptive* balancing is the parallel computation (and broadcast) of the maximum wave speeds λ_ℓ^i, which is problem-specific. For the wave equation (11), λ_ℓ^i were computed by computing random coefficients c_ℓ^i and then using (12).

5 Efficiency and linear scaling in numerical simulations

To ensure a fair comparison, the *adaptive* load balancing algorithm was tested on the same problem as the *static* load balancing, see (11) in subsection 3.2.

In Figure 5 we verify *strong scaling* of our implementation. We observed the scaling to be maintained for up to almost 10 000 cores at high efficiency. Simulations were executed on the same Cray XE6 (see [10]) architecture as in subsection 3.2. We believe that our parallelization algorithm will scale linearly for a much larger number of cores if the problem size is increased.

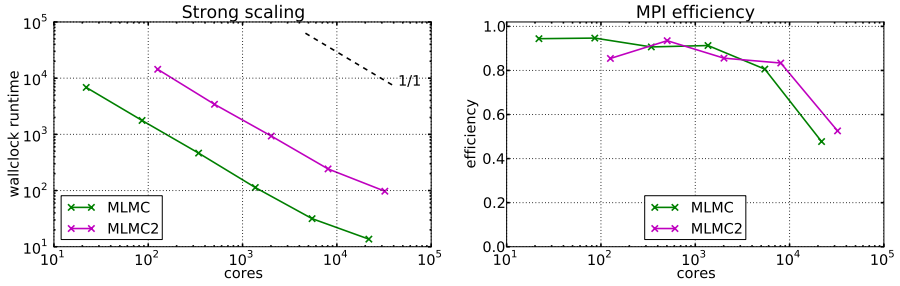

Fig. 5. Strong scaling of *adaptive* load balancing up to 10 00 cores. The efficiency is nearly optimal and is *much* better if compared to the *static* load balancing in Figure 4.

6 Discussion for improvements

Our implementation of the *adaptive* load balancing could, of course, be further improved. For instance, the pre-computation of λ_ℓ^i could be parallelized more efficiently, i.e. the remaining cores could also be incorporated, if needed.

For non-linear fluxes with random initial condition with large variance, $\lambda(\omega)$ also attains large variance, but only the initial data $\mathbf{U}_0(\cdot, \omega)$ can be used to estimate $\lambda(\omega)$. As $\mathbf{U}_0(\cdot, t, \omega)$ evolves to time $t > 0$, $\lambda(\omega)$ changes, hence the work estimates (14) are no longer valid due to assumption of equal time steps, violating the first property in subsection 3.1. In such cases the performances of the *adaptive* load balancing might be sub-optimal. In such cases, *dynamic* load balancing could be used, which might introduce significant additional overhead and deteriorate parallel scaling, see [2] and references therein.

7 Conclusion

MLMC-FVM algorithm is superior to standard MC algorithms for uncertainty quantification in hyperbolic conservation laws, and yet, as most sampling algorithms, it still scales linearly w.r.t. number of uncertainty sources. Due to its non-intrusiveness, MLMC-FVM was efficiently parallelized for multi-core architectures. For systems with *deterministic* fluxes, *static* load balancing was already available [8], which was shown to scale strongly and weakly on the high performance cluster [10] in multiple space dimensions. For *linear* systems with *random* fluxes, *adaptive* load balancing was introduced, which maintains the same scaling properties, but, by design, is applicable to a much wider class of problems.

References

1. ALSVID-UQ, v3.0. Available from `http://www.sam.math.ethz.ch/alsvid-uq`.
2. Sivarama P. Dandamudi. *Sensitivity evaluation of dynamic load sharing in distributed systems.* IEEE Concurrency, **6(3)**:62–72 1998.
3. R. L. Graham. *Bounds on Multiprocessing Timing Anomalies.* SIAM Journal on Applied Mathematics, **17(2)**:416–429, 1969.
4. R.A. LeVeque. *Numerical Solution of Hyperbolic Conservation Laws.* Cambridge Univ. Press 2005.
5. S. Mishra, Ch. Schwab, and J. Šukys. *Multi-level Monte Carlo finite volume methods for uncertainty quantification in nonlinear systems of balance laws.* Von Karman Institute Lecture Notes UQLNCSE6, 2013 (to appear). Available from: `http://www.sam.math.ethz.ch/reports/2012/08`.
6. S. Mishra and Ch. Schwab. *Sparse tensor multi-level Monte Carlo finite volume methods for hyperbolic conservation laws with random initial data.* Math. Comp. **280(81)**:1979–2018, 2012.
7. J. Šukys, Ch. Schwab and S. Mishra. *Multi-Level Monte Carlo Finite Difference and Finite Volume methods for stochastic linear hyperbolic systems.* MCQMC 2012 (in review). Available from: `http://www.sam.math.ethz.ch/reports/2012/19`.
8. J. Šukys, S. Mishra, and Ch. Schwab. *Static load balancing for multi-level Monte Carlo finite volume solvers.* PPAM 2011, Part I, LNCS **7203**:245–254. Springer, Heidelberg (2012).
9. *MPI: A Message-Passing Interface Standard.* Version 2.2, 2009. Available from: `http://www.mpi-forum.org/docs/mpi-2.2/mpi22-report.pdf`.
10. *Rosa,* Swiss National Supercomputing Center (CSCS), Lugano, `www.cscs.ch`.

Recent Research Reports

Nr.	Authors/Title

2013-11 F. Mueller and Ch. Schwab
Finite Elements with mesh refinement for wave equations in polygons

2013-12 R. Kornhuber and Ch. Schwab and M. Wolf
Multi-Level Monte-Carlo Finite Element Methods for stochastic elliptic variational inequalities

2013-13 X. Claeys and R. Hiptmair and E. Spindler
A Second-Kind Galerkin Boundary Element Method for Scattering at Composite Objects

2013-14 I.G. Graham and F.Y. Kuo and J.A. Nichols and R. Scheichl and Ch. Schwab and I.H. Sloan
Quasi-Monte Carlo finite element methods for elliptic PDEs with log-normal random coefficient

2013-15 A. Lang and Ch. Schwab
Isotropic Gaussian random fields on the sphere: regularity, fast simulation, and stochastic partial differential equations

2013-16 P. Grohs and H. Hardering and O. Sander
Optimal A Priori Discretization Error Bounds for Geodesic Finite Elements

2013-17 Cl. Schillings and Ch. Schwab
Sparsity in Bayesian Inversion of Parametric Operator Equations

2013-18 V. Kazeev and Ch. Schwab
Tensor approximation of stationary distributions of chemical reaction networks

2013-19 K. Schmidt and R. Hiptmair
Asymptotic Boundary Element Methods for Thin Conducting Sheets

2013-20 R. Kruse
Consistency and Stability of a Milstein-Galerkin Finite Element Scheme for Semilinear SPDE