Massively parallel Multilevel Monte Carlo methods

Ulrich Rüde
Cerfacs Algo Team
FAU Erlangen-Nürnberg

joint work with B. Gmeiner, D. Drzisga, R. Scheichl, B. Wohlmuth

October 3, 2016
Outline

- Motivation
  - Large Scale Computing and UQ
- Uncertainty quantification: Multilevel Monte Carlo Method
- Parallel multigrid methods
- Scheduling Multilevel Monte Carlo Methods
- Perspectives

Exploring limits of computing!
Uncertainty Quantification with Multilevel Monte Carlo Methods

Uncertainties: Monte Carlo Sampling

- Simple (scalar) quantities of interest
- Sampling quantities of interest
- Complex high dimensional stochastic parameter space
- Requires multiple solves forward problem
- Forward problem expensive (3D PDE)

MC becomes quickly prohibitively expensive
Monte Carlo Method

\[ u_\ell = u_\ell(x, \omega) \quad \text{solution of} \quad M_\ell(u_\ell; \omega) = 0, \quad \ell \geq 0. \]

\( \{V_\ell\}_{\ell \geq 0} \) nested sequence of FE spaces

\( M_\ell \) degrees of freedom

Computational goal: \( \mathbb{E}[Q] \) of \( Q(u) \)

Standard Monte Carlo estimate

\[ \hat{Q}_{L}^{\text{MC}, N} = \frac{1}{N} \sum_{i=1}^{N} Q_{L}^{i} \]

Even if the solver has only linear complexity, the cost grows with \( O(M_\ell N) \)

There are two types of error involved:

- bias error (resolution of PDE to coarse)
- sampling error (too few samples)
Monte Carlo Method

\[ u_\ell = u_\ell(x, \omega) \quad \text{solution of} \quad M_\ell(u_\ell; \omega) = 0, \quad \ell \geq 0. \]

Approximation bias error

\[ |\mathbb{E}[Q_L - Q]| \leq C_b M_L^{-\alpha} \leq \varepsilon_b \]

convergence rate when mesh size is reduced, typical \( \alpha = \frac{1}{3}, \frac{1}{6} \)

Cost for solving one sample: \( \mathcal{O}(M_\ell^\gamma N) \) for \( \gamma \geq 1 \)

Total cost of Monte Carlo (MC) sampling:

\[ \text{Cost} \left( \hat{Q}_L^{MC,N} \right) = \mathcal{O}(M^\gamma N) = \mathcal{O}(\varepsilon^{-2-\gamma/\alpha}). \]

Total mean square error (MSE):

\[ e \left( \hat{Q}_L^{MC,N} \right)^2 := \mathbb{E}[(\hat{Q}_L^{MC,N} - \mathbb{E}[Q])^2] = (\mathbb{E}[Q_L - Q])^2 + N^{-1} \mathbb{V}[Q_L] \]

\( \mathbb{V}[Q_L] \) denotes the variance of the random variable \( Q_L(u_L) \)
Multilevel Monte Carlo (MLMC) Method

Use \( Y_\ell(\omega) := Q_\ell(u_\ell, \omega) - Q_{\ell-1}(u_{\ell-1}, \omega) \) and \( Y_0 := Q_0 \),

and the telescoping sum \( \mathbb{E}[Q_L] = \mathbb{E}[Q_0] + \sum_{\ell=1}^{L} \mathbb{E}[Y_\ell] \)

for the MLMC estimator \( \hat{Q}_L^{\text{ML}} := \sum_{\ell=0}^{L} \hat{Y}_{\ell}^{\text{MC}, N_\ell} \)

\[
\text{Cost}(\hat{Q}_L^{\text{ML}}) = \sum_{\ell=0}^{L} N_\ell C_\ell, \quad \text{number of samples} \quad N_\ell, \ell = 0, \ldots, L
\]

MLMC error \( e \left( \hat{Q}_L^{\text{ML}} \right)^2 = \left( \mathbb{E}[Q_L - Q] \right)^2 + \sum_{\ell=0}^{L} N_\ell^{-1} \mathbb{V}[Y_\ell] \)

We here profit from a hugely reduced variance since \( \mathbb{V}[Y_\ell] \to 0, \) as \( M_{\ell-1} \to \infty. \)
Multilevel Monte Carlo (MLMC) Method

Use \[ M_L \geq (\varepsilon_b/C_b)^{-1/\alpha}. \]

and exploit the fact that we have freedom to choose the number of samples \( N_\ell \) on each level to minimize the cost

\[
N_\ell = \varepsilon_s^{-2} \left( \sum_{\ell=0}^{L} \sqrt{\mathbb{V}[Y_\ell]C_\ell} \right) \sqrt{\frac{\mathbb{V}[Y_\ell]}{C_\ell}}
\]

Assuming \( C_\ell \leq C_c M_\ell^\gamma \) and \( \mathbb{V}[Y_\ell] \leq C_v M_\ell^{-\beta} \)

the cost is

\[
\text{Cost}(\hat{Q}_L^{\text{ML}}) = \varepsilon_s^{-2} \left( \sum_{\ell=0}^{L} \sqrt{\mathbb{V}[Y_\ell]C_\ell} \right)^2 \leq C_{ML} \varepsilon^{-2-\max(0, \frac{\gamma-\beta}{\alpha})}
\]

Typically \( \beta \approx 2\alpha \) for smooth functionals \( Q(\cdot) \)

When \( \gamma > \beta \) and \( \beta = 2\alpha \) then \( \text{Cost}(\hat{Q}_L^{\text{ML}}) = \mathcal{O}(\varepsilon^{-\gamma/\alpha}) \)

Two orders of magnitude faster than MC
Adaptive Multilevel Monte Carlo Method

Sample estimator for $\nabla [Y_\ell]$

$$s^2_\ell := \frac{1}{N_\ell} \sum_{i=1}^{N_\ell} (Y^i_\ell - \hat{Y}^{MC,N_\ell}_\ell)^2$$

and estimates for the cost $C_\ell$. (e.g. by measuring run times) we can construct the adaptive MLMC algorithm:

1. Set $\varepsilon, \theta, L = 1$ and $N_0 = N_1 = N_{\text{Init}}$.
2. For all levels $\ell = 0, \ldots, L$ do
   a. Compute new samples of $Y_\ell$ until there are $N_\ell$.
   b. Compute $\hat{Y}^{MC,N_\ell}_\ell$ and $s^2_\ell$, and estimate $C_\ell$.
3. Update the estimates for $N_\ell$ using (2.14) and
   if $\hat{Y}^{MC,N_L}_L > (8^\alpha - 1)\varepsilon_b$, increase $L \rightarrow L + 1$ and set $N_L = N_{\text{Init}}$.
4. If all $N_\ell$ and $L$ are unchanged,
   Go to 5.
   Else Return to 2.
5. Set $\hat{Q}^{ML}_L = \sum_{\ell=0}^{L} \hat{Y}^{MC,N_\ell}_\ell$. 
Modern FE Solvers:
Parallel Multigrid Methods


## Multi-PetaFlops Supercomputers

<table>
<thead>
<tr>
<th>Sunway TaihuLight</th>
<th>JUQUEEN</th>
<th>SuperMUC (phase 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW26010 processor</td>
<td>Blue Gene/Q architecture</td>
<td>Intel Xeon architecture</td>
</tr>
<tr>
<td>10,649,600 cores</td>
<td>458,752 PowerPC A2 cores</td>
<td>147,456 cores</td>
</tr>
<tr>
<td>260 cores (1.45 GHz) per node</td>
<td>16 cores (1.6 GHz) per node</td>
<td>16 cores (2.7 GHz) per node</td>
</tr>
<tr>
<td>32 GiB RAM per node</td>
<td>16 GiB RAM per node</td>
<td>32 GiB RAM per node</td>
</tr>
<tr>
<td>125 PFlops Peak</td>
<td>5D torus interconnect</td>
<td>Pruned tree interconnect</td>
</tr>
<tr>
<td>Power consumption: 15.37 MW</td>
<td>5.8 PFlops Peak</td>
<td>3.2 PFlops Peak</td>
</tr>
<tr>
<td>TOP 500: #1</td>
<td>TOP 500: #13</td>
<td>TOP 500: #27</td>
</tr>
</tbody>
</table>
How big PDE problems can we solve?

- 400 TByte main memory = $4 \times 10^{14}$ Bytes = 
  5 vectors each with $10^{13}$ elements 
  8 Byte = double precision

- Matrix-free implementation necessary
  - even with a sparse matrix format, storing a matrix of 
    dimension $10^{13}$ is not possible

- Asymptotically optimal algorithm needed (multigrid)
Multigrid: The algorithm for $10^{13}$ unknowns

Goal: solve $A^h u^h = f^h$ using a hierarchy of grids

1. Relax on
   $A^h u^h = f^h$
2. Residual
   $r^h = f^h - A^h u^h$
3. Restrict
   $r^H = I_H^h r^h$
4. Interpolate
   $e^h = I_H^h e^H$
5. Correct
   $u^h \leftarrow u^h + e^h$

Multigrid uses coarse grids to accomplish the inevitable global data exchange in the most efficient way possible.
Hierarchical Hybrid Grids (HHG)


- Parallelize „plain vanilla“ multigrid for tetrahedral finite elements
  - partition domain
  - parallelize all operations on all grids
  - use clever data structures
  - matrix free implementation

- Do not worry (so much) about coarse grids
  - idle processors?
  - short messages?
  - sequential dependency in grid hierarchy?

- Elliptic problems always require global communication. This cannot be accomplished by
  - local relaxation or
  - Krylov space acceleration or
  - domain decomposition without coarse grid
Regular tetrahedral refinement in HHG

Structured refinement of tetrahedra

- 3-D refinement hierarchy of tetrahedral patches
- Geometric hierarchy with 1-layer halos for
  - (volume) elements
  - faces
  - edges
  - vertices

communication of ghost layers
HHG: A modern architecture for FE computations

Geometrical Hierarchy: Volume, Face, Edge, Vertex
HHG Solver for Stokes System
Motivated by Earth Mantle convection problem


\[-\nabla \cdot (2\eta \varepsilon(u)) + \nabla p = \rho(T)g, \]
\[\nabla \cdot u = 0, \]
\[\frac{\partial T}{\partial t} + u \cdot \nabla T - \nabla \cdot (\kappa \nabla T) = \gamma. \]

Stokes equation:
\[-\text{div}(\nabla u - pI) = f, \]
\[\text{div} u = 0 \]

FEM Discretization:
\[a(u_l, v_l) + b(v_l, p_l) = L(v_l) \quad \forall v_l \in V_l, \]
\[b(u_l, q_l) - c(p_l, q_l) = 0 \quad \forall q_l \in Q_l, \]

with:
\[a(u, v) := \int_{\Omega} \nabla u : \nabla v \, dx, \quad b(u, q) := -\int_{\Omega} \text{div} u \cdot q \, dx \]

\[
\begin{bmatrix}
A_l & B_l^T \\
0 & C_l + B_l A_l^{-1} B_l^T
\end{bmatrix}
\begin{bmatrix}
u_l \\
p_l
\end{bmatrix}
= 
\begin{bmatrix}
f_l \\
B_l A_l^{-1} f_l
\end{bmatrix}
\]

Scale up to \(\sim 10^{13}\) nodes/DOFs
⇒ resolve the whole Earth Mantle globally with 1km resolution

**u** velocity
**p** dynamic pressure
**T** temperature
**\nu** viscosity of the material
**\varepsilon(u) = \frac{1}{2}(\nabla u + (\nabla u)^T)** strain rate tensor
**\rho** density
**\kappa, \gamma, g** thermal conductivity, heat sources, gravity vector
Exploring the Limits …


- Multigrid with Uzawa smoother
- Optimized for minimal memory consumption
  - $10^{13}$ Unknowns correspond to 80 TByte for the solution vector
  - Juqueen has 450 TByte Memory
  - matrix free implementation essential

<table>
<thead>
<tr>
<th>nodes</th>
<th>threads</th>
<th>DoFs</th>
<th>iter</th>
<th>time</th>
<th>time w.c.g.</th>
<th>time c.g.</th>
<th>in %</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>80</td>
<td>$2.7 \cdot 10^9$</td>
<td>10</td>
<td>685.88</td>
<td>678.77</td>
<td>1.04</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>640</td>
<td>$2.1 \cdot 10^{10}$</td>
<td>10</td>
<td>703.69</td>
<td>686.24</td>
<td>2.48</td>
<td></td>
</tr>
<tr>
<td>320</td>
<td>5120</td>
<td>$1.2 \cdot 10^{11}$</td>
<td>10</td>
<td>741.86</td>
<td>709.88</td>
<td>4.31</td>
<td></td>
</tr>
<tr>
<td>2560</td>
<td>40960</td>
<td>$1.7 \cdot 10^{12}$</td>
<td>9</td>
<td>720.24</td>
<td>671.63</td>
<td>6.75</td>
<td></td>
</tr>
<tr>
<td>20480</td>
<td>327680</td>
<td>$1.1 \cdot 10^{13}$</td>
<td>9</td>
<td>776.09</td>
<td>681.91</td>
<td>12.14</td>
<td></td>
</tr>
</tbody>
</table>
Uncertainty Quantification

Scheduling Parallel Multilevel Monte Carlo Methods

Model problem from subsurface flow

Find $u(\cdot, \omega) \in V := H^1_0(D)$ such that

$$
\int_D \nabla v(x) \cdot (k(x, \omega) \nabla u(x, \omega)) \, dx = \int_D f(x) v(x) \, dx, \quad \text{for all } v \in V \text{ and } \omega \in \Omega
$$

• motivated from subsurface flow
• solution $u$ and coefficient $k$ are 3D stochastic fields
• quantity of interest

$$
Q(u) := u(x^*) \quad \text{or}
$$

$$
Q(u) := \frac{1}{|\Gamma|} \int_{\Gamma} -k \frac{\partial u}{\partial n} \, ds \quad \text{for some 2D manifold } \Gamma \subset \overline{D}
$$

• Discretization with tetrahedral FE, using HHG multigrid solver
• additionally, generation of samples of $k(\cdot, \omega) := \exp(Z(\cdot, \omega))$

by solving stochastic PDE

$$
(\kappa^2 - \Delta) Z(x, \omega) =^d W(x, \omega)
$$

$W$: Gaussian white noise with unit variance
Scheduling Multilevel Monte Carlo

We must exploit three levels of parallelism:
- levels may be executed in parallel
- the samples within a level may be executed in parallel
- the solver may be executed in parallel

Finding a good schedule is NP complete: must use heuristics
Scheduling Multilevel Monte Carlo

Solver times themselves are stochastic

- level synchronous homogeneous
- sample synchronous homogeneous
- dynamic sample synchronous homogeneous

It is essential to make flexible use of solver parallelism, but
- not all processor counts possible (e.g. power of 2)
- memory limitations
- strong scaling limitations (loss of efficiency)
- technical limitations (MPI communicators)
Performance results

- Adaptive MLMC
- estimating the variance for each level $s_\ell^2 := \frac{1}{N_\ell} \sum_{i=1}^{N_\ell} \left( Y^i_\ell - \hat{Y}^{\text{MC},N_\ell}_\ell \right)^2$
- and the cost of computing samples $C_\ell$.
- dynamical estimate of the number of required samples

<table>
<thead>
<tr>
<th>Processes</th>
<th>Resolution</th>
<th>Runtime</th>
<th>No. Samples</th>
<th>Correlation</th>
<th>Idle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fine</td>
<td>Total</td>
<td>length</td>
</tr>
<tr>
<td>4096</td>
<td>$1024^3$</td>
<td>$5.0 \cdot 10^3$ s</td>
<td>68</td>
<td>13 316</td>
<td>1.50E-02</td>
</tr>
<tr>
<td>32 768</td>
<td>$2048^3$</td>
<td>$3.9 \cdot 10^3$ s</td>
<td>44</td>
<td>10 892</td>
<td>7.50E-03</td>
</tr>
<tr>
<td>262 144</td>
<td>$4096^3$</td>
<td>$5.2 \cdot 10^3$ s</td>
<td>60</td>
<td>10 940</td>
<td>3.75E-03</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level</th>
<th>No. partitions</th>
<th>Scheduled</th>
<th>Calculated</th>
<th>Estimated</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2 048</td>
<td>7 506</td>
<td>8 192</td>
<td>3 726</td>
<td>686</td>
</tr>
<tr>
<td>1</td>
<td>256</td>
<td>2 111</td>
<td>2 304</td>
<td>429</td>
<td>193</td>
</tr>
<tr>
<td>2</td>
<td>32</td>
<td>382</td>
<td>384</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>57</td>
<td>60</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
Conclusions
Summary

- Non-intrusive MC sampling can be prohibitively expensive
- Intrusive MLMC methods may have much better complexity
- Multigrid methods and MLMC rely on the same hierarchical structure
- Parallel multigrid can solve PDE with $10^{13}$ unknowns
- MLMC parallelism requires complex scheduling strategies
- Solving demonstrated for
  - 10,000 samples
  - largest samples with $7 \times 10^{10}$ unknowns
  - on >250,000 processors
  - in <1.5 hours
Thank you for your attention!

Videos, preprints, slides at https://www10.informatik.uni-erlangen.de