Computer Physics Communications 192 (2015) 138–147

Contents lists available at ScienceDirect

Computer Physics Communications

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

Distributed Database Kriging for Adaptive Sampling (D^2KAS)

Dominic Roehm [a,c,*], Robert S. Pavel [b,e], Kipton Barros [c], Bertrand Rouet-Leduc [d,c], Allen L. McPherson [e], Timothy C. Germann [c], Christoph Junghans [e]

[a] *Institute for Computational Physics, Universität Stuttgart, 70569 Stuttgart, Germany*
[b] *Department of Electrical and Computer Engineering, University of Delaware, Newark, DE 19716, USA*
[c] *Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA*
[d] *Department of Materials, Science and Metallurgy, University of Cambridge, Cambridge, CB3, UK*
[e] *Computer and Computational Sciences Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA*

A R T I C L E   I N F O

Article history:
Received 13 October 2014
Accepted 10 March 2015
Available online 18 March 2015

Keywords:
Adaptive sampling
Heterogeneous Multiscale Methods
Elastodynamics
Cloud database
Kriging

A B S T R A C T

We present an adaptive sampling method supplemented by a distributed database and a prediction method for multiscale simulations using the Heterogeneous Multiscale Method. A finite-volume scheme integrates the macro-scale conservation laws for elastodynamics, which are closed by momentum and energy fluxes evaluated at the micro-scale. In the original approach, molecular dynamics (MD) simulations are launched for every macro-scale volume element. Our adaptive sampling scheme replaces a large fraction of costly micro-scale MD simulations with fast table lookup and prediction. The cloud database Redis provides the plain table lookup, and with locality aware hashing we gather input data for our prediction scheme. For the latter we use kriging, which estimates an unknown value and its uncertainty (error) at a specific location in parameter space by using weighted averages of the neighboring points. We find that our adaptive scheme significantly improves simulation performance by a factor of 2.5–25, while retaining high accuracy for various choices of the algorithm parameters.

1. Introduction

A fundamental challenge in computational materials science is to span the enormous gap between atomic and engineering scales. Hence, one of the major computational challenges is found in handling the sample size, since a massive number of atoms ($O(10^{23})$) is necessary to represent even a micro-scale piece of material. Many physical phenomena are more successfully described on the continuum level, in which a large collection of statistically homogeneous atoms can be described with a few physical values. Such a description can be effective to investigate the basic properties of ideal materials. However, in most materials systems, properties are crucially affected by atomic and micro-scale inhomogeneities. To incorporate atomistic details, simulations tend to be more appropriate than a theoretical approach. Unfortunately, it is completely impractical to simulate any sizable piece of material in full atomistic detail for a time-scale of seconds using molecular dynamics (MD) simulation, even on modern supercomputers. The development of coarse-graining techniques is therefore a highly active field [1–5], which are applicable for adaptive mesh refinement [6–8] and machine learning [9,10]. These techniques leverage scale separation to effectively integrate out fine-scale degrees of freedom, and to produce a statistical description of the dynamics at the macro-scale. In multiscale simulations the coarse-grained macro-scale dynamics is enhanced with constitutive data supplied at the micro-scale. In this work we focus on accelerating multiscale simulations with adaptive sampling, which seeks to improve efficiency by eliminating unnecessary micro-scale simulation without neglecting essential properties of the material.

A promising approach to multi-scale modeling is the Heterogeneous Multiscale Method (HMM) [11–13]. Here, "heterogeneous" emphasizes that the models at different scales may be of very different nature, e.g., molecular dynamics at the micro scale and continuum mechanics at the macro scale. A similar and concurrently developed framework is the "equation free" approach [14]. HMM assumes a well defined set of equations at the macroscale. In our case, these are the multidimensional hyperbolic conservation laws, which we evolve using a finite volume scheme, specifically a non-oscillatory central scheme [15–17]. Separation of time and length scales is assumed. This allows for independent micro-scale

* Corresponding author at: Institute for Computational Physics, Universität Stuttgart, 70569 Stuttgart, Germany. Tel.: +49 711 685 67705.
E-mail address: dominic.roehm@icp.uni-stuttgart.de (D. Roehm).

http://dx.doi.org/10.1016/j.cpc.2015.03.006
0010-4655/© 2015 Elsevier B.V. All rights reserved.

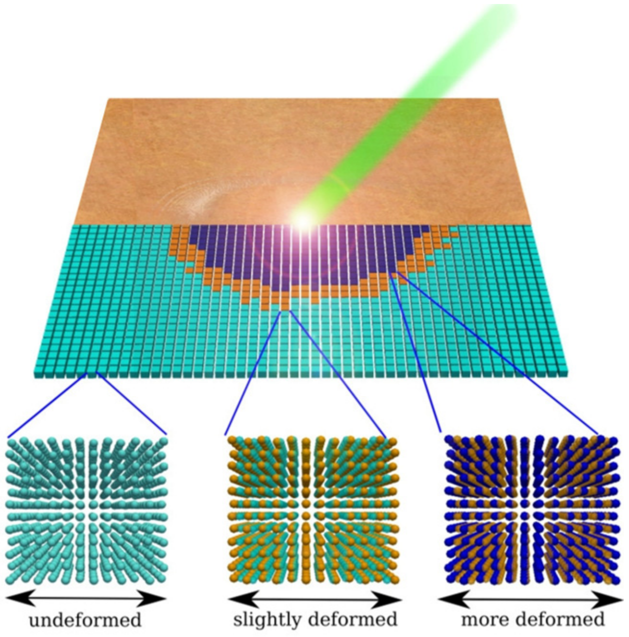

Fig. 1. Schematics of the physical problem considered here. Impact (laser beam) on a copper plate. The finite volume cubes represent the material on the continuum level, while the microscopic response of each volume element is provided by MD simulations of the underlying copper crystal.

simulations, which are connected at the macro scale as input for the continuum description.

We applied HMM to a set of differential equations for elastodynamics in order to model deformation-propagation in solid materials, specifically a defect-free copper crystal, including temperature and stochastic effects, as illustrated in Fig. 1. Simulating a macro-scale sample with this level of detail using pure MD simulations would be impossible even on state-of-the-art high performance computers. Despite the enormous efficiency gains enabled by HMM, spawning a micro-scale simulation at every macro-scale volume element at every time-step remains prohibitively expensive in many cases.

We demonstrate that adaptive sampling methods can further boost HMM performance by reducing the number of micro-scale simulations executed in each time step. This work is complementary to previous adaptive sampling methods [18–20], that reduce the number of micro-scale simulations by means of prediction or interpolation on the macro scale.

In this contribution we will introduce our implementation of an adaptive sampling scheme, which we call Distributed Database Kriging for Adaptive Sampling (D^2KAS). The method is based on the adaptive sampling method introduced by Knap [18] and Barton et al. [19]. These authors used a prediction scheme based upon a metric-tree database, while assuming a deterministic system. In our approach, we applied the prediction scheme to the HMM model for stochastic data supported by a cloud database. The adaptive sampling method is based on the "best linear unbiased prediction", also known as kriging. Kriging in context of machine learning has been applied to a vast amount of different problems, ranging from its origins in geostatistics for environmental data mining and modeling [21] to multipolar electrostatics [22] computations or constructions of accurate polarizable water potentials [23]. In our approach we utilize the high-performance cloud database Redis [24] to store the results of the MD simulations, which we either use directly as a lookup table or as input for kriging. D^2KAS also uses a basic mapping to avoid duplicated MD as well as kriging interpolations to reduce the total number of parallel tasks.

In this paper, we will show by means of two test problems that our adaptive scheme is sufficiently robust and efficient and

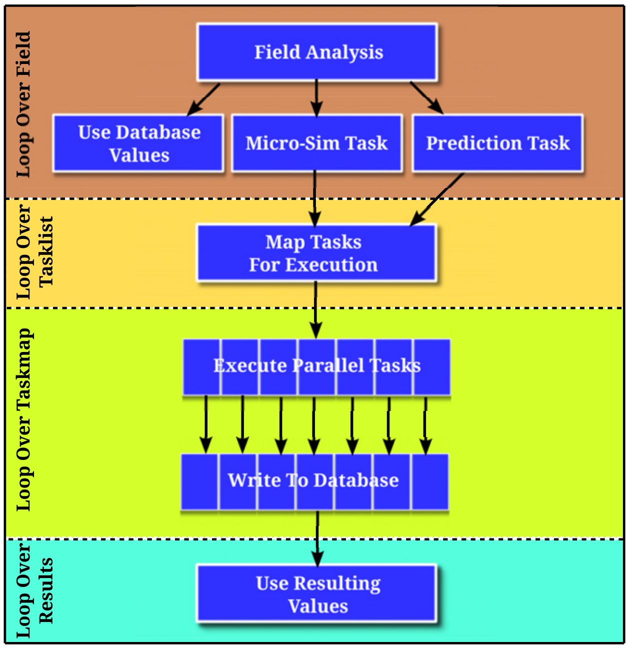

Fig. 2. Schematic of the implementation of the Distributed Database Kriging for Adaptive Sampling (D^2KAS). Execution starts after some initial routines with a check of the database (top of diagram). The execution scheme shows one loop for computing a half step in time. A detailed description of the single loops is given in Section 4.

provides a general framework for different prediction methods and cloud databases. Our approach is suitable for emerging programming models, highly parallel and heterogeneous computing architectures, and implementation with task-based programming languages, libraries, and runtime systems such as Charm++ [25] or Intel's CnC [26].

The paper is structured as follows: Section 2 starts with a short survey and some general remarks about the D^2KAS workflow. In Section 3, we take a look at the general HMM framework and its application to elastodynamics. An introduction about kriging follows in Section 3.3, and in Section 4 the key-value database Redis usage as well as the implementation and its features, which include gradient analysis and task mapping are explained. Section 5 presents two test problems, which are used to demonstrate our method and assess the performance of the adaptive sampling and its dependence on different numerical tolerance. Finally, in Section 6 we close with an outlook on possibilities and challenges the D^2KAS framework provides.

2. D^2KAS in a nutshell

The basic workflow diagram in Fig. 2 shows the main loop of our adaptive sampling method. Assuming a simulation of an area divided into small domains (see Fig. 1), we need to compute fluxes (e.g. the stress tensor) through the surfaces of the volume based on conserved quantities (e.g. the strain tensor) inside the volume (Section 3). To gather the necessary data for each volume we have three possibilities: First, use a result stored in our database, accessed by a key based on the conserved quantities (Section 4); Second, estimate the fluxes with the help of kriging based on a subset of results stored in the database (Section 3.3); Third, compute new fluxes by executing a micro-scale simulation (Section 3.1). The results of the MD runs are stored in the database using a different set of keys as the kriging results. This avoids the use of estimated values as input for the kriging.

Fig. 2 shows the path of decisions made during the update loop over the field. We start with checking the database for results which match our input data. If this was not successful, we apply our gradient analysis of the stress field to decide whether we will try to estimate the result using kriging or if we have to execute an MD simulation. If the gradient is smaller than a certain threshold we add the input data to a task list collecting all kriging tasks. A similar list collects all the MD simulations that have to be executed. After mapping the tasks to remove duplicated ones, we start all tasks in parallel. If a kriging task has not been successful, which depends on the computed error, an MD simulation is immediately started. After all tasks have been accomplished the results are distributed back to the field. Further information about the implementation and a detailed schematic are given in Section 4. We utilized a cloud database, which can be accessed by the parallel tasks to gather information as well as write results to it. This general scheme is based on a macro-scale framework solving partial differential equations, which is supported by a highly parallel database and a dynamic runtime that spawns independent micro tasks for parallel execution. Reducing costly micro tasks by a prediction method using the results stored in the database leads to a further boost in the performance of our model. In our specific case we use a finite volume solver with a non-oscillatory central scheme for hyperbolic differential equations on the macroscale, solving a set of PDE for elastodynamics as described in the next section. The simulations on the micro-scale are classical molecular dynamics simulations, executed by the proxy application CoMD [27]. As prediction method we use kriging, a best linear unbiased predictor. The novelty of our approach is utilizing a distributed cloud database for both table lookup and input for the kriging in combination with adaptive sampling.

3. Method

3.1. HMM for elastodynamics

In the present work, we take a look at two test scenarios with different complexity. Both are describing an elastic wave propagation in a perfect crystal with finite temperature. Based on the continuum description with macroscopic conservation laws HMM incorporates momentum and energy fluxes calculated microscopically using time-averaged MD computations. One assumes that the microscopic MD simulations are ergodic and that the microscopic quantities are unique functions of the macroscopic inputs, although finite sampling times introduce stochastic noise into the results, as opposed to the deterministic fine-scale models used in previous adaptive sampling works [19,18]. Therefore, it is valid to extend the microscopically averaged quantities e.g. stress, momentum and energy density, to the macroscopic scale [12].

Starting with the micro-scale molecular dynamics, e.g. Newton's equations, the following set of conservation laws

$$\partial_t \rho + \nabla \cdot \mathbf{q} = 0 \tag{1}$$

$$\partial_t \mathbf{q} + \nabla \cdot \tau = 0 \tag{2}$$

$$\partial_t e + \nabla \mathbf{j} = 0, \tag{3}$$

may be derived [28]. The mass, momentum, and energy densities are given as distributions,

$$\rho(\mathbf{r}, t) = \sum_i m_i \delta(\mathbf{r} - \mathbf{r}_i(t)) \tag{4}$$

$$\mathbf{q}(\mathbf{r}, t) = \sum_i m_i \mathbf{v}_i(t) \delta(\mathbf{r} - \mathbf{r}_i(t)) \tag{5}$$

$$e(\mathbf{r}, t) = \sum_i \left[\frac{1}{2} m_i \mathbf{v}_i^2 + \frac{1}{2} \sum_{j \neq i} \phi(\mathbf{r}_{ij}(t)) \right] \delta(\mathbf{r} - \mathbf{r}_i(t)), \tag{6}$$

where m_i, $\mathbf{r}_i$, and $\mathbf{v}_i$ are the mass, position, and velocity of particle i and $\phi(\mathbf{r}_{ij}(t))$ is the pairwise interaction potential. An extension to more complex potential forms is straight forward.

The fluxes are spatial averages over the domain Ω. The average momentum flux is

$$\tau_\Omega = -\frac{1}{|\Omega|} \sum_{i,j} c_{ij} \left[\delta_{ij} m_i \mathbf{v}_i \otimes \mathbf{v}_i + \mathbf{f}_{ij} \otimes \mathbf{r}_{ij} \right], \tag{7}$$

and the energy flux is

$$\mathbf{j}_\Omega = \frac{1}{2|\Omega|} \sum_{i,j} c_{ij} \left\{ \mathbf{v}_i \left[\frac{1}{2} m_i \mathbf{v}_i^2 \delta_{ij} + \phi(\mathbf{r}_{ij}) \right] - \frac{1}{2} (\mathbf{v}_i + \mathbf{v}_j) \cdot \mathbf{f}_{ij} \mathbf{r}_{ij} \right\}. \tag{8}$$

The force $\mathbf{f}_{ij}$ represents the interaction between atom i and atom j. The parameter $0 < c_{ij} \leq 1$ represents the fraction of the line connecting $\mathbf{r}_i$ and $\mathbf{r}_j$ that lies within Ω, which results in $c_{ij} = 1$ in case of periodic boundary conditions.

Eqs. (1)–(8) represent the basis of the HMM concept. While the conserved fields ρ, $\mathbf{q}$, and e are integrated on macroscopic space and time scales, the fluxes τ and $\mathbf{j}$ are estimated as statistical MD averages on microscales. In fact, we average over length and time scales much smaller than what is being represented at the macro scale. The validity of HMM is based on the ergodicity of MD, a strong separation of both time and length scales, and the assumption of local equilibrium.

3.2. Non-oscillatory central schemes

The use of non-oscillatory central schemes reduces computational costs while providing numerical stability. A second order scheme for the two-dimensional case was published by Jiang and Tadmor [15]. The scheme is a two-step predictor–corrector method. The first step is based on given cell averages whose change in time are estimated with the help of computed fluxes through the surfaces of the cells. Afterwards, the corrector step uses so-called staggered averaging to correct the predicted midvalues and to realize the evolution of these averages. For the present problem a two-dimensional system of conservation laws can be written as:

$$\partial_t \mathbf{w} + \partial_x f(\mathbf{w}) + \partial_y g(\mathbf{w}) = 0. \tag{9}$$

In [15] the time derivative of $\partial_t \mathbf{w}$ written in terms of spatial derivatives $\partial_x f(\mathbf{w}')$ and $\partial_y g(\mathbf{w}`)$ is given as:

$$w_{jk}^{n+\frac{1}{2}} = \bar{w}_{jk}^n - \frac{\lambda}{2} f(w')_{jk} - \frac{\mu}{2} g(w`)_{jk}, \tag{10}$$

To compute the midvalues (10), the approximate fluxes have to be evaluated. Using these values together with the staged average the corrector step reads:

$$\begin{aligned}
\bar{w}_{j+\frac{1}{2},k+\frac{1}{2}}^{n+1} = &\frac{1}{4}\left(\bar{w}_{jk}^n + \bar{w}_{j+1,k}^n + \bar{w}_{j,k+1}^n + \bar{w}_{j+1,k+1}^n\right) \\
&+ \frac{1}{16}\left(w'_{jk} - w'_{j+1,k} + w'_{j,k+1} - w'_{j+1,k+1}\right) \\
&+ \frac{1}{16}\left(w`_{jk} - w`_{j,k+1} + w`_{j+1,k} - w`_{j+1,k+1}\right) \\
&- \frac{\lambda}{2}\left[f\left(w_{j+1,k}^{n+\frac{1}{2}}\right) - f\left(w_{j,k}^{n+\frac{1}{2}}\right) + f\left(w_{j+1,k+1}^{n+\frac{1}{2}}\right) - f\left(w_{j,k+1}^{n+\frac{1}{2}}\right)\right] \\
&- \frac{\mu}{2}\left[g\left(w_{j,k+1}^{n+\frac{1}{2}}\right) - g\left(w_{j,k}^{n+\frac{1}{2}}\right) + g\left(w_{j+1,k+1}^{n+\frac{1}{2}}\right) - g\left(w_{j+1,k}^{n+\frac{1}{2}}\right)\right],
\end{aligned} \tag{11}$$

where $\lambda = \Delta t/\Delta x$ and $\mu = \Delta t/\Delta y$. As indicated by $j + 1$ and $k + 1$, the new calculated values of $\bar{w}^{n+1}$ are on the surface, which means that this is only a half step in space. Fig. 3 illustrates the

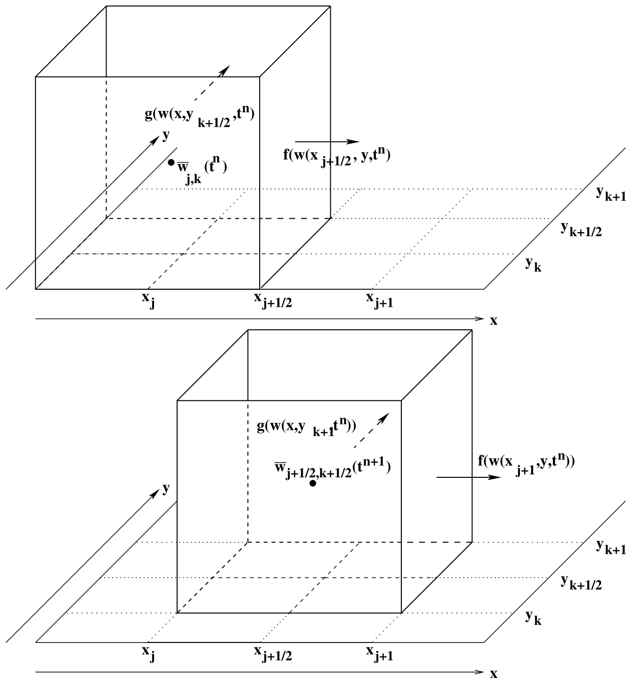

Fig. 3. Illustration of the fluxes f and g through the surface and the conserved quantities w inside the volume.

conserved quantities inside the volume and the fluxes through the surface of the volume for one half step. For the full step we need to execute these half steps twice. A difficult question is what the spatial derivative looks like on the mesh point. Ref. [15] used the MinMod limiter scheme to reduce high frequency oscillations and increase stability [29,30]. The MinMod scheme relies on the maximum principle for scalar approximations and the direct slopes for w' and $w`$ are:

$$w'_{jk} = \mathrm{MM}\left\{\bar{w}^n_{j+1,k} - \bar{w}^n_{j,k}, \frac{1}{2}(\bar{w}^n_{j+1,k} - \bar{w}^n_{j-1,k}), \bar{w}^n_{j,k} - \bar{w}^n_{j-1,k}\right\}, \quad (12)$$

and

$$w`_{jk} = \mathrm{MM}\left\{\bar{w}^n_{j,k+1} - \bar{w}^n_{j,k}, \frac{1}{2}(\bar{w}^n_{j,k+1} - \bar{w}^n_{j,k-1}), \bar{w}^n_{j,k} - \bar{w}^n_{j,k-1}\right\}. \quad (13)$$

where

$$\mathrm{MM}\{v_1, v_2, \ldots\} = \begin{cases} \min_p\{v_p\} & \text{if } v_p > 0\ \forall p, \\ \max_p\{v_p\} & \text{if } v_p < 0\ \forall p, \\ 0 & \text{otherwise.} \end{cases} \quad (14)$$

For more details about the staggered average corrector scheme we refer the reader to [15].

3.3. *Kriging*

The use of the word "kriging" in (geo-)statistics stands for "optimal prediction". The power of this method lies in its ability to treat not only scalars, but also vectors located in a high-dimensional space to predict the value of a vector located at a certain position in the high-dimensional space by computing a weighted average of the known vectors in the neighborhood of the point. Making use of matrix representations for the formulas allows not only for an efficient implementation for the computation of the predicted values, but computes an error of the estimation at the same time. In this paper, we only give a brief introduction about ordinary kriging, further information can be found e.g. in [31–33].

Suppose prediction of the value $Z(\mathbf{s}_0)$ at some spatial location $\mathbf{s}_0$ is desired using a linear function of the data of the form

$$Z'(\mathbf{s}_0) = \sum_{i=1}^{n} \lambda_i Z(\mathbf{s}_i) \quad (15)$$

which is both unbiased and minimizes the mean-square error

$$\sigma_k^2(\mathbf{s}_0) = [Z(\mathbf{s}_0) - Z'(\mathbf{s}_0)]^2. \quad (16)$$

Then the optimal λs satisfy an $(n+1)$-dimensional linear equation. The minimized mean-square error is given by

$$\sigma_k^2(\mathbf{s}_0) = \sum_{i=1}^{n} \lambda_i \gamma(\mathbf{s}_i - \mathbf{s}_j) + m, \quad (17)$$

where λ_i are the Lagrangian parameters and γ is the variogram. There are several parameters affecting the result and the error estimation of kriging. In fact, all the parameters are contained in the computation of the theoretical variogram [31], which is a function describing the degree of spatial dependence of a spatial random field or stochastic process, in the present case $Z(\mathbf{s}_0)$. But the empirical variogram cannot be computed at every lag distance h (distance between two sample points) and due to variation in the estimation it is not ensured that every variogram is valid. However, kriging only works with valid variograms. To overcome this problem the empirical variograms are thus often approximated by a model function that ensures validity [32]. We use the following (spherical) variogram model [33]:

$$\gamma(h) = (s-n)\left[\left(\frac{3h}{2r} - \frac{h^3}{r^3}\right) 1_{0,r}(h) + 1_{[,\infty)(h)}\right] + n 1_{(0,\infty)}(h). \quad (18)$$

The parameters of the variogram are:

1. nugget n: The measurement errors occurring as offset of the variogram at the origin.
2. sill s: The convergence value of the variogram (variance).
3. range r: The distance in which the variogram (almost) equals the value of the sill.

For our purpose the spherical variogram model has been the best choice, since the error estimation needs to be symmetric, too. For two other models (Gaussian, exponential), we tested the values themselves were symmetric with respect to negative values, but the error estimations showed small differences which led to an asymmetric acceptance of the predicted values. For the present study, we used the spherical model of Eq. (18) with $n = 0.0$, $s = 1.0$ and $r = 0.125$, leading to the best prediction in our test problems.

4. Implementation

The key to our approach lies in the use of locality-aware hashes [34,35] and the database's set data structure. Essentially, we create a range along all seven dimensions (e.g. four components of the strain tensor $\mathbf{A}$, two dimensional momentum density $\mathbf{q}$ and energy density e) of our conserved vector and generate a truncated hash for each data point, which corresponds to a conserved vector.

While our method could be adapted for many different types of database, we determined that the following characteristics were needed:

1. Key-value storage.
2. Distributed.
3. Multiple values allowed per key.

Key-value storage is essential for fast generation of matching input and output fields, while the ability to allow for multiple values per key is necessary for locality-aware hashing. As such, we chose to avoid traditional SQL databases and instead utilize a NoSQL database [36] with a key-value store approach [37,38]. More specifically, we chose the open-source database Redis [24] due to its demonstratively high performance and the ability to use its set data structures to create buckets of nearest neighbors. Redis is an open-source NoSQL database [36] that maintains the database in-memory and supports access across a network and, optionally, a distributed database with consistency and coherence. While the core is written in ANSI C, there are interfaces for most modern languages. Specifically, we used hiredis [39], a light-weight C interface, for this implementation. Redis maintains the whole dataset in memory to decrease access times and increase performance. Persistence, and fault tolerance, is achieved through a combination of snapshotting, where the dataset is periodically written to disk, and an approach that records each operation in an on-disk journal and periodically applies the operations to conserve space. This combination greatly decreases the time required to store the database while also ensuring that the current state can be reconstructed at any point.

For the purpose of scalability, Redis utilizes a hierarchical master–slave replication [24] in which each instance of Redis is capable of being both a master of some instances and a slave to another. This allows data to be replicated across the entire distributed database for the purpose of scalability and resiliency through data redundancy. The use of a distributed database is essential due to the large number of TCP/IP connections requested by the parallel tasks.

When processing a point, we generate the set key based upon the conserved vector to obtain the bucket and check the CoMD database, a database consisting of previously computed values obtained via CoMD [27]. CoMD is a molecular dynamics proxy application, which we modified for our specific problem, as described in Section 5. Prior to returning the results of the execution, we sort the conserved vectors with respect to their distance to the actual value so that the first returned value can be checked to determine if it fulfills a threshold, which can be zero, to see if the computed result is already in the database. If the first returned value falls within the threshold, we can write the result directly to the field.

Otherwise, we then check the gradient in four directions. If the gradient smaller than a specific threshold, we repeat the process with the kriging database. If it is not, we use CoMD to compute said point. Furthermore, the gradient threshold is dynamically adjusted, based on the number of successful kriging tasks. E.g. if all kriging tasks have been successful the gradient threshold gets increased. The initial value of the kriging threshold depends on the initial strength of the strain. Adjusting the kriging threshold dynamically during the runtime ensures that the overhead due to unsuccessful kriging tasks is minimized (see below).

The kriging database behaves similarly to the CoMD database, but only contains the results of kriging. In fact, if we speak about a kriging database we are using the same Redis database but with another set of keys for the kriging results. Once again we check if the nearest neighbor falls within the threshold. If not, we perform kriging to predict the value of the point, as previously discussed. If this prediction does not fall within a specific error threshold, CoMD is called on the point (see Fig. 4).

Our implementation is designed around a task-based approach, with two major types of tasks with drastically varying execution lengths (shown below in Fig. 9). First a serial CoMD database task checks for input for the entire macro solver field in the database. Afterwards, the first tasks added to the parallel execution map are the kriging tasks, which consist of, first, checking the kriging database and then either using said value as a result or using kriging itself to compute a value. If the value is not sufficient, the kriging task becomes a CoMD task. The second type added to the task list are the CoMD tasks, consisting of calls to the CoMD library to compute the value and writing the result into the database.

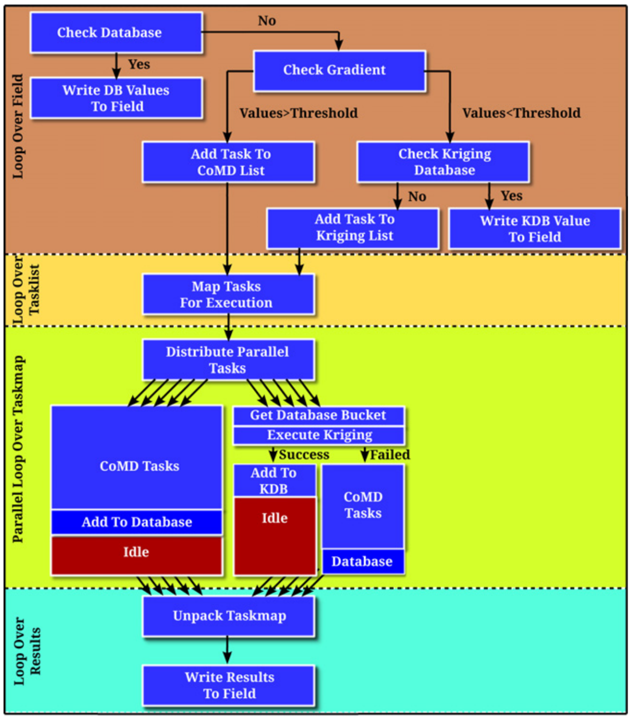

Fig. 4. Schematic of the implementation of the kriging supported adaptive sampling (KAS) model for stochastic data. Execution starts after some initial routines with a check of the database (top of diagram). The execution scheme shows one loop for computing Eq. (10) or Eq. (11) for a half step in time. Therefore, to complete one total time step, it is necessary to execute four times the illustrated scheme over the field.

By utilizing a task-based approach we are able to take advantage of the largely embarrassingly parallel nature of HMM while also accommodating for the drastically varying execution times of the tasks. Since, any given CoMD task takes approximately the same amount of time, a kriging task may consist solely of a database access and the kriging function, and a CoMD call as well. Additionally, the database access itself involves communication and is subject to contention.

However, for the purpose of this paper we focused primarily on the benefits of the kriging technique itself as opposed to any additional benefits that can be gained through more complex runtimes. As such, all data have been obtained using OpenMP [40] and the runtime environment Charm++ [25]. Future work will consider and utilize more runtimes that can fully take advantage of our scheme [41].

5. Results

In this section we will present the performance and the stability of the method described above. To do this, we investigated the behavior of the system for two different test problems. The first test problem (TP1) is a quasi-1D wave propagation as illustrated in Fig. 5. The second test problem (TP2) is a circular impact, affecting all components of the strain tensor (see Fig. 1 time series in Fig. 6).

Unless otherwise specified, the following setup was used. The MD simulations are based on a defect-free crystal of copper using an embedded atom method (EAM) interatomic potential [42]. The periodic simulation domain consists of $6 \times 6 \times 6$ fcc unit cells with lattice constant $a = 3.618$ Å. The output of the estimated

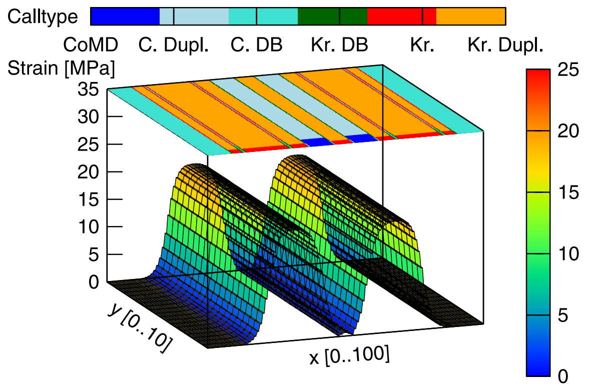

Fig. 5. Illustration of the stress (z-axis) and the corresponding tasks (colormap at the top), for TP1, with a two-dimensional grid containing 100 × 10 grid points. Dark blue are the CoMD hits, light blue the similar CoMD hits, which are evaluated with the mapping procedure. Turquoise are database values and green the kriging database values. The kriging hits are orange and red are the similar kriging hits, which are evaluated with the mapping procedure like the CoMD hits. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

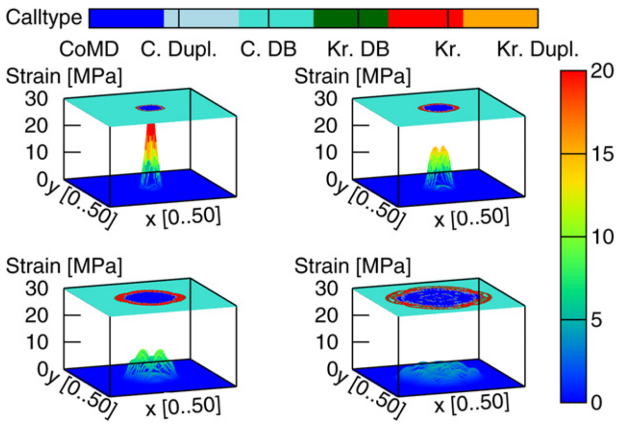

Fig. 6. The figures illustrate a time series of a TP2 simulation with circular initial conditions of the strain with radius 5 for timesteps 1, 19, 49 and 99. The two-dimensional grid has 50 × 50 grid points. The colorscheme at the top indicates the different hit types (dark blue: CoMD, orange: kriging, turquoise: database), while the illustration on the bottom shows the strength of the strain. The anisotropy due to the copper crystals strain–stress relation at the MD level is leading to the observed asymmetric response. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

virial stress and the energy density are a time-average over the last 20% of our MD simulation. We integrate for a time $\tau = 10$ ps using velocity-Verlet time steps of size $\Delta t = 10$ fs. On the macro scale level we introduce a strained region in an unstrained system thermalized to the temperature 129 K. This equilibrium configuration has energy density $e = -0.296$ eV/Å^3. The time and length scales of the macro solver can be independent of the micro scale, but with no explicit system in mind we do not introduce any separation of time and length scale. Therefore, the total area on the macro scale corresponds to either 217.08 nm × 21.708 nm (TP1) or 108.5 nm × 108.5 nm (TP2), both for total simulation time of 0.5 µs.

Table 1 shows the wall-clock simulation time for our two test problems (TP1 and TP2) using various feature combinations. The direct numerical simulation (DNS) result is the time that would have been required if CoMD simulations were always executed at every volume element, every timestep, as in the original HMM approach. In case of mapping, CoMD tasks with duplicated input parameters are removed. The third case utilizes the CoMD database (CoMD DB) on top of the mapping scheme. Finally, the remaining cases are using the kriging feature (with duplicate mapping for kriging tasks as well), which can be supplemented by a kriging database (Kr. DB).

Mapping helps to reduce 90% of the execution time for TP1 (with perfect symmetry in the y-direction), but has no benefit for TP2. When the number of compute nodes/cores n_p is smaller ($n_p < n_t$) than the maximum number of independent tasks n_t, enabling the CoMD database leads to an speedup of ≈65% on top of the mapping speedup. And furthermore, the kriging predictor leads to another speedup up to ≈49% in the first case. The kriging database feature must be treated carefully, since it can even increase the total simulation time, caused by too few successful hits. In case of an equal number ($n_p = n_t$) of nodes/cores and tasks only minor differences appear, since the total time is limited by the execution time of the CoMD tasks (see Table 1 for details).

Table 1
The table shows the absolute simulation time for our two test problems (TP1 *HMM step*/$N = 0.4$, TP2 *HMM step*/$N = 0.08$) having different features disabled/enabled and different compute node/cores to tasks ratios. The results are shown in seconds. The first row shows the result for the DNS run, e.g. only CoMD tasks are executed. The measurements in case of mapping enabled are done by removing all duplicated tasks. The third row shows the results with enabled CoMD database. The execution times for the runs with enabled kriging as well as enabled kriging database are shown in the last two rows. All measurements were executed on AMD Opteron Processor nodes.

Features	TP1 $n_p = n_t$	TP1 $n_p < n_t$	TP2 $n_p < n_t$
DNS	408 591.0	1 109 110.0	325 152.0
Mapping	40 859.1	110 911.0	325 152.0
CoMD DB	40 853.2	67 498.6	145 379.0
Kriging	40 454.2	45 322.1	118 231.0
Kr. DB	40 256.1	46 221.6	118 908.0

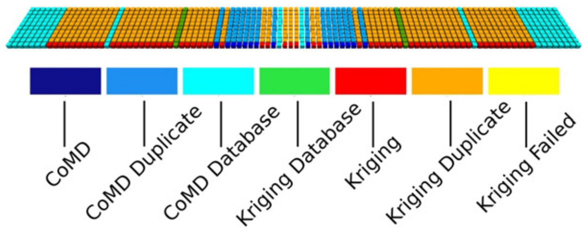

Fig. 7. The distribution of different tasks for the flat wave example of Fig. 5. Obviously the mapping of similar points (duplicates) leads to a reduction of the workload by 90%. Due to the evolved state of the wave, which almost reached the periodic boundaries in the x-direction, the amount of plain database values is reduced. While the access of plain database values is reduced, the amount of successful kriging database access is already notable at the front of the traveling waves. Besides the direct access of the database values, the values are used to execute the kriging, which leads to a reduction of almost 50% in this example. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5.1. Test problem 1 (flat wave)

In this example an initial deformation of 4% in the x-direction is applied to 10% of cells in the center of the simulation area (e.g., a grid of 100 × 10 cells per volumes in the present example). The initial condition has to be applied on the macro scale, since our adaptive sampling relies on the conserved quantities in the first place. In order to present the results of our simulations, we first take a look at the dynamic workload and the reduction of the actual number of executed CoMD instances per time-step. Fig. 7 illustrates the different tasks executed to provide the necessary data for each finite volume. The snapshot was taken after a few integration steps. In the regions with high strain, many CoMD tasks (dark-blue) need to be executed, whereas kriging (red) can be successfully applied to the front of the wave. In the so-far unaffected area, the database can be directly used (turquoise).

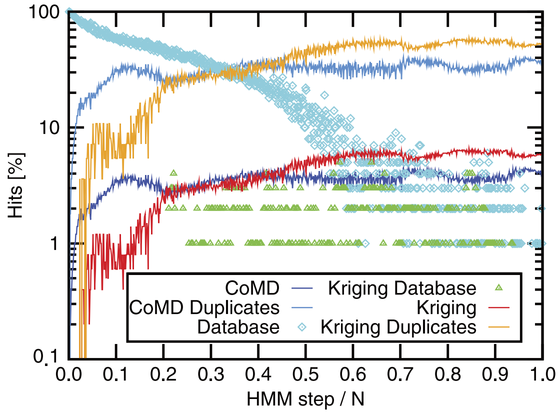

Fig. 8. Percentage of total tasks per quarter step for TP1. Turquoise diamonds show the number of CoMD database tasks and green triangles represent the number of kriging database tasks. The dark-blue line shows the number of CoMD tasks, while the light-blue line shows the number CoMD duplicates. The red line shows the number of kriging tasks, while the orange line shows the number of kriging duplicates. For the measurement we used a database threshold of 10^{-5} and a kriging error threshold of 10^{-3}. After an increase of the CoMD tasks at the beginning, due to the evolving of the wave through the area, the number of CoMD reaches a stable value at around 4% of the total number of tasks. All the other 96% of the tasks are either evaluated with the help of kriging (6%), the databases or are duplicates. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

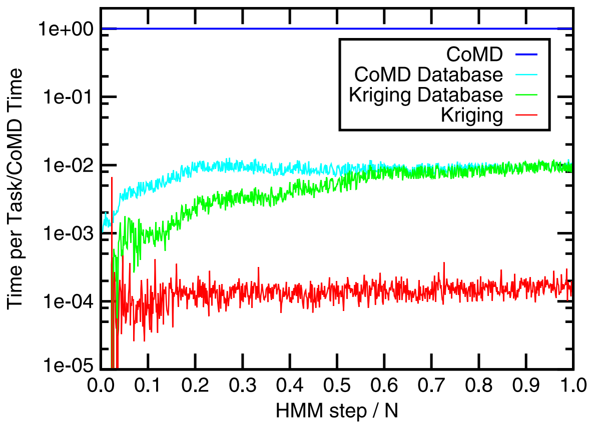

Fig. 9. Duration of different tasks per quarter step normalized by the CoMD execution time for TP1. For every hit type the time was measured, showing the relative duration independent of the available parallelization. Obviously, the CoMD tasks take way longer compared to the almost negligible duration of the other tasks. The CoMD task includes the time storing the value in the database, too, despite this time is negligible. The database task contains the total time it takes to check, sort and get the desired bucket/value for the values within the threshold for the *entire* macro-solver grid. The same is true for the kriging database task. The kriging task includes sorting and getting a bucket from the database, executing the kriging and if successful storing it in the kriging database. A database threshold of 10^{-5} and a kriging threshold of 10^{-3} have been used.

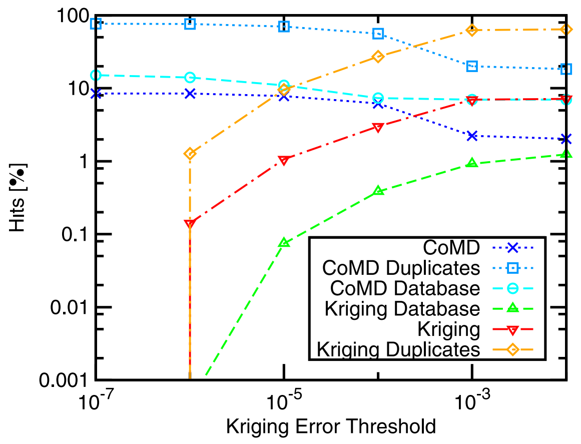

Fig. 10. Percentage of tasks per total simulation as dependence on error threshold for the kriging estimation for TP1. The figure shows the percentage of hits by the different tasks over the kriging error threshold. The database threshold is kept constant with 10^{-7}. Apparently, for the lowest threshold the number of kriging hits drops to zero, because the database catches all possible kriging tasks in advance. As expected with increasing kriging threshold the total number of successful kriging tasks rises.

Fig. 8 shows the percentage of each of the different tasks in a solution of the system over a long simulation run (0.5 µs) that results in waves traveling several times throughout the system. However, due to the finite volumes so called numerical friction leads to a decreasing strength of the strain over time. Hence, after the very first step with only two CoMD hits (due to initial condition, resulting in 0.2% of the total number of possible hits), the database can be used to access exact values for the unaffected nodes. As the wave evolves over time the number of database values that can be directly reused decreases and the number of necessary CoMD hits increases. This is offset by an increase in the number of duplicate hits our method is able to detect and eliminate as well as an increase in the number of opportunities for kriging (and the detection of duplicate kriging hits).

In addition to the use of the CoMD database as a plain lookup table, we introduced a database error threshold, which allows for the use of results gained by input values that are within the threshold. Hence, the CoMD database hit rate depends on that database threshold. For this specific test problem the number of CoMD and kriging tasks reach an almost constant rate for the long time limit. Apparently, only a small number of tasks are using the kriging database, where no threshold is allowed. Restricting the kriging database to exact results only is due to the already present error of the predicted values. Allowing for values with an additional error as input for the prediction would result in an uncontrolled error exposure throughout the system.

Fig. 9 shows the time required by each task during the simulation. We focus on the time in terms of tasks, as we dynamically schedule tasks so as to support a wide range of host system configurations with a varying number of available CPU cores. E.g. a CoMD task takes about 20 s on an AMD Opteron Processor 6168 core, which is substantially longer than any other tasks. Since the actual execution time is strongly affected by the underlying hardware, we normalize our measurements by the CoMD execution time. The database task, which includes searching the bucket and sorting the result for the *entire* field, takes, at most, about 0.007%. While the kriging database tasks become increasingly slow as the database gets populated, we measure it to take, at most, 0.01%. To avoid excessive database access for the kriging, we restrict the input to the *ten* nearest neighbors. With the latter, kriging itself is, again, much faster, despite the need to both read from and write to the database. This is due to the non-blocking nature of writes with the Redis database.

Next, we investigate the influence of certain thresholds on the actual number of different tasks during the example simulation. We set up two different thresholds: one for the acceptance of the kriging result and one for the acceptance of the database value. Fig. 10 shows the number of task hits as a percentage of the total simulation over the size of the kriging error threshold. As the error threshold for the kriging increases, the number of successful kriging tasks also increases, while the number of CoMD tasks and database tasks decreases. In the first place mainly database hits are reduce/replaced by the kriging task, but for larger thresholds a significant number of CoMD tasks are replaced, too. For this particular test problem a kriging threshold larger than 10^{-4} leads to more kriging tasks than CoMD tasks, which results in a massive speedup.

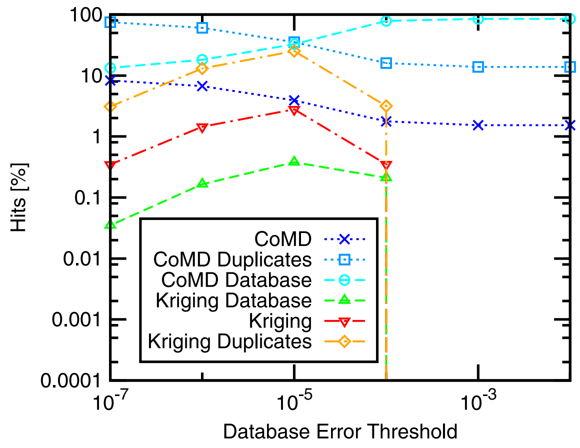

Fig. 11. Percentage of tasks per total simulation as dependence on error threshold of the database results for TP1. The plot shows the percentage of hits with a kriging error threshold, which is always one order of magnitude bigger than the different database thresholds, in order to have at least some points successfully evaluated by the kriging. This leads to an increasing number of successful database hits as well as kriging for low thresholds. For an error threshold of 10^{-4} so many database hits are successful, that only a minor number of tasks is left to be either evaluated by CoMD or the kriging. Apparently, for thresholds larger than that all hits are either database or CoMD type.

Fig. 11 shows the execution of different tasks as a function of the database error threshold. As the database error threshold increases, the plain database hits are increasing. Due to the fact that we have to increase the kriging error threshold at the same time, to ensure at least some successful kriging tasks, the total number of kriging tasks rises for small thresholds, too. However, for a threshold of 10^{-4} the number of successful database hits is so large that the number of kriging tasks is decreasing, while for error threshold beyond that every tasks is either a database hit or a CoMD task.

In summary, increasing error thresholds leads to a reduction in the number of expensive CoMD hits and therefore reduces the execution time significantly. Due to our specific implementation, the database always gets checked in the first place, reducing the number of possible kriging tasks (e.g. especially in case of using the same error threshold for the database and the kriging). In turn this gain in performance must be paid with a reduction in the accuracy, which appeared to be always at least in the order of the thresholds.

Besides the performance of the model, we have to be careful about the stability of the simulation. In order to check the effects of the different thresholds and statistical errors, we applied Gaussian noise onto the CoMD as well as the database and kriging results. Our method seems to be extremely stable and is able to provide reliable results, even with Gaussian noise about the order of the values of the fluxes. However, at some point the kriging prediction breaks down which results in an increase in the number of CoMD tasks.

5.2. Test problem 2 (circular impact)

The flat wave example discussed above is appropriate for investigating the mapping of the duplicated CoMD/kriging hits and the basic behavior of the prediction via kriging. But for simulations of impacts like the laser application on a copper plate it suffers from simplicity. In this section, we will apply D^2KAS on the most difficult setup: A circular “impact” in the center of the plate affecting all components of the strain tensor and interference due to the periodic boundaries. We start with a setup similar to the flat wave example, but in a region with 50×50 cells ($\approx$10,000 nm^2) and a deformed area in the center with radius $r = 5.425$ nm. Again, we induce a deformation of 4% in x-direction but this time in y-direction, too. In order to gain the maximum complexity we also introduce

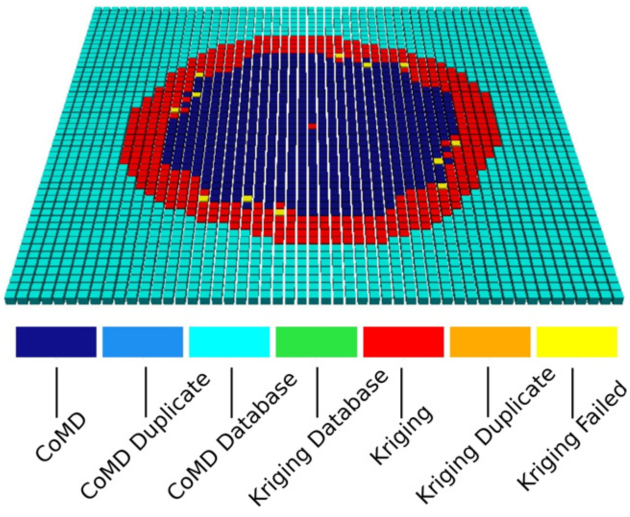

Fig. 12. The different tasks for the circular example are illustrated after a small number of time steps. Obviously, there are no duplicates (compare TP1) and therefore all the points in the center have to be evaluated by single CoMD runs. In this example a rather high database error threshold of 10^{-4} was used, leading to the observed circular area of kriging duplicates surrounding the CoMD hit area. Without the database threshold a squared area of kriging hits would surround the CoMD hits (based on the rectangular grid). In the area with a small gradient in the strain the kriging can reduce the number of CoMD hits significantly.

an initial extension of 2% onto the xy- and yx-components of the strain tensor. The resulting simulation is illustrated in a times series of snapshots shown in Fig. 6.

After the initial deformation the wave travels throughout the system. During the simulation the number of CoMD hits as well as the number of successful kriging tasks are rising, while the number of plain database hits is decreasing. Fig. 12 shows a snapshot of the applied tasks. The circular task distribution is a result of the combination of all three thresholds. As the center has to be computed by CoMD, the border region can be successfully evaluated with the help of kriging and database values. The circular shape is a direct result of the database threshold because of the underlying scheme; a quadratic/rectangular area. Without a database threshold (threshold $= 0$) kriging tasks are applied in this area. But with the database threshold taking away the tasks with the smallest deviations, the resulting circular shaped area appears.

Fig. 13 shows the percentage of successful hits for our second example system. We observed that the number of necessary CoMD tasks is rising to almost 90% in the end. This is due to the complexity of the setup leading to an interference of waves throughout the periodic boundaries. In the first part of the simulation the number of CoMD hits is continuously rising as well as the number of kriging hits, while the number of plain database values is decreasing, as in the flat wave example. But since there are no duplicated hits the actual number of hits is much larger than in the flat wave example. In Fig. 13 one recognizes a dip in the number of kriging hits around 0.4, which refers to the wave reaching the periodic boundaries. After this increasing complexity, the number of kriging hits drop to a fairly small amount of about 20%.

This worst case scenario shows the limit of the method, where either more advanced kriging methods or additional adaptive sampling methods need to be applied. However, the basic question is how realistic this worst case scenario is with respect to any production run. Since, we are interested in the simulation of comparable large metal plates, where waves of different complexity can travel through leading to small regions of high strain and large almost unaffected regions. Under such circumstances, the largely unaffected regions will consist of a number of database accesses and detected duplicate hits comparable to the quasi-1D wave. Therefore, this methodology is sufficient to reduce the number of CoMD hits, which can be handled by state-of-the-art supercomputers.

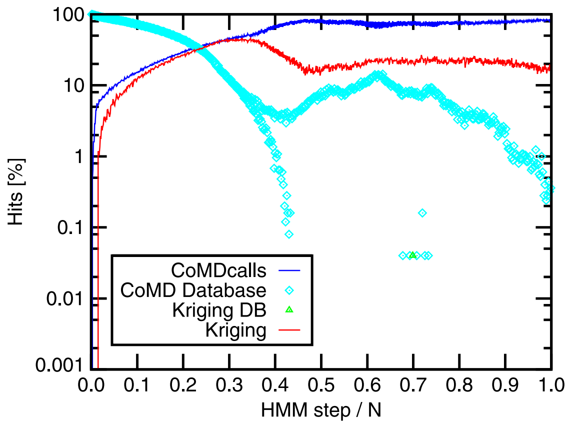

Fig. 13. Percentage of tasks per quarter step as trend over the total simulation run (see Fig. 6 for setup). Red diamonds show the number of database tasks and black triangles are the number of kriging database tasks (rare in this case). The blue line is the number of CoMD tasks and the green line is number of kriging tasks. We used again a database threshold of 10^{-5} and a kriging error threshold of 10^{-3}. The number of CoMD tasks as well as the number of kriging tasks is quadratic as expected and shows a remarkable kink at time 0.4 when the wave reaches the periodic boundaries. Due to the strong interference of the wave at the periodic boundaries the number of necessary CoMD hits rise to almost 90% and the remaining tasks are mainly evaluated with the help of kriging. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6. Conclusion

The presented scheme combines a prediction method with a heterogeneous multi-scale model for stochastic data (common to finite temperature processes) in order to introduce a new adaptive sampling scheme. The scheme is supported by a distributed cloud database utilized with key-value storage. The performance of the implementation in a parallel high-performance framework (Charm++) has been investigated for two different test problems, illustrating that a huge number of CoMD tasks can be saved without affecting the stability of the method. The framework efficiently distributed the independent MD and prediction tasks on state-of-the-art compute clusters. Furthermore, additional features like the mapping of similar tasks and the use of a different key for the kriging results have been discussed.

Our Distributed Database Kriging for Adaptive Sampling method (Charm++ version) showed that if the maximum number of tasks is larger than the number of compute nodes/cores – which is usually the case – a speedup of a factor of 2.5–25 can be achieved even for our small test problems. Our adaptive sampling method with the distributed cloud database may provide the base for further enhancements of the physics including micro/mesoscopic details like defects, fractures as well as crystal domains. It also shows the necessity of dynamic runtime systems and schedulers that can deal with the load imbalance arising from the varying number of tasks in every time step. The distribution of the tasks by these frameworks cause high traffic, which can only be handled by the distributed database support. Using a (exchangeable) cloud database in a parallel high-performance framework and the incorporation of next generation techniques illustrate the versatility of our method [41].

Acknowledgments

We thank E. Venmugil, A. Y. Rivera and E. Cieren for fruitful discussions at the beginning of this project as well as J. Mohd-Yusof for support with the CoMD code. We further thank Phil Miller for his support with the Charm++ version of the code. This work was supported by the Los Alamos Information Science & Technology Center (IS&T) Co-Design Summer School, the US Department of Energy (DOE), Office of Advanced Scientific Computing Research (ASCR) through the Exascale Co-Design Center for Materials in Extreme Environments (ExMatEx), and the Center for Nonlinear Studies (CNLS). Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by Los Alamos National Security, LLC, for the National Nuclear Security Administration of the US Department of Energy under contract DE-AC52-06NA25396. Simulations were performed on the CCS-7 cluster Darwin. Assigned: LA-UR-14-27213.

References

[1] D. Givon, R. Kupferman, A. Stuart, Extracting macroscopic dynamics: model problems and algorithms, Nonlinearity 17 (6) (2004) R55.
[2] P. Español, Statistical mechanics of coarse-graining, in: M. Karttunen, A. Lukkarinen, I. Vattulainen (Eds.), Novel Methods in Soft Matter Simulations, in: Lecture Notes in Physics, vol. 640, Springer, Berlin, Germany, 2004, pp. 69–115.
[3] S.O. Nielsen, P.B. Moore, B. Ensing, Adaptive multiscale molecular dynamics of macromolecular fluids, Phys. Rev. Lett. 105 (2010) 237802.
[4] Y. Chen, J. Zimmerman, A. Krivtsov, D.L. McDowell, Assessment of atomistic coarse-graining methods, Internat. J. Engrg. Sci. 49 (12) (2011) 1337–1349.
[5] V. Rühle, C. Junghans, Hybrid approaches to coarse-graining using the votca package: liquid hexane, Macromol. Theory Simul. 20 (7) (2011) 472–477.
[6] D. Terzopoulos, M. Vasilescu, Sampling and reconstruction with adaptive meshes, in: Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on, IEEE, 1991, pp. 70–75.
[7] J.F. Thompson, Z.U. Warsi, C.W. Mastin, Numerical Grid Generation: Foundations and Applications, Vol. 45, North-Holland, Amsterdam, 1985.
[8] R.J. Lipton, J.F. Naughton, D.A. Schneider, Practical Selectivity Estimation Through Adaptive Sampling, Vol. 19, ACM, 1990.
[9] K. De Jong, Learning with genetic algorithms: an overview, Mach. Learn. 3 (2–3) (1988) 121–138.
[10] S.F. Smith, A learning system based on genetic adaptive algorithms, University of Pittsburgh, 1980.
[11] E. Weinan, B. Engquist, Z. Huang, Heterogeneous multiscale method: a general methodology for multiscale modeling, Phys. Rev. B 67 (9) (2003) 092101.
[12] X. Li, et al., Multiscale modeling of the dynamics of solids at finite temperature, J. Mech. Phys. Solids 53 (7) (2005) 1650–1685.
[13] E. Weinan, B. Engquist, X. Li, W. Ren, E. Vanden-Eijnden, Heterogeneous multiscale methods: a review, Commun. Comput. Phys. 2 (3) (2007) 367–450.
[14] I.G. Kevrekidis, C.W. Gear, J.M. Hyman, P.G. Kevrekidid, O. Runborg, C. Theodoropoulos, et al., Equation-free, coarse-grained multiscale computation: enabling mocroscopic simulators to perform system-level analysis, Commun. Math. Sci. 1 (4) (2003) 715–762.
[15] G.-S. Jiang, E. Tadmor, Nonoscillatory central schemes for multidimensional hyperbolic conservation laws, SIAM J. Sci. Comput. 19 (6) (1998) 1892–1917.
[16] G.-S. Jiang, D. Levy, C.-T. Lin, S. Osher, E. Tadmor, High-resolution nonoscillatory central schemes with nonstaggered grids for hyperbolic conservation laws, SIAM J. Numer. Anal. 35 (6) (1998) 2147–2168.
[17] A. Kurganov, E. Tadmor, New high-resolution central schemes for nonlinear conservation laws and convection–diffusion equations, J. Comput. Phys. 160 (1) (2000) 241–282.
[18] J. Knap, N. Barton, R. Hornung, A. Arsenlis, R. Becker, D. Jefferson, Adaptive sampling in hierarchical simulation, Internat. J. Numer. Methods Engrg. 76 (4) (2008) 572–600.
[19] N.R. Barton, J. Knap, A. Arsenlis, R. Becker, R.D. Hornung, D.R. Jefferson, Embedded polycrystal plasticity and adaptive sampling, Int. J. Plast. 24 (2) (2008) 242–266.
[20] B. Rouet-Leduc, K. Barros, E. Cieren, V. Elango, C. Junghans, T. Lookman, J. Mohd-Yusof, R.S. Pavel, A.Y. Rivera, D. Roehm, et al., Spatial adaptive sampling in multiscale simulation, Comput. Phys. Comm. 185 (7) (2014) 1857–1864.
[21] M. Kanevski, R. Parkin, A. Pozdnukhov, V. Timonin, M. Maignan, V. Demyanov, S. Canu, Environmental data mining and modeling based on machine learning algorithms and geostatistics, Environ. Modell. Softw. 19 (9) (2004) 845–855.
[22] M.J. Mills, P.L. Popelier, Intramolecular polarisable multipolar electrostatics from the machine learning method kriging, Comput. Theor. Chem. 975 (1) (2011) 42–51.
[23] C.M. Handley, G.I. Hawe, D.B. Kell, P.L. Popelier, Optimal construction of a fast and accurate polarisable water potential based on multipole moments trained by machine learning, Phys. Chem. Chem. Phys. 11 (30) (2009) 6365–6376.
[24] P.N.S. Sanfillippo, Redis database, 2014. URL: http://redis.io.
[25] L.V. Kale, S. Krishnan, CHARM++: A Portable Concurrent Object Oriented System Based on C++, Vol. 28, ACM, 1993.
[26] K. Knobe, Ease of use with concurrent collections (CnC), in: Proceedings of the First USENIX Conference on Hot Topics in Parallelism, USENIX Association, Berkeley, CA, USA, 2009, pp. 17–22.
[27] J. Mohd-Yusof, CoMD: classical molecular dynamics proxy application, version 1.1, 2013. URL: https://github.com/exmatex/CoMD.

[28] J. Irving, J.G. Kirkwood, The statistical mechanical theory of transport processes. IV. The equations of hydrodynamics, J. Chem. Phys. 18 (1950) 817.
[29] A. Harten, The artificial compression method for computation of shocks and contact discontinuities. III. Self-adjusting hybrid schemes, Math. Comp. 32 (142) (1978) 363–389.
[30] P.K. Sweby, High resolution schemes using flux limiters for hyperbolic conservation laws, SIAM J. Numer. Anal. 21 (5) (1984) 995–1011.
[31] N. Cressie, Spatial prediction and ordinary kriging, Math. Geol. 20 (4) (1988) 405–421.
[32] N. Cressie, The origins of kriging, Math. Geol. 22 (3) (1990) 239–252.
[33] N. Cressie, Statistics for Spatial Data, John Wiley and Sons, Inc., 1993.
[34] P. Indyk, R. Motwani, Approximate nearest neighbors: towards removing the curse of dimensionality, in: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, STOC'98, ACM, New York, NY, USA, 1998, pp. 604–613.
[35] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing, in: VLDB, Vol. 99, 1999, pp. 518–529.
[36] J. Han, E. Haihong, G. Le, J. Du, Survey on NoSQL database, in: Pervasive Computing and Applications (ICPCA), 2011 6th International Conference on, IEEE, 2011, pp. 363–366.
[37] M. Stonebraker, SQL databases v. NoSQL databases, Commun. ACM 53 (4) (2010) 10–11.
[38] R. Cattell, Scalable SQL and NoSQL data stores, ACM SIGMOD Rec. 39 (4) (2011) 12–27.
[39] P.N.S. Sanfilippo, Hiredis interface, 2014. URL: https://github.com/redis/hiredis.
[40] L. Dagum, R. Menon, OpenMP: an industry standard API for shared-memory programming, Comput. Sci. Eng. 5 (1) (1998) 46–55. IEEE.
[41] R. Pavel, D. Roehm, C. Mitchell, C. Junghans, K. Barros, J. Mohd-Yusof, T. Germann, A. McPherson, Cloud+X: a Service-Based HPC Software Stack, 2014.
[42] Y. Mishin, M. Mehl, D. Papaconstantopoulos, A. Voter, J. Kress, Structural stability and lattice defects in copper: Ab initio, tight-binding, and embedded-atom calculations, Phys. Rev. B 63 (22) (2001) 224106.