Computer Physics Communications 204 (2016) 64–73

Contents lists available at ScienceDirect

Computer Physics Communications

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

Efficient parallelization of analytic bond-order potentials for large-scale atomistic simulations

C. Teijeiro [a,*], T. Hammerschmidt [a], R. Drautz [a], G. Sutmann [a,b]

[a] Interdisciplinary Centre for Advanced Materials Simulation (ICAMS), Ruhr-University Bochum, D-44801 Bochum, Germany

[b] Jülich Supercomputing Centre (JSC), Institute for Advanced Simulation (IAS), Forschungszentrum Jülich, D-52425 Jülich, Germany

ARTICLE INFO

Article history:
Received 30 December 2015
Received in revised form
17 March 2016
Accepted 19 March 2016
Available online 31 March 2016

Keywords:
Bond-order potentials
Large-scale simulations
Parallelization
MPI

ABSTRACT

Analytic bond-order potentials (BOPs) provide a way to compute atomistic properties with controllable accuracy. For large-scale computations of heterogeneous compounds at the atomistic level, both the computational efficiency and memory demand of BOP implementations have to be optimized. Since the evaluation of BOPs is a local operation within a finite environment, the parallelization concepts known from short-range interacting particle simulations can be applied to improve the performance of these simulations. In this work, several efficient parallelization methods for BOPs that use three-dimensional domain decomposition schemes are described. The schemes are implemented into the bond-order potential code BOPfox, and their performance is measured in a series of benchmarks. Systems of up to several millions of atoms are simulated on a high performance computing system, and parallel scaling is demonstrated for up to thousands of processors.

1. Introduction

Atomistic simulations have been proven successful in describing soft and hard materials to explain or predict their behavior under given thermodynamic conditions. These properties are thereby fully determined by the interatomic interactions and the dynamical behavior of a system can be accessed by exploring the system's phase space. In the simplest case, approximations based on pair interactions between particles are applied, and they can be linearly superposed to obtain the total force acting on a particle. However, the high computational complexity of $\mathcal{O}(N^3)$ with increasing number of degrees of freedom N limits its applicability to systems of $N \approx 1000$ atoms, while linear scaling $\mathcal{O}(N)$ implementations of density functional theory (DFT) start to become more efficient than the $\mathcal{O}(N^3)$ implementations also at about 1000 atoms. Therefore, if collective effects such as structural transitions, defect formation or non-equilibrium processes are to be studied on longer time and length scales, accurate approximations to describe the features of interest have to be introduced.

Analytic bond-order potentials are based on a coarse graining of DFT and compute atomic energies and forces by a moments expansion of length m of the local density of states, $n_{i\alpha}$ [1–3]. Thus, these potentials allow to compute atomistic interactions with high accuracy at a computational complexity of $\mathcal{O}(N)$. In the BOP formalism, the total binding energy of the system, U_B, is given as $U_B = U_{bond} + U_{prom} + U_{rep}$, where the promotion energy U_{prom} can be calculated as sums over occupied electronic states and the repulsive energy U_{rep} is represented as a classical pair or many-body interaction. The calculation of the bond energy takes most of the computational cost, and is given by

$$U_{bond} = 2 \sum_{i\alpha} \int_{-\infty}^{E_F} dE\,(E - E_{i\alpha})\, n_{i\alpha}(E) \tag{1}$$

where the local density of states $n_{i\alpha}$ is represented as a moments expansion, with the moments

$$\mu_{i\alpha}^N = \int E^N n_{i\alpha}(E) dE = \sum_{j\beta k\gamma \ldots q\kappa} \langle i\alpha|\hat{H}_{i\alpha j\beta}|j\beta\rangle\langle j\beta|\hat{H}_{j\beta k\gamma}|k\gamma\rangle \ldots \langle q\kappa|\hat{H}_{q\kappa i\alpha}|i\alpha\rangle. \tag{2}$$

In Eq. (1) E_F represents the Fermi energy and $E_{i\alpha}$ the energy of the occupied orbital α of atom i. The density of states $n_{i\alpha}$ is constructed by sampling a neighborhood of each atom i, which is accomplished by building self-returning paths as a chain of links between nearest neighbors (e.g. $\langle i\alpha|\hat{H}_{i\alpha j\beta}|j\beta\rangle$). The final

* Corresponding author.
E-mail addresses: carlos.teijeirobarjas@rub.de (C. Teijeiro), thomas.hammerschmidt@rub.de (T. Hammerschmidt), ralf.drautz@rub.de (R. Drautz), g.sutmann@fz-juelich.de (G. Sutmann).

http://dx.doi.org/10.1016/j.cpc.2016.03.008
0010-4655/

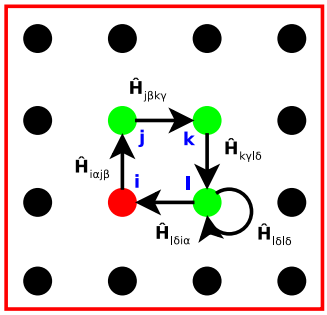

Fig. 1. Sample path of length 5 (contribution to the 5th moment for atom i and orbital α).

result is obtained by summing up the contributions of all path combinations from atom i of given length. The length of paths is determined by the order of the moment expansion, m, where the accuracy converges to the tight binding result with increasing number of moments. A sample path from an atom i is depicted in Fig. 1. For a more detailed description of bond-order potentials, see e.g. Refs. [4,5].

In order to exploit the locality of the energy calculation, it is natural to transfer well established concepts from the parallelization of short range interacting systems [6]. Efficient implementations for simple interaction models have proven their applicability to system sizes of billions of atoms [7,8] and beyond [9]. For simulations with local interactions, the domain decomposition method has become a standard approach. It consists in the distribution of both particles and space onto the processors, which allows to increase the overall size of the system since (1) the workload is shared between the processors and (2) the size of memory is shared, both of which are limiting factors in single core computations.

In the present article we propose different approaches based on domain decomposition for the parallelization of the bond-order potential code BOPfox [10]. The paper is organized in the following parts: Section 2 presents a general motivation and basic information about the sequential simulation code. Section 3 describes and compares the implementation of several parallel algorithms. Section 4 presents the performance evaluation of the implementations in terms of benchmarks for different number of threads, moment expansions and problem sizes. Finally, Section 5 gives the main conclusions of this work.

2. Motivation and code description

The BOPs provide a rigorous expansion for the density of states and the binding energy. The algorithmic problem is the construction of all closed paths of a given length (cf. Eq. (1) and Fig. 1) between neighboring atoms. As shown in Ref. [11], the complexity is confirmed to be $\mathcal{O}(N)$ when increasing the number of atoms N, but the prefactor per atom increases as $m^{9/2}$ with the moment expansion m. This strong increase of the prefactor for energy and force computations poses a significant computational load for multi-million atom systems, which are necessary, e.g., to describe the development of dislocation networks. Therefore an efficient parallelization is necessary to reduce the overall wall-clock time of large-scale simulations.

An initial analysis of the simulation code BOPfox reveals that the large number of path combinations that need to be evaluated for every atom clearly dominates the CPU time compared with the rest of tasks. Each path is computed as the result of a chain of multiplications of small pairwise Hamiltonian matrices between neighbor atoms (cf. Fig. 1), and all together contribute to the computation of energies and forces in the system. This task is accomplished in three steps for a given moment expansion m:

- Paths of length 1 from every atom are directly obtained from the bond information (i.e. pairwise Hamiltonians) associated with its nearest neighbors, including itself.
- Paths of length between 2 and $\lceil m/2 \rceil$ (from now on referred to as *half-length paths*) are computed.
- Paths of length between $\lceil m/2 \rceil + 1$ and m are obtained by merging (i.e. multiplying) pairs of the paths computed previously.

A rough profiling of the BOPfox code indicates that these multiplications consume more than 90% of the total execution time of a sample execution [11]. Therefore, the computation of paths is the target for the parallelization presented in this work.

3. Domain decomposition algorithms

A parallelization based on the concept of independent domains implies a tradeoff between two factors: (1) the use of local computations, which imply the definition of an overlap area with a certain level of redundancy, and (2) the use of communications, in order to transfer partial computations of paths from one domain to the other. Communications are required when a given domain does not have all necessary information to compute all self-returning paths for local atoms, which is usually the case for atoms close to the domain boundary. It is therefore required to balance the cost associated with data transfer between domains and redundant computations within the overlap region.

An additional consequence of this tradeoff is that, when some extreme cases are considered, the nature of the simulation hampers a really efficient parallelization. Considering moment expansions of length m, each atom is accessing all atoms in a spherical volume of radius $\lceil m/2 \rceil \times R_{cut}$, which should be of the order of a domain size and considerably smaller than the system size for an efficient parallel execution. According to this, a minor assumption has been considered for the present implementations of domain decomposition, i.e. the computation of paths requires information from only the local domain $\mathcal{D}$ and its neighbor domains, with which $\mathcal{D}$ shares a common border. The idea is to restrict the local-area information in every process to its associated domain and a relatively small portion of its neighbor domains, so that the number of domains in which a system is subdivided is coherent with the system size.

Previous simulations of pair potentials or BOPs with low-order expansions (available in widely used simulation codes like LAMMPS [12,13]) can benefit from highly efficient parallel approaches such as the neutral-territory [14] and the eighth-shell approach [15,16], which may also be used for analytic BOPs. However, here the intense computation in a volume around each atom suggests the definition of overlap zones according to the moment expansion m, in order to allow a mainly local computation of half-length paths for each domain. Consequently, the size of the domain is preferably defined large enough with respect to the overlap zone in order to keep communications as simple and reduced as possible.

Following these premises, three types of algorithms are now presented in order to illustrate different levels of the tradeoff between communications and computations. Starting from a scenario with redundant path computations with no additional communications, the tradeoff is applied to get an approach with no redundancy and complementary communications. According to this, the order of presentation of the algorithms is as follows: (1) minimum redundancy (*MIN-REDUND*), (2) path communication algorithms (*FULL-SHELL* and *HALF-SHELL*) and (3) force communications (*BOND-COMMS*). Fig. 2 illustrates the differences between these approaches in terms of the tradeoff between local computations and the amount of required communications with the type of exchanged information (either path matrices or force contributions). All these algorithms have been implemented using the MPI library, which is a de-facto standard approach for parallel computing on distributed systems.

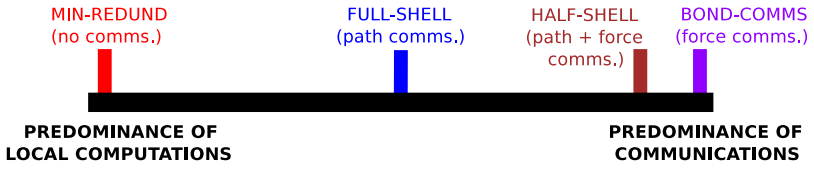

Fig. 2. Summary of parallelization approaches in terms of the tradeoff between computations and communications.

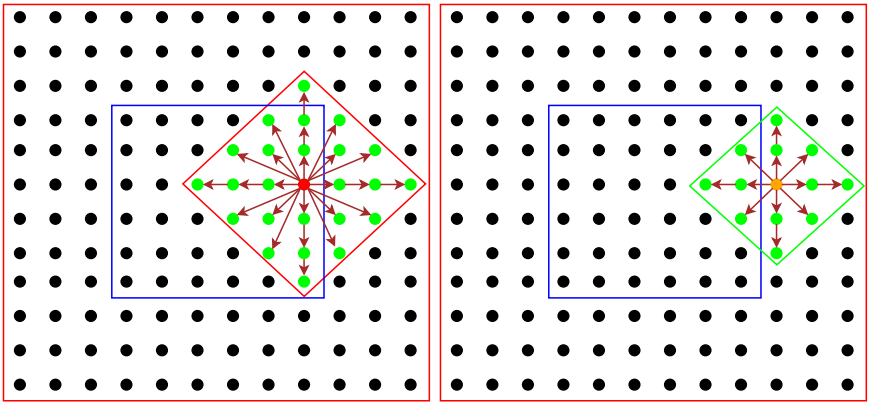

Fig. 3. Set of half-length paths from a domain atom (left) and overlap atom (right) used for the merge procedure with $m = 6$. The blue square encloses the domain atoms, and the overlap zone is delimited by the red square. The endpoints of the paths from the overlap atom must also be endpoints of the domain atom: longer paths from the overlap atom are not relevant for energy and force computations using the target domain atom. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.1. Minimum redundancy algorithm

This algorithm focuses on the computation of paths based only on local information stored in the corresponding process (i.e. using domain data and overlap zones, without any communication during path computations) with a minimum number of multiplications. Considering a moment expansion m, the half-length paths are necessarily computed for every domain atom i, and then merged with the paths that start on any neighbor atom of the corresponding i. According to this, there are two necessary conditions that must be fulfilled for path computations in the overlap area:

- The half-length paths should be computed only for the overlap atoms j that have at least one nearest neighbor i in the domain.
- The length of the paths computed from atoms in the overlap area should be at most $m - h - 1$, where $h = \lceil m/2 \rceil$, in order to compute force contributions between a domain atom i and a neighbor atom j in the overlap zone.

Fig. 3 presents a scheme of a particle configuration for a moment expansion of $m = 6$, where the endpoints of the half-length path computations for two neighbor atoms, one in the domain (left) and another in the overlap zone (right), are delimited. In this example, the paths up to length 3 from the domain atom only need to be merged with the paths up to length 2 from the overlap area atom, so that force contributions up to length 5 (i.e. $m - 1$) are obtained and the self-returning path of length 6 is computed using an additional matrix multiplication.

In order to obtain paths of length between $\lceil m/2 \rceil + 1$ and $m - 1$, the following algorithm is performed for every atom i in the domain and every half-path length $p = 1 \ldots \lceil m/2 \rceil$ (see Fig. 4 for a graphical explanation with the pseudocode for the computation of interference paths, where reverse of a path is computed by transposing the path matrix):

- The endpoints k of length h are marked in an auxiliary array.
- For each neighbor j of atom i, every endpoint k' of paths of length $h' = m - h - 1$ is obtained: if one atom k' matches any of the marked values k, then the path $\mathcal{P}_{i,k}$ and the reverse of path $\mathcal{P}_{j,k}$ are multiplied.

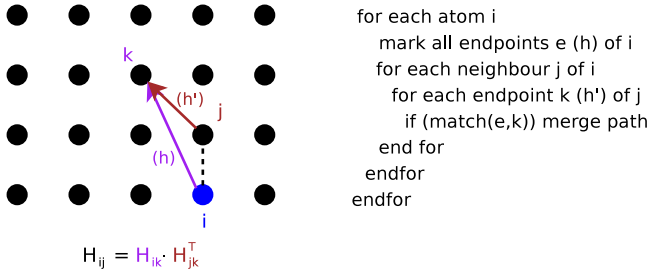

Fig. 4. Path merging for algorithm *MIN-REDUND* (sample computation for interference paths, where $H_{kj} = H^T_{jk}$).

It is important to note that this algorithm is applied regardless of the location of j: it can be either a domain or an overlap atom. Fig. 5 illustrates two merged paths from atom i to j, where in one case the merging endpoint is an atom in the domain (k_1) and in the other an atom in the overlap zone (k_2).

Fig. 6 illustrates the computation of half-length paths from the domain and the overlap atoms presented in Fig. 3, overlapping both sets of endpoints and also showing the situation from the point of view of the two neighbor domains (i.e. the overlap atom in the graph on the left is the domain atom in the graph of the right, and vice versa). The endpoints obtained from the atom in the overlap zone are in both cases a subset of the endpoints associated with the domain atom, and therefore all redundant paths are used for longer path computations at the merge stage. Moreover, as the endpoints of paths of length up to $m - 1$ in the overlap area are restricted to nearest neighbors of domain atoms, and considering that all half-length paths are built then merged on demand, it is consequently guaranteed that only the minimum necessary set of paths are computed.

3.2. Path communication algorithms

The basis of these algorithms is the use of communications of computed paths between neighbor domains in order to obtain the full path computation, so that redundant calculations in

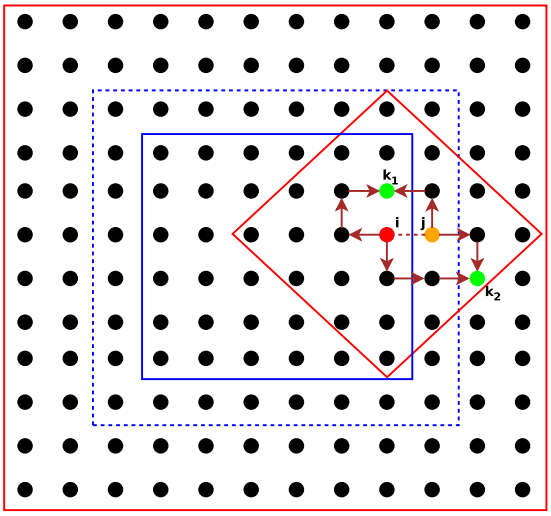

Fig. 5. Paths merged in domain and overlap endpoints (k_1 and k_2, respectively). The dotted square represents the overlap atoms that may have at least one nearest neighbor in the domain (e.g. atom j).

the overlap zone are avoided. Here the overlap atoms are used exclusively to obtain half-length paths that start in the atoms in the domain: no path computations are performed from any of them. In order to obtain the full paths, two different approaches are presented here: (1) the *FULL-SHELL* approach and (2) the *HALF-SHELL* approach.

The *FULL-SHELL* algorithm computes half-length paths and their associated force contributions locally for every domain atom, in the same way as the *MIN-REDUND* algorithm (i.e. according to the conditions explained in Section 3.1). Then, communications are performed in order to send the paths of length $l = (m + 1)/2 \ldots m - 1$ (where $m = \lceil m/2 \rceil$) from every domain atom that has a neighbor atom in a different domain, which will therefore receive these paths. The selected range of path lengths l is the minimum required to build full paths correctly at the merge stage without further communications. Fig. 7 illustrates the communications of this algorithm. Every domain sends the necessary paths to its corresponding neighbor, and also receives the necessary set of paths from every neighbor.

The *HALF-SHELL* algorithm performs path communications similarly to *FULL-SHELL* but only for half of the neighbor processors, whereas force communications are performed for the other half. More formally, as the overlap area around each domain is subdivided in regions associated with different neighbor domains, the algorithm arranges these regions in two equal sets: on one set, half-length paths are received from the neighbor domain in order to compute the full paths locally (as for *FULL-SHELL* and *MIN-REDUND*), whereas the other set will directly receive the final force contributions associated to the corresponding atoms. Consequently, for every domain border between neighbor domains d_1 and d_2 with overlapping zones ov_1 and ov_2 respectively, either ov_1 or ov_2 is considered to have path communications and local computations, and the other overlap zone must necessarily be interpreted as having remote computations and force communications.

Fig. 8 shows the scheme of these computations and communications for the same simplified system shown in Fig. 7: for *HALF-SHELL*, the leftmost graph also identifies domain atoms as target for path computations; then, the central graph presents the path communications that allow the force calculations associated with bonds in the overlap area, after computing half-length paths only for domain atoms; and finally the rightmost graph shows the communications of forces that have been previously computed in neighbor processes.

The differences in the communication sizes between *FULL-SHELL* and *HALF-SHELL* are illustrated in Figs. 9 and 10, which represent two different domains of a simplified 2D sample system for a moment expansion of 6. Each domain has its overlap zone and the nearest neighbor area marked in a dotted square, and a target atom is marked in red on each domain, alongside with the area of endpoints covered by half-length paths. Fig. 9 presents path communications on both sides, and all communications for the target atom present the same size. Nevertheless, Fig. 10 shows how the force communications are smaller with respect to path communications, because on the sender side they only involve those domain atoms whose nearest neighbor is an overlap atom (i.e. a domain atom in the receiving side).

3.3. Force communication algorithm

This algorithm exploits the communication of bond-based force contributions only, instead of transmitting path information. The overlap atoms here have the same role as in the *FULL-SHELL* and *HALF-SHELL* algorithms (Section 3.2), i.e. there is no local computation of paths that start on an overlap atom. Instead, the computed paths for each domain atom are constructed by merging, without receiving any other path information from neighbor domains. In order to implement this algorithm, there are two key differences with respect to the previous approaches:

- The merging point for all paths of length $p > \lceil m/2 \rceil$ is always an atom in the domain.
- The force contributions computed by merging paths can be associated with any pair of neighbor atoms in the domain or the overlap zone.

After the computation of half-length paths and their associated force contributions, the merging algorithm iterates on every domain atom i. For a given combination of path lengths sm and lm, every endpoint e of paths of length lm from atom i is marked, and then every nearest neighbor k of every endpoint j of length sm of i is explored. If there is a coincidence between any k and a marked endpoint e, then the path should be merged, and thus a force contribution between the corresponding atoms j and k is obtained. This procedure is illustrated in Fig. 11, with a graph and the pseudocode of this algorithm for the computation of interference paths (cf. Fig. 4 for differences with the previous parallelization approaches).

Finally, the communication routine must send partial force contributions associated with any atom for which a path has been computed in the local processor. This implies that any atom in the overlap area, with its associated bonds, can be the target of communications. Fig. 12 presents an example of the communications performed by the *BOND-COMMS* algorithm for the simplified case of two domains in a 2D system. Unlike in the previous Figs. 9 and 10, the dotted area that defines the atoms involved in communications covers the whole overlap zone, instead of only the nearest neighbors of domain atoms. In the illustrated example, both left and right domains are computing a path between two atoms located in their respective overlap areas. Consequently, the associated force contribution has to be communicated to the corresponding domain. Here it is important to note that a process needs to gather force contributions if at least one of the atoms of the target bond is defined as a domain atom, i.e. a force contribution may need to be sent to up to two different processes.

4. Performance benchmarks

4.1. Setup

The evaluation testbed used in this work is the JUROPA supercomputer at Jülich Supercomputing Centre (JSC), which

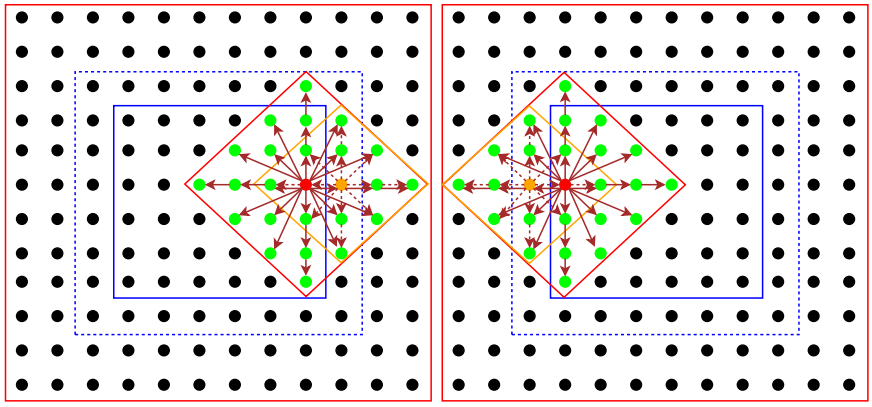

Fig. 6. Computations of algorithm *MIN-REDUND* on neighbor domains.

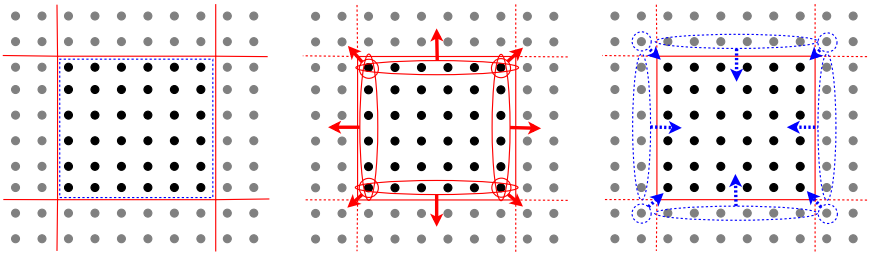

Fig. 7. Generic scheme of communications and computations for the algorithm *FULL-SHELL* for a domain in a simplified 2D system with overlap atoms from neighbor domains: every process sends/receives information on border atoms to/from all neighbors in a single step, after the computation of half-length paths (center and right graphs, respectively).

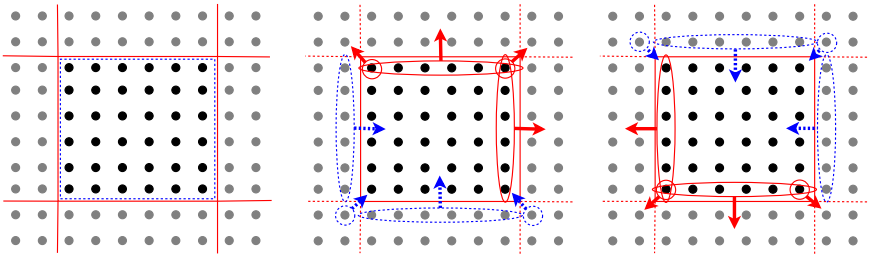

Fig. 8. Generic scheme of communications and computations for the algorithm *HALF-SHELL*: every processor sends/receives information on border atoms to/from half of the neighbor processors after the computation of half-length paths, and then communications are performed to send/receive the final forces to/from the other half of the neighbor processors.

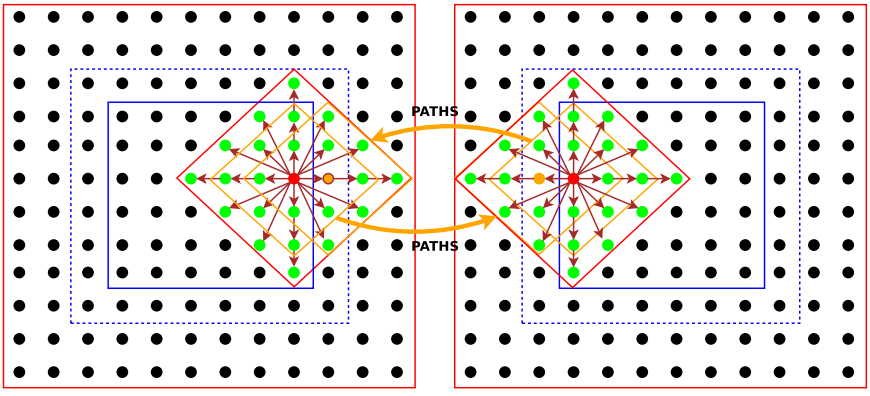

Fig. 9. Example of *FULL-SHELL* on neighbor domains: paths are exchanged between neighbor processes and force contributions are locally computed in parallel.

is a representative general-purpose supercomputer with hybrid shared/distributed memory architecture. It has 2208 compute nodes, with 2 Intel Xeon X5570 quad-core processors at 2.93 GHz (i.e. 8 cores per node), 24 GB of DDR3 memory at 1066 MHz and InfiniBand QDR HCA with non-blocking Fat Tree topology. The MPI compiler used for this evaluation is ParaStation MPI 5.0.29-1, which is the high performance MPI implementation that has shown to provide the best efficiency on JUROPA. All test codes have been compiled with the optimization flag -O3.

In order to test the efficiency of the different parallel algorithms, the simulated systems in this section are defined as 3D cubes (with periodic boundary conditions) composed of regular bcc crystals,

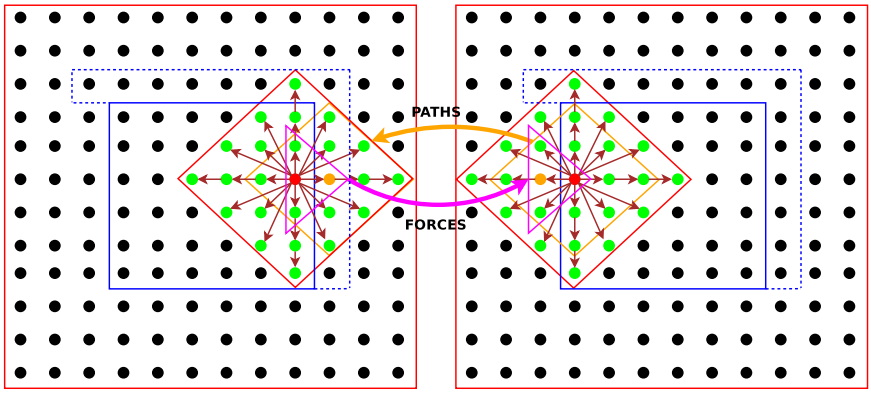

Fig. 10. Example of *HALF-SHELL* on neighbor domains: paths are sent for half of the neighbor processors, then force contributions are computed and sent back.

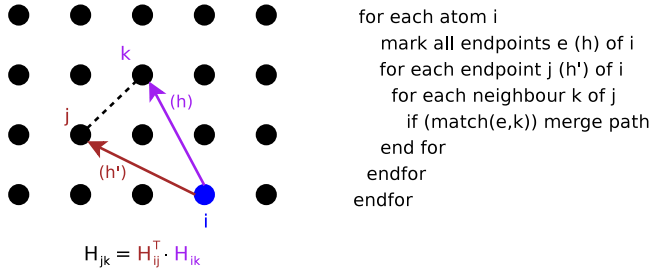

Fig. 11. Path merging for algorithm *BOND-COMMS* (sample computation for interference paths, where $H_{ji} = H_{ij}^{T}$).

Table 1
Total matrix multiplications (31,250 atoms, 6 moments, parallel executions with 8 processes).

Algorithm	# Matrix multiplications	
	Half paths	Merge paths
SEQUENTIAL	37,500,000	27,781,250
WORK-DISTR	37,500,000	27,781,250
DOM-DECOMP	113,632,144	84,251,152
MIN-REDUND	56,197,326	30,168,038
FULL-SHELL	37,500,000	30,168,038
HALF-SHELL	37,500,000	27,781,250
BOND-COMMS	37,500,000	27,781,250

and the MPI parallelization is accomplished using a 3D cartesian grid. The goal is to guarantee a regular structure and a homogeneous density of atoms for the whole system, which provides balanced data and work distribution for every process. In general, a minimum parallel grid of 8 processes (i.e. $2 \times 2 \times 2$) is considered for all parallel executions, where each process communicates in all cartesian directions with another neighbor process.

In addition to the algorithms presented in Section 3, two more basic algorithms are used for performance comparisons: (1) *WORK-DISTR* and (2) *DOM-DECOMP*. The *WORK-DISTR* approach represents the simple work distribution algorithm (see Fig. 13), and it is used as a model for parallel computations because it mimics the execution of the sequential implementation with perfect balancing, but replicating the whole system in each processor. On the other hand, the *DOM-DECOMP* algorithm (see Fig. 14) is used here as the most simple approach that allows good memory scalability, but with a large number of redundant computations that limits performance, as presented in Ref. [17].

4.2. Comparison of number of matrix multiplications

The main computational workload of the algorithm comes from the matrix multiplications used for path computations. Tables 1 and 2 present the number of these operations performed by all the algorithms described in this work, using a test simulation with 31,250 atoms up to the 6th moment on 8 processors. The original sequential algorithm (*SEQUENTIAL*) is used in Table 1 as a reference for the total number of operations required for the computation of both half-length paths and merged paths, as stated in the second and third columns.

The simple parallelization approach *DOM-DECOMP* is performing half-length path computations for all atoms in a given domain and the associated overlap zone. Therefore, redundant computations are present for all atoms $N_{overlap}$ in the overlap zone, which can be quantified asymptotically as follows, considering a cubic domain of length L with a homogeneous atom density ρ using a moment expansion m and a cut-off radius R_{cut}:

$$\begin{aligned} N_{overlap} = {} & 2\rho\left(L + 2\cdot\left\lceil\frac{m}{2}\right\rceil\cdot R_{cut}\right)^2\cdot\left(\left\lceil\frac{m}{2}\right\rceil\cdot R_{cut}\right) \\ & + 2\rho L\left(L + 2\cdot\left\lceil\frac{m}{2}\right\rceil\cdot R_{cut}\right)\cdot\left(\left\lceil\frac{m}{2}\right\rceil\cdot R_{cut}\right) \\ & + 2\rho L^2\cdot\left(\left\lceil\frac{m}{2}\right\rceil\cdot R_{cut}\right). \end{aligned} \tag{3}$$

In the case of *MIN-REDUND*, the redundancy area is restricted to the nearest neighbors of atoms in the domain, and therefore reduced to an expression that does not depend on the moment expansion m:

$$\begin{aligned} N_{overlap} = {} & 2\rho(L + 2R_{cut})^2\cdot R_{cut} + 2\rho L(L + 2R_{cut})\cdot R_{cut} \\ & + 2\rho L^2\cdot R_{cut}. \end{aligned} \tag{4}$$

As a result of the previous conditions, the algorithms presented in Section 3 show significant computational improvements with respect to *DOM-DECOMP*. The *MIN-REDUND* approach presents around 18.6×10^6 redundant matrix multiplications with respect to *SEQUENTIAL* because of the construction of half-length paths in the overlap area, as well as additional 2.4×10^6 operations to merge them, which represent the cost of having local computations. The algorithm *FULL-SHELL* builds half-length paths without redundancy, but after performing communications to exchange them, it presents the same redundancy as *MIN-REDUND* when building merged paths. This situation occurs because the full set of half-length paths is generated and exchanged with neighbor domains in order to perform a local computation of force contributions, and this is only possible by merging the same paths twice for every bond between atoms in different domains. Nevertheless, this fact is solved by the *HALF-SHELL* approach, which replicates the perfect memory distribution of the *WORK-DISTR* approach while offering memory scalability, even though communications of half-length paths and force contributions are needed.

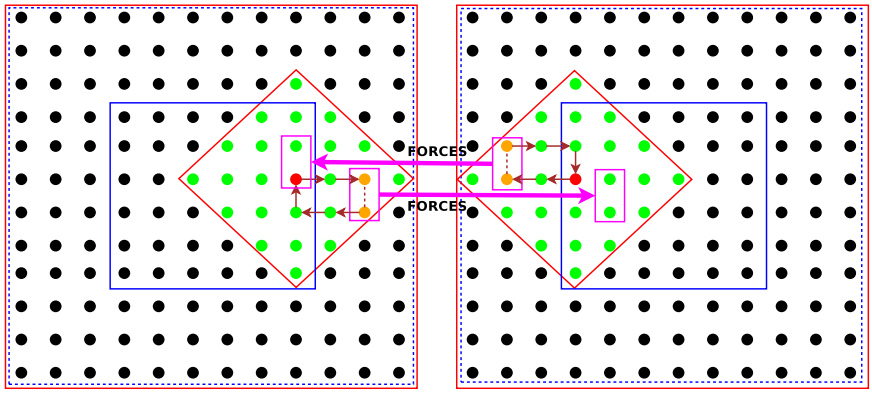

Fig. 12. Schematic of the *BOND-COMMS* algorithm for two neighbor domains. The red domain atoms act as merging points of paths that start and end in the orange atoms (by computing the corresponding inverses, analogously to Fig. 11), and after all these force contributions are computed locally for every possible merging point in the local domain, these partial results are sent to the corresponding neighbor process. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

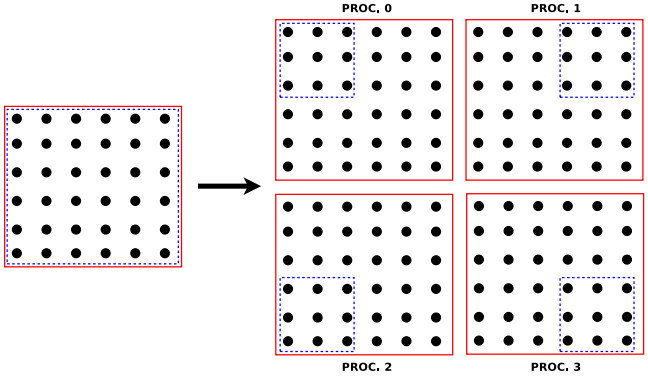

Fig. 13. Scheme of the simple work distribution algorithm *WORK-DISTR*.

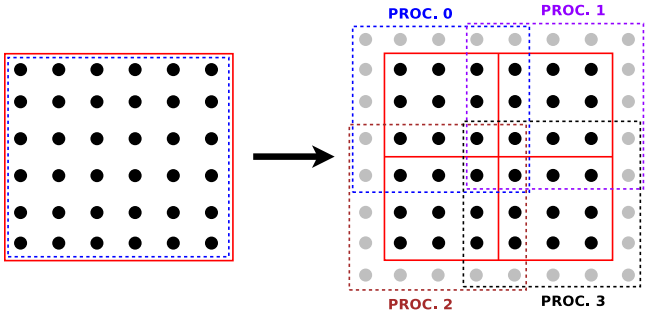

Fig. 14. Scheme of the basic domain decomposition algorithm *DOM-DECOMP* with overlap zone.

Finally, *BOND-COMMS* presents the same total number of operations as *WORK-DISTR* and *HALF-SHELL*, and only the path merge routine presents a slightly different number of operations compared to these previous cases. This fact happens because this algorithm does not merge paths according to the starting point, but to the merge point: therefore, the ideal work distribution could only be obtained if all domains had exactly the same atom layout, but here the differences in the limits of every domain cause a small fluctuation in the number of operations (around 1%, as Table 2 shows).

4.3. Analysis of different system sizes

Figs. 15–17 present the performance results of the different parallel implementations for executions with 8 and 64 processes (i.e. $2 \times 2 \times 2$ and $4 \times 4 \times 4$ grids, respectively) using moment expansions of $m = 6$ and $m = 8$. The system sizes in the X-axis are indicated in terms of the number of atoms in the system, but the system sizes tested here increase with the number of bcc cells per dimension (i.e. a polynomial increase, because a system with *ncells* per dimension has $2 \cdot (ncells)^3$ atoms).

The execution times in Fig. 15 indicate that the *WORK-DISTR* algorithm presents the best results for simulations using 8 processes, but with very low memory scalability. The redundant computations in the overlap zone define the differences in performance between *DOM-DECOMP* and *MIN-REDUND*. Additionally, the merge procedure presented in Fig. 11 allows better memory scalability for *DOM-DECOMP*, as merged paths are built by iterating on their merging point *i*. Thus, it is possible to perform independent path computations for every atom without keeping all paths for all atoms in memory at a time, as done in the merge algorithm in Fig. 4.

The communications in *FULL-SHELL* and *HALF-SHELL* show to be useful to obtain very similar performance to *WORK-DISTR* with better memory scalability, even though the system sizes simulated are not as large as in the *MIN-REDUND* case because of the allocation of communication buffers. Nevertheless, the *BOND-COMMS* algorithm presents a very good scalability that is comparable to the *DOM-DECOMP* approach, because of the use of the memory-efficient path merge routine mentioned before. As *BOND-COMMS* requires the allocation of communication buffers, more memory space is consumed than in *DOM-DECOMP*, but this additional buffering is used mainly for communicating force contributions, which show much lower memory consumption compared to path information for *FULL-SHELL* and *HALF-SHELL*.

The results for 64 processes in Fig. 16 follow a similar trend for the simplest algorithms. The low scalability of *WORK-DISTR* is clearly illustrated, as the increase in the number of threads of the system does not overcome the memory limitation, thus *DOM-DECOMP* and *MIN-REDUND* are used as reference values for comparison for longer system sizes. In general, all approaches based on domain decomposition are able to perform large simulations with 64 processes, reaching a maximum size of 190 cells per dimension (13,718,000 atoms) for *DOM-DECOMP*. The *FULL-SHELL* algorithm obtains very similar performance to the *MIN-REDUND* approach with 6 moments, even though these differences increase for 8 moments because of the communications overhead. The use of *HALF-SHELL* improves the performance of *MIN-REDUND* because of the combined approach using path and force communications, but once again *BOND-COMMS* provides the best combination of high performance and memory scalability.

Considering these results, and also the fact that an 8-moment simulation already offers a good approximation for BOPs, it is

Table 2
Matrix multiplications per process (31,250 atoms, 6 moments, 8 processes).

Algorithm	# Matrix multiplications per proc.			
	Minimum per proc.		# Maximum per proc.	
	Half paths	Merge paths	Half paths	Merge paths
WORK-DISTR	4,680,000	3,467,100	4,710,000	3,489,325
DOM-DECOMP	14,204,018	10,531,394	14,204,018	10,531,394
MIN-REDUND	6,986,107	3,762,621	7,140,342	3,796,156
FULL-SHELL	4,680,000	3,762,621	4,710,000	3,796,156
HALF-SHELL	4,680,000	3,467,100	4,710,000	3,489,325
BOND-COMMS	4,680,000	3,449,647	4,710,000	3,525,722

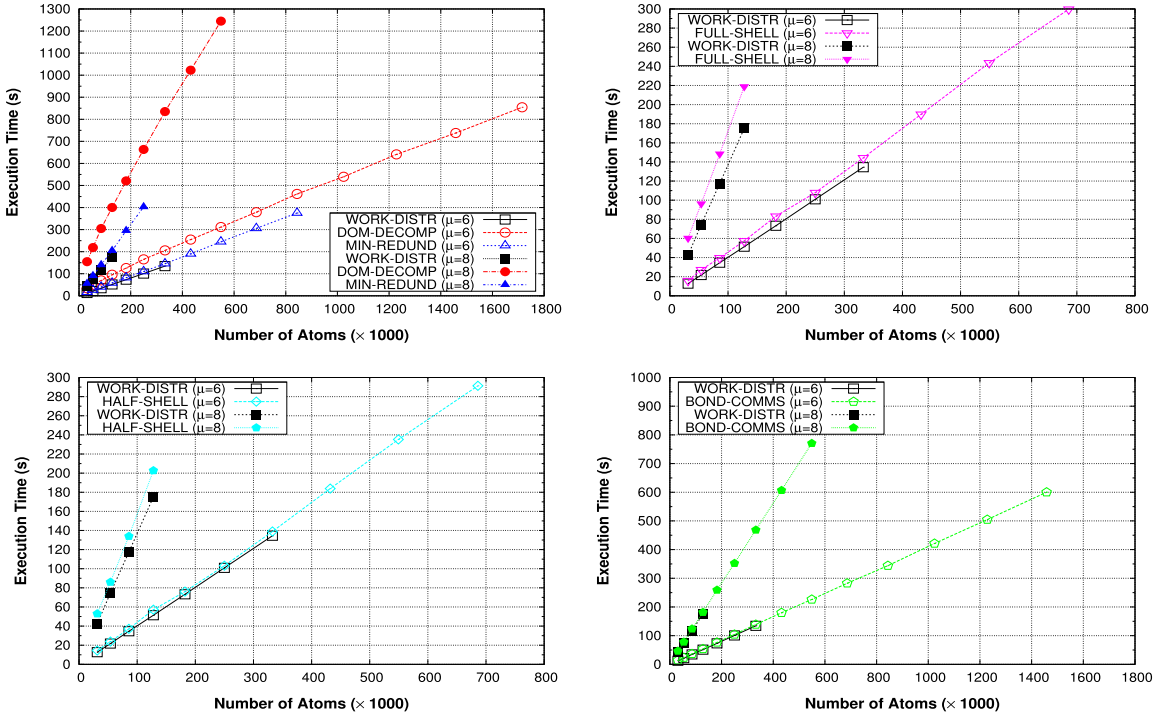

Fig. 15. Performance results with 8 processes, comparing the *WORK-DISTR* algorithm with other approaches, shown from left to right and from up to down: (1) *DOM-DECOMP* and *MIN-REDUND*, (2) *FULL-SHELL*, (3) *HALF-SHELL* and (4) *BOND-COMMS*.

possible to conclude that the use of communications for the whole overlap area with *BOND-COMMS* is not a relevant disadvantage in order to obtain the best performance for a typical analytic BOP simulation. In fact, the real drawback for the simulation is the communication of half-length paths, which represent a large amount of information, even when only a thin layer of atoms in the overlap zone is considered (i.e. nearest neighbors of domain atoms). This is due to the dependency between the moment expansion and the size of the overlap zone: a larger moment expansion increases the size of the overlap area that *BOND-COMMS* needs to send, but it also increases the number of half-length paths that have to be computed locally by *MIN-REDUND* and sent by *FULL-SHELL* and *HALF-SHELL*. These facts can be explained formally by describing the maximum size of communications per atom in terms of double-precision values for *HALF-SHELL* ($Comms_{HS}$) and *BOND-COMMS* ($Comms_{BC}$) as follows, considering homogeneous domains with atom density ρ composed by atoms with n_{orb} orbitals and k neighbors within a cut-off radius R_{cut}, and using a moment expansion m:

$$Comms_{HS} = \frac{4\pi}{3} \cdot \rho \cdot \left(\left\lceil \frac{m}{2} \right\rceil \cdot R_{cut} \right)^3 \cdot n_{orb}^2 \tag{5}$$

$$Comms_{BC} = \left\lceil \frac{m}{2} \right\rceil \cdot k \cdot n_{orb}^2. \tag{6}$$

As Eqs. (5) and (6) show, the maximum communication per atom for *HALF-SHELL* depends directly on the volume of interest around the given atom according to the moment expansion m, whereas for *BOND-COMMS* communications are always a function of the nearest neighbors.

Fig. 17 summarizes the general performance comparison between the *DOM-DECOMP* and *BOND-COMMS* algorithms for simulations using 6 moments, using a different number of processes up to the maximum possible simulation size on JUROPA. The results clearly indicate that the maximum system sizes tend to grow linearly with the number of processes in the parallel execution, without any significant restriction.

4.4. Scalability analysis

The strong scaling behavior of the different parallel codes is shown in Fig. 18 for small (31,250 atoms) and medium (250,000 atoms) system sizes with 6 moments. For the small test case, the executions up to 16 threads (not all of them are shown here) are more efficient with *WORK-DISTR*. For larger thread counts the speedup starts decreasing, and this situation becomes worse for the medium test case. The reason is the replication of the whole system in memory and the large collective reduction operations over force computations, which start saturating performance notably as the number of threads increases. The algorithms based

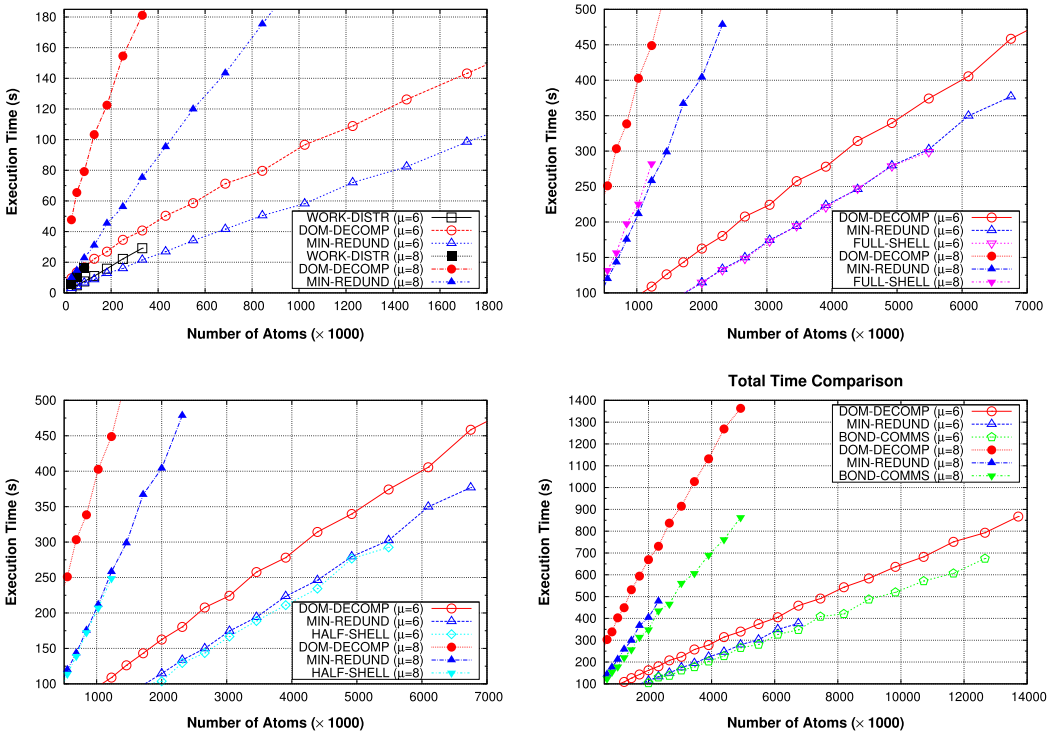

Fig. 16. Performance results with 64 processes, comparing both *DOM-DECOMP* and *MIN-REDUND* algorithms with other approaches, shown from left to right and from up to down: (1) *WORK-DISTR*, (2) *FULL-SHELL*, (3) *HALF-SHELL* and (4) *BOND-COMMS*.

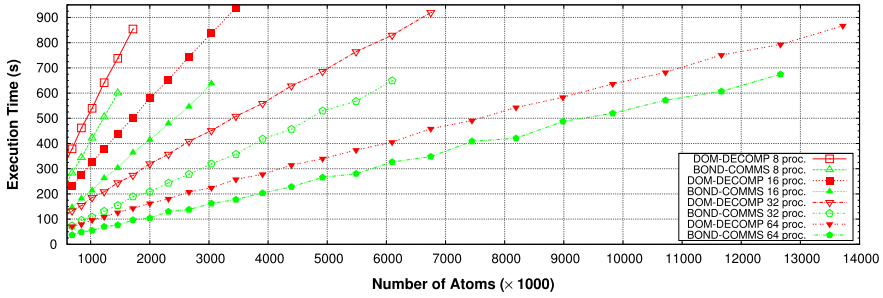

Fig. 17. Performance results with 8, 16, 32 and 64 processes.

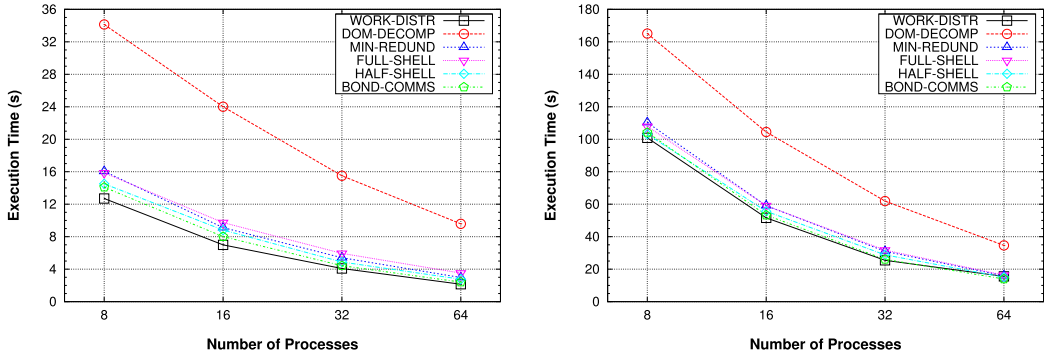

Fig. 18. Strong scaling comparison for 31.25K (left) and 250K atoms (right).

on domain decomposition show good scaling behavior, which only gets a slightly worse evolution for the *FULL-SHELL* approach, basically because of the significant amount of half-length path communications. Once again, *BOND-COMMS* presents the best overall scaling for test cases with the largest thread count.

Finally, Fig. 19 illustrates the weak scaling behavior of the simulation using a small system size (1024 atoms) per process, with 6 moments. Each simulated system size and its corresponding grid of processes have been created in order to obtain cubic domains for every process, so that there is no significant difference in terms of number of atoms, layout and workload. The small system size has been conveniently chosen to allow the execution of 8 processes per node, thus benefiting from more efficient intra-node communications. The results of *WORK-DISTR* are not shown here, as representative results cannot be obtained because of the restricted memory scalability.

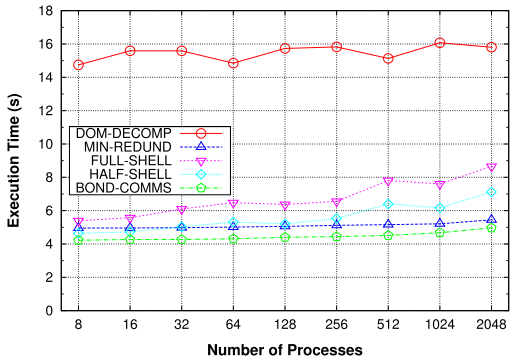

Fig. 19. Weak scaling comparison with 1024 atoms per process.

The best weak scaling behavior is shown by *BOND-COMMS*, which also shows a very regular behavior comparable to the *MIN-REDUND* approach. This indicates that the force communications do not represent a strong drawback on performance for this system size, even though the *BOND-COMMS* tests with 1024 and 2048 processes present slight increases in the execution time that approach the results of *MIN-REDUND*. The *DOM-DECOMP* approach presents a relatively stable behavior because of the lack of communications, even though the redundancy keeps its results far from the optimal performance. Regarding the *FULL-SHELL* and *HALF-SHELL* algorithms, a very significant increase in the execution time appears for larger simulations, which indicates that the half-length path communications start to dominate the performance of the code: *HALF-SHELL* only performs better than *FULL-SHELL* because half of these communications are used, but the trend is completely analogous in both cases.

It is interesting to note that the use of 1024 atoms per process for the given atomic configuration is close to the lower size limit of the domains, i.e. the point where the extent of the overlap areas would need to include atoms from overnext-neighbor domains. Consequently, these results confirm that the overhead of communications for *BOND-COMMS* is controllable for highly parallel executions, as well as for the case of large domains shown in the previous section.

5. Conclusions

We present the analysis and implementation of high-performance parallel analytic bond-order potentials (BOPs), from the description of core parts of the original sequential code to the performance analysis of different parallel approaches. The key contributions of the present paper are (1) the description of possible work distribution and domain decomposition techniques for this simulation, (2) the study of the exploitation of data locality through the definition of an overlap zone around each domain, (3) the use of different communication patterns for the transfer of path and/or force information, and especially (4) the presentation of the first large-scale simulations of analytic BOPs for millions of atoms, with the possibility of taking advantage of larger computational resources to scale the system in an efficient way.

The use of a domain decomposition has shown to be the best approach for memory and performance scalability: the system is therefore distributed in different domains with an associated overlap zone, which allows local computations for half-length paths. However, the whole set of paths requires either redundant computations on the overlap zone or communications with other domains. Five different parallel approaches (*DOM-DECOMP*, *MIN-REDUND*, *FULL-SHELL*, *HALF-SHELL* and *BOND-COMMS*) have been studied in order to show different refined computations, from high redundancy with fully local computations to low redundancy with higher communication load. The use of redundancy to avoid communications has shown very good memory scalability for *DOM-DECOMP* and *MIN-REDUND*, but the results can only converge to an optimal value if the volume of the overlap area becomes negligible with respect to the domain size. Moreover, path communications in *FULL-SHELL* and *HALF-SHELL* represent a large burden in performance, especially as the number of processes and the moment expansion increase. As a result, the *BOND-COMMS* approach has shown to provide the best compromise between performance and scalability, because of the lack of redundant computations and the path merge routine, which only requires the communication of force contributions.

Acknowledgment

This work has been financially supported by ThyssenKrupp Steel Europe AG.

References

[1] R. Drautz, D.G. Pettifor, Valence-dependent analytic bond-order potential for transition metals, Phys. Rev. B 74 (2006) 174117.
[2] R. Drautz, D.G. Pettifor, Valence-dependent analytic bond-order potential for magnetic transition metals, Phys. Rev. B 84 (2011) 214114.
[3] R. Drautz, T. Hammerschmidt, M. Cak, D.G. Pettifor, Bond-order potentials: derivation and parameterization for refractory elements, Mod. Sim. Mat. Sci. Eng. 23 (2015) 074004.
[4] D. Pettifor, Bonding and Structure of Molecules and Solids, Oxford Science Publications, Oxford, 1995.
[5] T. Hammerschmidt, R. Drautz, Bond-order potentials for bridging the electronic to atomistic modelling hierarchies, in: S.B.J. Grotendorst, N. Attig, D. Marx (Eds.), Multiscale Simulations Methods in Molecular Sciences, vol. 42, John von Neumann Institute for Computing, Jülich, 2009, pp. 229–246.
[6] S. Plimpton, Fast parallel algorithms for short range molecular dynamics, J. Comput. Phys. 117 (1995) 1.
[7] F. Abraham, R. Walkup, H. Gao, M. Duchaineau, T.D.L. Rubia, M. Seager, Simulating materials failure by using up to one billion atoms and the world's fastest computer: Brittle fracture, Proc. Natl. Acad. Sci. 99 (2002) 5777–5782.
[8] K. Kadau, T.C. Germann, P.S. Lomdahl, Molecular-dynamics comes of age: 320 billion atom simulation on bluegene/L, Internat. J. Modern Phys. C 17 (2006) 1755.
[9] T.C. Germann, K. Kadau, Trillion-atom molecular dynamics becomes a reality, Internat. J. Modern Phys. C 19 (2008) 1315–1319.
[10] T. Hammerschmidt, B. Seiser, M.E. Ford, D.G. Pettifor, R. Drautz, BOPfox program for tight-binding and bond-order potential calculations.
[11] C. Teijeiro, T. Hammerschmidt, B. Seiser, R. Drautz, G. Sutmann, Complexity analysis of simulations with analytic bond-order potentials, Mod. Sim. Mat. Sci. Eng. 24 (2016) 025008.
[12] LAMMPS Homepage, http://lammps.sandia.gov, Accessed December 2015.
[13] H. Aktulga, J. Fogarty, S. Pandit, A. Grama, Parallel reactive molecular dynamics: Numerical methods and algorithmic techniques, Parallel Comput. 38 (2012) 245–259.
[14] D. Shaw, A fast, scalable method for the parallel evaluation of distance-limited pairwise particle interactions, J. Comput. Chem. 26 (2005) 1318–1328.
[15] K. Bowers, R. Dror, D. Shaw, Zonal methods for the parallel execution of range-limited n-body simulations, J. Comput. Phys. 221 (2007) 303–329.
[16] M. Kunaseth, R. Kalia, A. Nakano, K. Nomura, P. Vashishta, A scalable parallel algorithm for dynamic range-limited n-tuple computation in many-body molecular dynamics simulation, in: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'13), Denver, CO, USA, 2013, pp. 71:1–71:12.
[17] C. Teijeiro, T. Hammerschmidt, R. Drautz, G. Sutmann, Parallel bond order potentials for materials science simulations, in: 4th Intl. Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering, PARENG'15, Dubrovnik, Croatia, 2015, Paper 4.