Available online at www.prace-ri.eu

Partnership for Advanced Computing in Europe

Development of an Advanced Implicit Solvent Model in DL_POLY

D. Grancharov[a], N. Ilieva[a], E. Lilkova[a], L. Litov[a], S. Markov[a], P. Petkov[a]*, I. Todorov[b]

[a]*NCSA, Akad. G. Bonchev 25A, Sofia 1311, Bulgaria*
[b]*STFC Daresbury Laboratory, Daresbury, Warrington WA4 4AD, UK*

Abstract

A library, implementing the AGBNP2 [1, 2] implicit solvent model that was developed within PRACE-2IP [3] is integrated into the DL_POLY_4 [4] molecular dynamics package in order to speed up the time to solution for protein solvation processes. Generally, implicit solvent models lighten the computational loads by reducing the degrees of freedom of the model, removing those of the solvent and thus only concentrating on the protein dynamics that is facilitated by the absence of friction with solvent molecules. Furthermore, periodic boundary conditions are no longer formally required, since long-range electrostatic calculations cannot be applied to systems with variable dielectric permittivity. The AGBNP2 implicit solvation model improves the conformational sampling of the protein dynamics by including the influence of solvent accessible surface and water-protein hydrogen bonding effects as interactive force corrections to the atoms of protein surface. This requires the development of suitable bookkeeping data structures, in accordance with the domain decomposition framework of DL_POLY, with dynamically adjustable inter-connectivity to describe the protein surface. The work also requires the use of advanced b-tree search libraries as part of the AGBNP library, in order to reduce the memory and compute requirements, and the automatic derivation of the van der Waals radii of atoms from the self-interaction potentials.

Introduction

Water is the most common solvent for most biological reactions and plays a vital role in determining the structures and dynamics, and hence the function, of globular proteins. Therefore, it is of primary importance to account for the water-solute interactions in an MD simulation. In biomolecular modeling there are two alternatives for describing the solvent. Firstly there is the approach where, within the explicit solvent framework, movements of individual water molecules are calculated explicitly. Due to the enormous number of solvent degrees of freedom, this methodology is not particularly useful and suitable for molecular systems undergoing significant structural transitions, i.e. in protein folding, allostery processes, calculations of relative free energies of molecular conformations, studying protein-protein interactions, etc. An alternative approach is to replace the real water environment consisting of discrete molecules by a continuum with the dielectric and “hydrophobic” properties of water. This methodology allows enhanced sampling of conformational space due to lack of solvent viscosity and provides an effective way for free energy estimation by reducing the number of local “noise” minima, arising from the small variations in solvent configuration [5].

The implemented AGBNP2 (Analytical Generalized Born plus Non-polar) implicit solvation model [2] uses a parameter-free and conformational-dependent algorithm to estimate the pairwise descreening scaling coefficients

*Corresponding author. *E-mail address*: *peicho.petkov@gmail.com*

in the evaluation of Born radii. The same algorithm is also used to evaluate atomic surface areas. In addition, a non-polar estimator is introduced that does not depend exclusively on the solute surface area. It is based on the decomposition of the non-polar hydration free energy into a cavity term, proportional to surface area, and an attractive dispersion energy term. The latter derives the continuum solvent solute-solvent Van der Waals interaction energy using a functional form based on the Born radius of each atom. The non-polar model depends linearly on adjustable parameters that measure the effective surface tension and effective strength of solute-solvent Van der Waals interactions. The AGBNP2 model is applicable to a wide range of molecules and functional group topologies and types. The model is also suitable to study absolute hydration free energies as well as conformational equilibria. The model was previously implemented into a Fortran90 library [3] (PRACE 2IP) relying on OpenMP parallelisation. It was agnostic in terms of what particle software it would be used by and thus took general input in terms of 2-body interaction list, short-range cutoff, and Born radii and Lennard-Jones interaction parameters for the species of the solvated molecule.

DL_POLY_4 is a general purpose classical molecular dynamics simulation software, that can be used to simulate a wide variety of molecular systems including simple liquids, ionic liquids and solids, small polar and non-polar molecular systems, bio- and synthetic polymers, ionic polymers and glasses, solutions, simple metals and alloys. The software package is written in modularised Fortran90 with a parallelisation strategy based on 3D equi-spatial domain decomposition (DD). The DD implementation relies on a static mapping of domains to MPI tasks. The communications between tasks are mostly local and involve exchange of cutoff based halo regions by neighbouring domains in order to facilitate independent work on particle interactions for domain particles within a cutoff distance off the domain border (i.e. particles that are halo for the neighbouring domains). This DD setup leads to almost perfect work load balancing and thus ensures for an excellent scalability provided that the particle density does not vary significantly in space and time. It is worth mentioning that there are a small number of global operations needed during the MD step cycle as well as non-linearly scaling algorithms such as I/O to disc and SPME electrostatics evaluation, based on 3D FFT, which may affect the overall performance when in use.

So far when studying bimolecular systems, DL_POLY only supported explicit solvent simulations under periodic boundary conditions. The paper discusses the most important aspects of the implementation of the AGBNP2 implicit solvation model within the program’s code. The motivation for this work was not only to enable DL_POLY_4 with an advanced implicit solvation methodology but also provide a unique method for calculations of hydration energies. This will allow DL_POLY users to employ implicit solvent description when simulating macromolecular systems or biologically relevant large-scale processes and tune the model specifically to their systems of interest when comparison of hydration energies is required between different types of solvated macromolecules.

Theory

In the AGBNP2 model [2], the solute is described as a set of overlapping spheres of radius R_i, centered on the atomic positions $\vec{\boldsymbol{r_i}}$ and its volume is given by the Poincaré formula [1]. The self-volume of atom *i* that measures the solute volume, which belongs exclusively to that atom is:

$$V_i' = V_i - \frac{1}{2}\sum_j V_{ij} + \frac{1}{3}\sum_{i<j} V_{ijk} + \cdots, \tag{1}$$

where $V_i = \frac{3}{4}\pi R_i^3$ is the volume of atom *i*, V_{ij} is the volume intersection of atoms *i* and *j* (second order intersection), V_{ijk} is the volume of intersection of atoms *i*, *j*, and *k* (third order intersection), and so on. The total volume of the molecule is $V = \sum_i V_i{}'$. The overlap volume formed by n spheres is then approximated by the integral of the product of the n corresponding Gaussian functions:

$$V_{12\ldots n}^{g} = p_{12\ldots n}\, ex\, p(-K_{12\ldots n}\,)\left(\frac{\pi}{\Delta_{12\ldots n}}\right)^{\frac{3}{2}}, \tag{2}$$

where the coefficients *p*, *K* and *Δ* are defined as follows:

$$p_{12\ldots n} = p^n, \qquad p = \frac{4\pi}{3}\,\left(\frac{\kappa}{\pi}\right)^{\frac{3}{2}} \tag{3}$$

$$K_{12..n} = \frac{1}{\Delta_{12\ldots n}}\ \sum_{i=1}^{n}\sum_{j=i+1}^{n} c_i\, c_j\, r_{ij}^2\,; \qquad \Delta_{12\ldots n} = \sum_{i=1}^{n} c_i\,; \qquad c_i = \frac{\kappa}{R_i^2}; \qquad \kappa = 2.227 \tag{4}$$

Only intersection volumes that are above a certain threshold are taken into account:

$$V_{12\dots n} = \begin{cases} 0, & V_{12\dots}^{g} \leq v_1 \\ V_{12\dots n}^{g} \, f_w\,(u), & v_1 < V_{12\dots}^{g} < v_2; \\ V_{12\dots n}^{g}, & V_{12\dots}^{g} \geq v_2 \end{cases} \qquad u = \frac{V_{12\dots}^{g} - v_1}{v_2 - v_1}; \qquad f_w\,(x) = x^3\,(10 - 15x + 6x^2)\,, \tag{5}$$

where $v_1$and v_2 are predefined constants.

The atomic surface is the derivative of the volume with respect to the atomic radius R_i,

$$A_i = f_a\,\left(\frac{\partial V}{\partial R_i}\right); \qquad f_a\,(x) = \begin{cases} \frac{x^3}{a^2 + x^2}, & x > 0 \\ 0, & x \leq 0 \end{cases}, \tag{6}$$

where f_a ensures positivity of A.

Once the geometric properties of the molecule are calculated, we can calculate the different contributions to the solvation free energy, given in the AGBNP2 model by

$$\Delta G_h = \Delta G_{gb} + \Delta G_{cav} + \Delta G_{vdw} + \Delta G_{hb}. \tag{7}$$

The non-polar term consists of a term ΔG_{cav}, describing cavity formation energy, and a Van der Waals interaction energy term ΔG_{vdw} with solvent molecules:

$$\Delta G_{cav} = \sum_i \gamma_i\, A_i, \tag{8}$$

where γ_i is the surface tension parameter assigned to atom *i* and A_i is its Van der Waals surface area;

$$\Delta G_{vdw} = \sum_i \frac{\alpha_i\, a_i}{(B_i + R_w)^3}; \qquad \alpha_i \approx 1; \qquad a_i = -\frac{16}{3}\,\pi\rho_w\,\epsilon_{iw}\,\sigma_{iw}^6\,, \tag{9}$$

where α_i is an adjustable dimensionless parameter of the order of 1 and ρ_w is the number density of water at standard conditions, σ_{iw} and ϵ_{iw} are the Lennard-Jones parameters for the interaction between atom *i* and an oxygen atom of the water model, and $R_w = 1.4$ Å.

The solute-solvent electrostatic interaction is given by:

$$\Delta G_{gb} = u_\epsilon \sum_i \frac{q_i^2}{B_i} + 2u_\epsilon \sum_{i<j} \frac{q_i q_j}{f_{ij}}. \tag{10}$$

The Born radius B_i of a solute atom is calculated using

$$B_i^{-1} = f_b\,(\beta_i\,) = \begin{cases} \sqrt{b^2 + \beta_i^2}, & \beta_i > 0 \\ b, & \beta_i \leq 0 \end{cases}; \qquad b^{-1} = 50\text{Å}. \tag{11}$$

Again, the function $f_b(\beta)$ keeps the result finite. Its parameter β_i is obtained using the volume scaling s_{ij} and the pair descreening functions Q_{ij} (Appendix B in ref [1])

$$\beta_i = \frac{1}{R_i} - \frac{1}{4\pi} \sum_{i \neq j} s_{ij} Q_{ij}\,; \qquad s_{ij} = s_j + \frac{V'_{ij}}{V_j}; \qquad s_j = \frac{V'_j - d_j A_j}{V_j}; \qquad d_j = \frac{1}{3}\,R'_j \left[1 - \left(\frac{R_j}{R'_j}\right)^3\right] \tag{12}$$

$$V'_{ji} = V'_{ij} = \frac{1}{2}\,V_{ij} - \frac{1}{3} \sum_k V_{ijk} + \frac{1}{4} \sum_{k<l} V_{ijkl} - \cdots, \tag{13}$$

where R'_j is the augmented radius $\mathrm{R}' = \mathrm{R} + \mathrm{dR}, \mathrm{dR} = 0.5$ Å.

Results

The previously developed Fortran90 code of the AGBNP2 library [3] was incorporated in a local version of the DL_POLY_4.05.1 source code. The integration complied with most of the data structures already available in the code. It was designed to stand alone as complementary and independently as possible. Configuration related input data, such as number of atoms, atomic positions, atom names and major Verlet neighbour listings, are imported from DL_POLY's *config_module*. Interaction parameters had to be created on parsing short-range interaction parameters in order to define specific epsilon and sigma in a Lennard-Jones casting for all possible kinds of short-range interaction. These were incorporated in the *vdw_module* and then used as input from there. Working precision and some relevant constants needed for the calculations within the AGBNP2 library are used from the *setup_module*. The *agbnp2_module* contains the AGBNP2 library adapted in a manner commensurate with the MPI framework of DL_POLY_4's domain decomposition so that it included its original OpenMP parallelism within each MPI domain task. The work on integrating the AGBNP2 library within DL_POLY_4.05.1 included the following optimisation and adaptation tasks:

- Arrays referring to neighbour lists (doublets, triplets, quadruplets) were modified to match the look up style in DL_POLY. List ends were hardcoded in the 0th element of the list in order to optimise multiply nested *do* loops for look up and add up functions when calculating contributions;
- The DL_POLY's Verlet neighbour list (VNL), which is unordered and single-sided, had to be split into two lists for the AGBNP2 library: **(i)** a very short range one for neighbours up to 3 Angstroms distance so that the search over possible doublets, triplets and quadruplets is minimised as based on cross-sectioning of spheres with Born radii (< 3 Angstroms) of the species involved in the multiplets; and **(ii)** a complementary list for neighbours from 3 Angstroms to the full range of the non-bonded cut-off. This list is needed for applying corrections to the GB energy calculations.
- The split of this list involved changes to the original AGBNP2 library in terms of loops optimisations as otherwise the library would not have worked at all for architectures with small memory per core allowance and as we found worked very slowly due to the excessive search over the long VNL as supplied originally by DL_POLY.
- DL_POLY was enabled to provide the Lennard-Jones's characteristic length (sigma) and energy (epsilon) values as required by the OPLSAA force-field for calculations. As DL_POLY has no force-filed of its own a number of routines and modules had to be adapted so that this was possible for all possible short-range potentials forms (about 10 different potentials, available in DL_POLY), including numerical search for potentials supplied in a tabulated form.
- All force pre-calculations in the library had to be modified to include interactions of halo atoms with domain atoms according to the domain decomposition of DL_POLY. This involved selective extension of do loops over ranges of domain and halo.
- All full force calculations and their contribution had to be carefully filtered so that only qualifying atoms on the domain had the application and correction forces and energies added despite that domain atoms may interact with halo ones. This was necessary to ensure that energy contributions are not miscounted (the potential energy does not drift) and no total force is generated in the system (the kinetic energy does not lead the system to overheating).

For the purposes of demonstrating performance and scalability, a model system of our own research (the antimicrobial peptide magainine) was enlarged to a size of 46336 atoms and scaled up by a factor of two, to 92672 atoms, and by a factor of four, to 185344 atoms, using the NFOLD system enlarging routines of DL_POLY_4.

It is worth noting that the enlargement of the original system was needed because it is relatively small to demonstrate scaling by MPI size especially when the solvent is absent. This is due to the restrictions of the DD parallelisation of DL_POLY_4 that require system sizes of at least few cut-offs width per domain with relatively constant particle density across space and time. There are plenty of examples within the PDB database that are much larger and "possibly" more suitable to test this development. However, these were not considered because they had not been studied much by modellers due to the cost associated by their size, especially when solvated, hence, leading to un-refined force-field descriptions and energies of hydration, not comparable by alternative types of calculations.

The three example systems had the same force field parameters with van der Waals parameters taken according to the CHARMM22 force field, which was converted to DL_POLY input using the DL_FIELD program. For the purposes of fair scaling comparisons, the same short-range cut-off of 7 Å was employed in all benchmarking runs. It should be pointed out that larger cut-offs will lead to larger computational loads, which is better for exploiting scalable performance but worse in terms of time to solution. It also leads to a larger memory demand, which may limit the minimum node count required in order to fit the run on a given architecture (as it did in this study). The reported execution times are the time per timestep averaged over 5 steps. The test runs were performed on a Linux cluster with Intel® Xeon® E5540 @ 2.53GHz chips with 2 quadcore CPUs per node and hyperthreading enabled, which gives 16 threads per node. They are connected via Infiniband Mellanox Technologies MT26418 cards. The reported results were obtained using the *GNU* Fortran90 compiler, *gfortran* version 4.8.1, with the default optimisation level O3, and the *OpenMPI* library, version 1.6.5.

We first investigate the performance of the model as a function of number of threads per MPI task (domain). Figure 1 shows the average execution time per MD step as a function of MPI task size using a different number of threads per task. One can see that increasing the OpenMP threads per MPI task improves the performance scalability only to 4 threads per task before it plateaus as further increases only lead to the approximately same execution times. This loss of further scalability improvements can be explained as driven by the escalation of cache misses occurring when the number of threads per node (2 CPUs with 4 cores each) exceeds the number of cores per CPU. The performance upon number of threads per node is purely due to the procedure for searching corrected second order cross-section volumes (equation 30 in [3]) using b-tree owned by each of the threads. When the number of threads increases so that they have to run on both CPUs of each node the threads per node no longer share the same first level of cache and hence the saturation in performance.

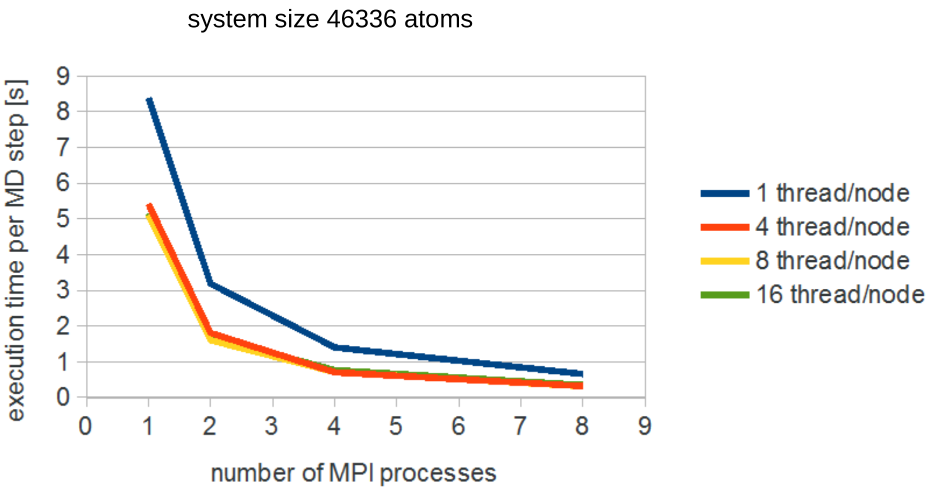

Figure 1: Average execution time per step as a function of the number of the nodes (one MPI process per node) for different number of threads per node.

We re-plot Figure 1 into Figure 2 to show the speed-up as a function of number of nodes with respect to the performance on one node. This makes it easy to see in Figure 2 that the achieved speed-ups with respect to the number of nodes with only one MPI task per node are excellent.

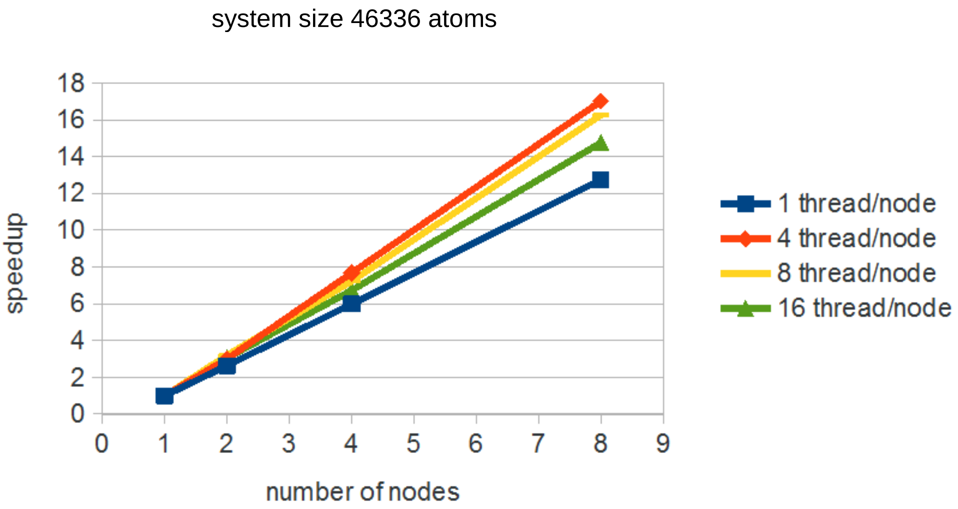

Figure 2: Speedup as a function of number of the nodes (with respect to the performance of one node).

The performance speed-up as function of number of threads for the system consisting of 46336 atoms is shown in Figure 3. As one can see, the performance scalability is close to ideal. We expect that this performance could be sustained up to about 1024 threads before the systems size to MPI tasks (domains) ratio puts DL_POLY_4 in unfavourable regimes of performance with respect to the cut-off employed. It is worth noting that system sizes are small due to the absence of discrete water molecules and that plays a limiting factor on the size of the MPI tasks before time spent in communication prevails over compute time.

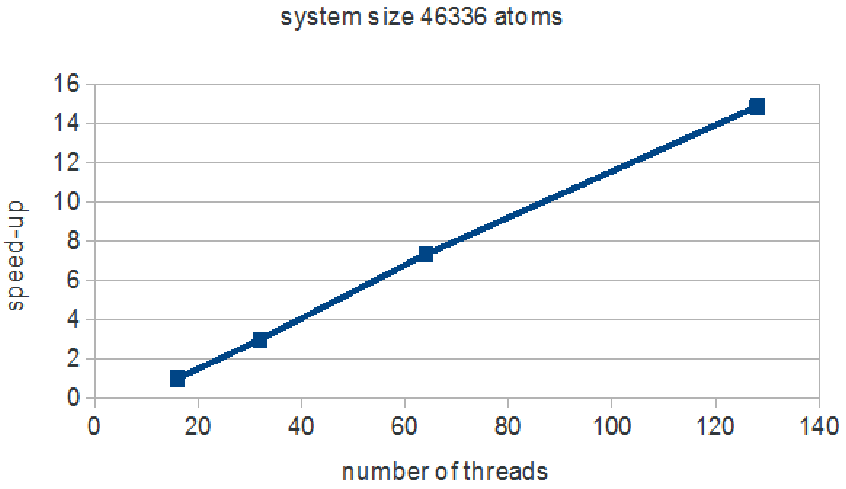

Figure 3: Speed-up upon number of threads for the system of 46336 atoms. The hybridised parallelisation involved a load of 16 OpenMP threads per MPI task.

Figure 4 presents the performance scaling of the implementation within DL_POLY_4 by system size for a fixed compute resource. It plots the average execution time per timestep on 8 nodes (128 threads) for the three systems of sizes 46336, 92672 and 185344 atoms. It is worth mentioning that in this specific version of DL_POLY_4 there is no OpenMP parallelism outside the AGBNP2 model library. Thus, the most compute intensive task within DL_POLY_4, the build-up of the Verlet neighbour list structures, is not OpenMP parallelised and hence the executions times are larger than those expected for these system sizes. It is the domination of this task that in fact leads to the super scaling observed in the figure. It is solely due the decreasing compute cost of the linked-cells construction pre-factor with respect to the cost for building up the Verlet neighbour list as the system size increases.

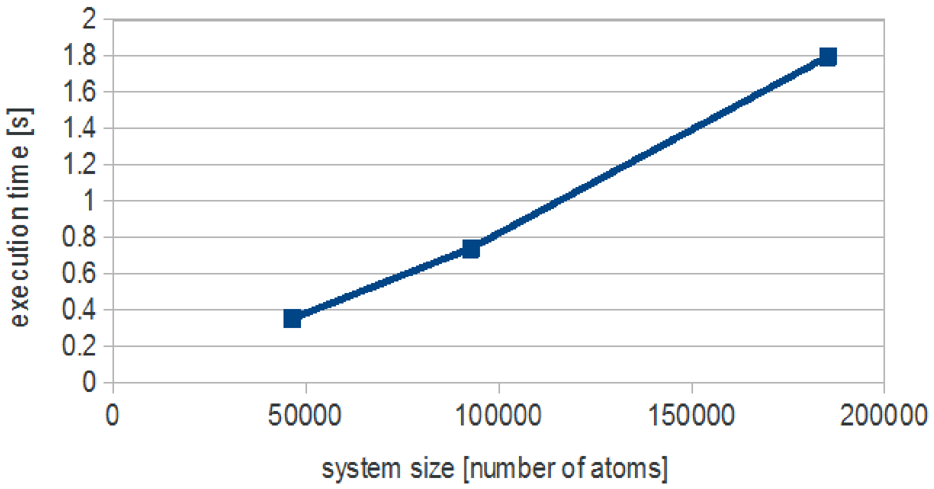

Figure 4: Execution time on 8 MPI tasks with 16 OpenMP threads each versus system size.

Conclusions

The AGBNP2 inclusion within DL_POLY_4 was motivated by the need of implicit solvation methodology for biochemical users interested in simulating macromolecular systems or biologically relevant large-scale processes. The model provides a unique and highly tunable method for calculating hydration energies of biological macromolecules. This is particularly useful for tackling the socio-economic challenges related to the modeling of molecular systems undergoing significant structural transitions, i.e. in protein folding, allostery processes, calculations of relative free energies of molecular conformations, studying protein-protein interactions, etc.

We discussed the most important aspects of the integration work of the AGBNP2 implicit solvation model within DL_POLY_4.05.1 and presented the results from this coupling. We showed that it for the chosen benchmarks and conditions the program exhibited excellent parallel performance in the scaling tests. The AGBNP2 model was successful in generating the correct energy contributions for the model system and reproduced them correctly per atom by both varying the model system size and changing the number of MPI tasks and OpenMP threads per task. The performance of this coupling could be further improved by including OpenMP parallelism within the rest of the DL_POLY_4 algorithms.

References

[1] E. Gallicchio and R.M. Levy, AGBNP: An Analytic Implicit Solvent Model, J. Comput. Chem. 25(4) 479-499 (2004).

[2] E. Gallicchio, K. Paris and R.M. Levy, The AGBNP2 Implicit Solvation Model, J. Chem. Theory Comput. 5(9), 2544–2564 (2009).

[3] P. Petkov, S. Markov, and I. Todorov, Development of AGBNP2 Implicit Solvent Model Library for MD Simulations, PRACE Whitepaper 111, http://www.prace-ri.eu/IMG/pdf/wp111.pdf

[4] I.T. Todorov, W. Smith, K. Trachenko and M.T. Dove, DL_POLY_3: new dimensions in molecular dynamics simulations via massive parallelism, J. Mater. Chem., 16, 1911-1918 (2006).

[5] A. Onufriev, The generalized Born model: its foundation, applications, and Limitations, September 8, 2010, http://people.cs.vt.edu/~onufriev/PUBLICATIONS/gbreview.pdf

Acknowledgements

This work was financially supported by the PRACE-3IP project funded in part by the EUs 7th Framework Programme (FP7/2007-2013) under grant agreement no. RI-312763