Supplementary information to: Parallel Hybrid Simulations of Block Copolymer Nanocomposite Systems using Coarray Fortran

Javier Diaz[1], Marco Pinna*[2], Andrei V. Zvelindovsky[2] and Ignacio Pagonabarraga†[1,3,4]

[1] CECAM, Centre Européen de Calcul Atomique et Moléculaire, École Polytechnique Fédérale de Lausanne, Batochime - Avenue Forel 2, 1015 Lausanne, Switzerland
[2]Centre for Computational Physics, University of Lincoln, Brayford Pool, Lincoln, LN6 7TS, UK
[3]Departament de Física de la Matèria Condensada, Universitat de Barcelona, Martí i Franquès 1, 08028 Barcelona, Spain
[4]Universitat de Barcelona Institute of Complex Systems (UBICS), Universitat de Barcelona, 08028 Barcelona, Spain

March 8, 2021

1 2D stencil

Figure S1 shows the 2D stencil analogy to the 3D stencil used in the simulations. Calculating the laplacian of a 2D array $\nabla\psi \approx << \psi >> -\psi$ at point $\mathbf{r}$ in the grid requires looping over the nearest-neighbors (NN) and the next-near-neighbors (NNN).

2 Computational resources

Access has been provided to several HPC facilities for this work. For reference, we list the available resources, brief description, access method and specifications.

2.1 CSCS - Centro Svizzero di Calcolo Scientifico (Swiss National Supercomputing Centre)

Access has been provided to the *Piz Daint* (website) supercomputer through the *Preparatory Project* allocation scheme. Specifications: Cray XC40 Intel® Xeon® E5-2690 v3 @ 2.60GHz (12 cores, 64GB RAM) and NVIDIA® Tesla® P100 16GB - 5704 Nodes In this machine the CRAY Fortran compiler has been used without special flags.

The Cray Aries interconnect links all compute nodes in a Dragonfly topology

2.2 Mare Nostrum / BSC-CNS (Barcelona Supercomputing Center-Centro Nacional de Supercomputación)

Access has been provided via the HPC-EUROPA3 (INFRAIA-2016-1-730897), with the support of the EC Research Innovation Action under the H2020 Programme; Mare Nostrum (BSC-CNS) (website) compute nodes have 2 sockets Intel Xeon Platinum 8160 CPU with 24 cores each @ 2.10GHz for a total

*mpinna@lincoln.ac.uk
†ipagonabarraga@ub.edu

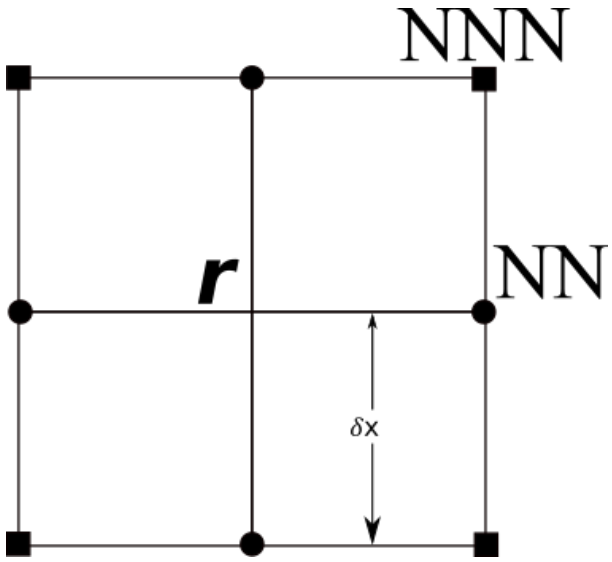

Figure S1: 2D stencil for the calculation of the laplacian. NN and NNN stand for the nearest-neighbor and next-nearest-neighbors, respectively, for the point **r** in the 2D grid.

of 48 cores per node. In this machine the GNU Fortran compiler has been used with the OpenCoarrays wrappers[1], with level 3 of optimisations.

In this machine network communications use 100Gb Intel Omni-Path Full-Fat Tree.

2.3 ARCHER - Advanced Research Computing High End Resource, UK

Access has been provided via the ARCHER Driving test. Compute nodes in ARCHER (website) have Cray XC30 MPP supercomputer where each node has two 12-core Intel Ivy Bridge, using a Cray Fortran compiler. In this machine the CRAY Fortran compiler has been used without additional optimisation flags.

The Cray Aries interconnect links all compute nodes in a Dragonfly topology

2.4 University of Lincoln [UoL]

Additionally, the Centre for Computional Physics HPC facilities at the University of Lincoln (UoL) was used with gfortran for 48 AMD Opteron 6348 CPUs per node. In this machine the GNU Fortran compiler has been used with the OpenCoarrays wrappers[1] with level of compiler optimisations.

Connection between nodes use InfiniBand.

3 Strong scaling in additional supercomputer facilities

We report the strong scaling of the purely CDS code for additional supercomputers. In figure S2 we can observe the strong scaling for three system sizes in each supercomputer. The scaling is comparatively bad for a smaller sized system $V = 128^3$ in the UoL cluster as expected for the hardware specifications. A better scaling can be achieved in two national super computers with more modern hardware: ARCHER and Mare Nostrum in (b) and (c), respectively. The Mare Nostrum supercomputer displays a satisfactory scaling as well as ARCHER supercomputer with a close to idea behaviour.

4 Comparison MPI vs CAF

In order to compare the performance of the current CAF implementation and the previously reported MPI algorithm, we display the data points from figure 1 in the main text along with the data points from reference Guo *et al* **2007**[2] (figure 5). It is clear that the scaling of the CAF code is always above the

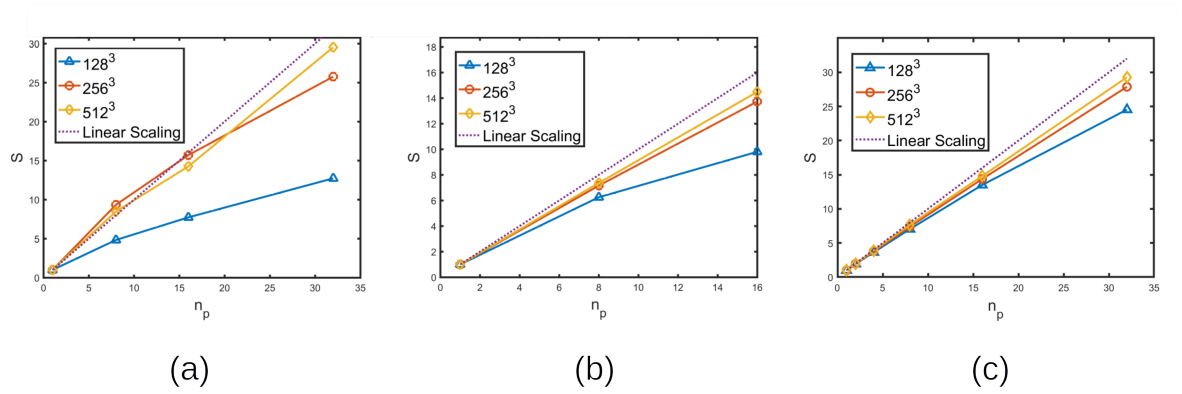

Figure S2: Single node strong scaling on additional supercomputers with the number of processors and several system sizes. The ideal scaling is shown as a dashed line. The used supercomputers (compilers) are: (a) UoL (gfortran), (b)ARCHER (CRAY) and (c) Mare Nostrum (gfortran).

MPI curve, for all three system sizes considered. It is important to note that hardware improvements in the time between these two publications should definitely be taken into consideration. To facilitate the comparison, we note that the MPI simulations were performed on SGI Altix 3700 computer which has 56 Intel Itanium-2 CPUs (1.3 GHz 3 MB L3 Cache), as stated in reference [2], using MPICH (Message Passing Interface Chameleon) with Intel. On the other hand, the hardware details of the CSCS supercomputer can be found in section 2.1.

5 Efficiency of the purely polymeric code

The efficiency of a parallel algorithm is defined as

$$E(m, n_p) = \frac{T(m, 1)}{n_p T(m, n_p)} \tag{1}$$

where n_p is the number of processors, m is the total scale of the problem and $T(m, n_p)$ is the computing time. In Fig. S5 the efficiency is displayed for several distributions of processors $n_p = N_x N_y N_z$ along X,Y and Z, respectively and a total number of $n_p = 8$ processors in two machines: (a) Mare Nostrum and (b) CSCS. Although changes in the relative elapsed time are not considerable, an optimal speed-up can be achieved using a non-square partition (ie different from $N_x = N_y = N_z$). Fortran access to arrays in memory privileges a minimal jump in the Z direction. For this reason, it is generally recommended to perform loops in the order Z-Y-X. This can explain the hierarchy of elapsed time for $S(118 > S(181) > S(811))$. Further understanding of the more complex arrangements require knowledge over the particular architecture of the HPC facilities, in this case, Mare Nostrum and CSCS supercomputers.

6 Strong scaling of the hybrid code in Mare Nostrum

In figure S6 the strong scaling for the hybrid BCP/NP code is shown with simulations performed in Mare Nostrum. The system size is $V = 512^3$ grid points while the number of particles is $N_p = 10^6$. The dotted line shows the ideal scaling. The scaling is shown to be satisfactory for up to 16 processors, but it has noticeably decreased for $n_p = 32$.

In figure S7 we break down the different contributions to the computational time in three representative conditions: (left) 8 cores with concentration $\phi_p = 2.5 \times 10^{-2}$ of NPs, (middle) 8 codes with $\phi_p = 1 \times 10^{-1}$ and (right) 16 codes with $\phi_p = 1 \times 10^{-1}$, using a fixed volume $V = 256^3$ and particle size $R_{eff} = 1.56$. The different contributions to the total time are separated in terms of the role in the hybrid simulation: polymer(red), coupling(blue) and colloidal (yellow).

Due to the various contributions, we describe them in the following lines:

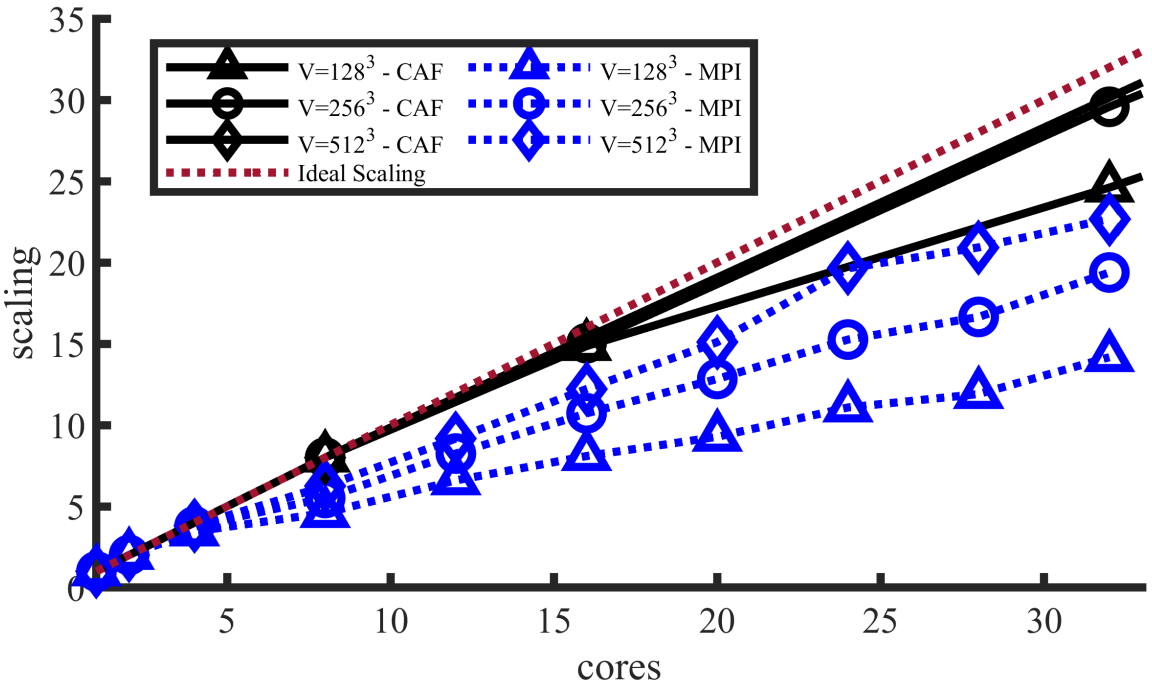

Figure S3: Comparison between the scaling of the CAF (black markers, solid line) and the MPI (blue markers, dotted lines) code. The ideal scaling $S = cores$ is shown as a dotted line. The data points for the CAF scaling are the same as in figure 1 in the main text, while the MPI data points are extracted from figure 5 in reference [2].

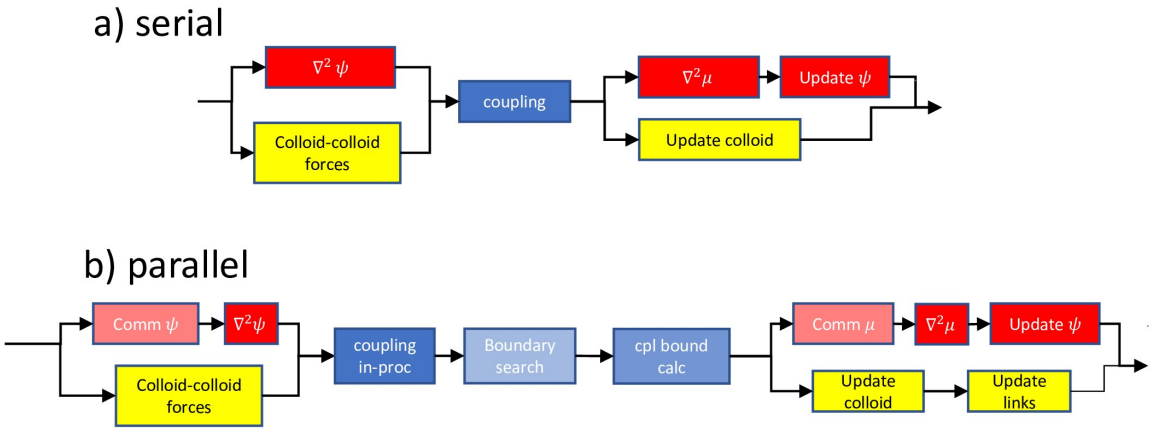

Figure S4: Flowchart of the algorithm: A single iteration (*ie*, a single time step) is shown from left to right, indicating the polymer, colloid and coupling in red, yellow and blue respectively. In a) the simpler serial algorithm is shown, while in b) we display the algorithm for the CAF implementation, including the communication steps.

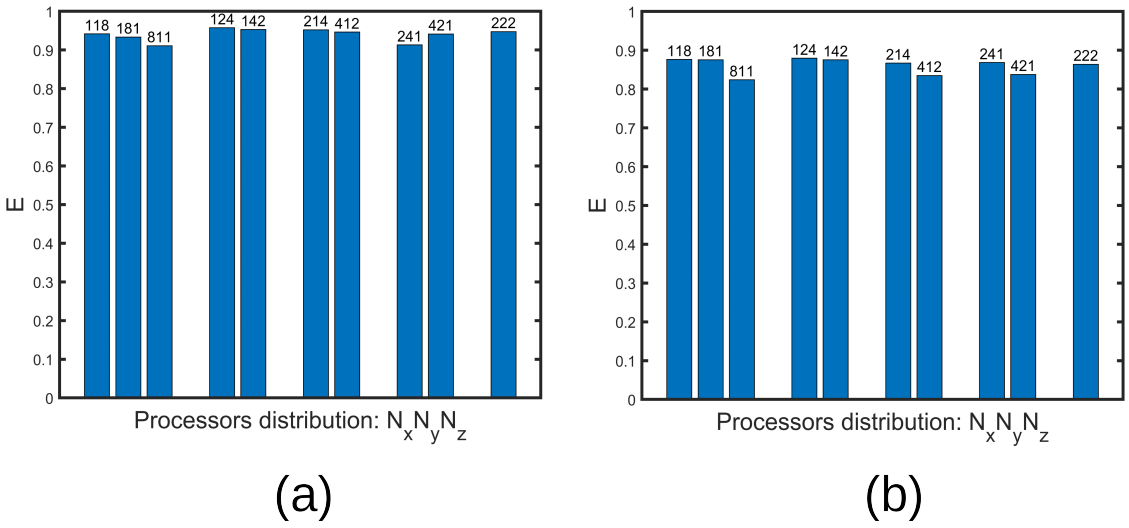

Figure S5: Efficiency of the parallel algorithm in (a) Mare Nostrum and (b) CSCS for $n_p = 8$ processors. The processors are distributed in the X, Y and Z direction as shown in each bar.

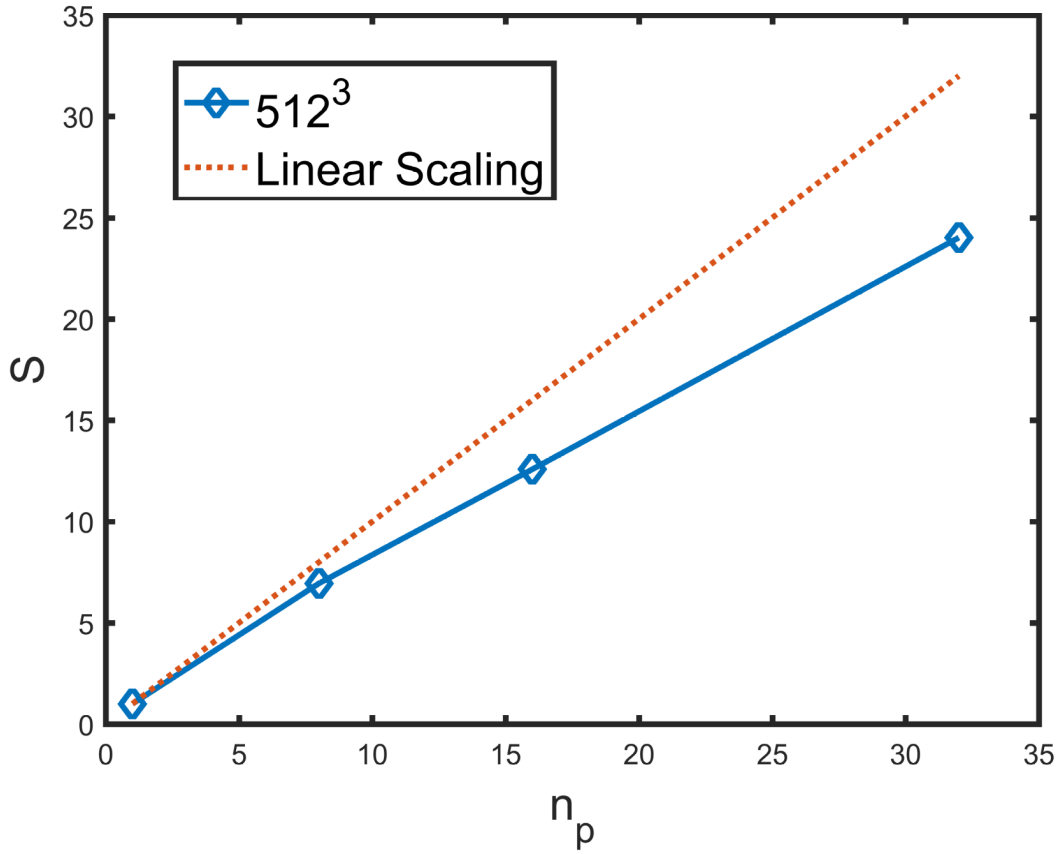

Figure S6: Strong scaling of the hybrid BCP/NP code vs the number of processors n_p in a single node. The speed-up S is compared with the ideal, linear scaling(dotted line). The number of particles is $N_p = 10^6$ and simulations were performed in Mare Nostrum. The coupling constant is set to $\sigma = 1$.

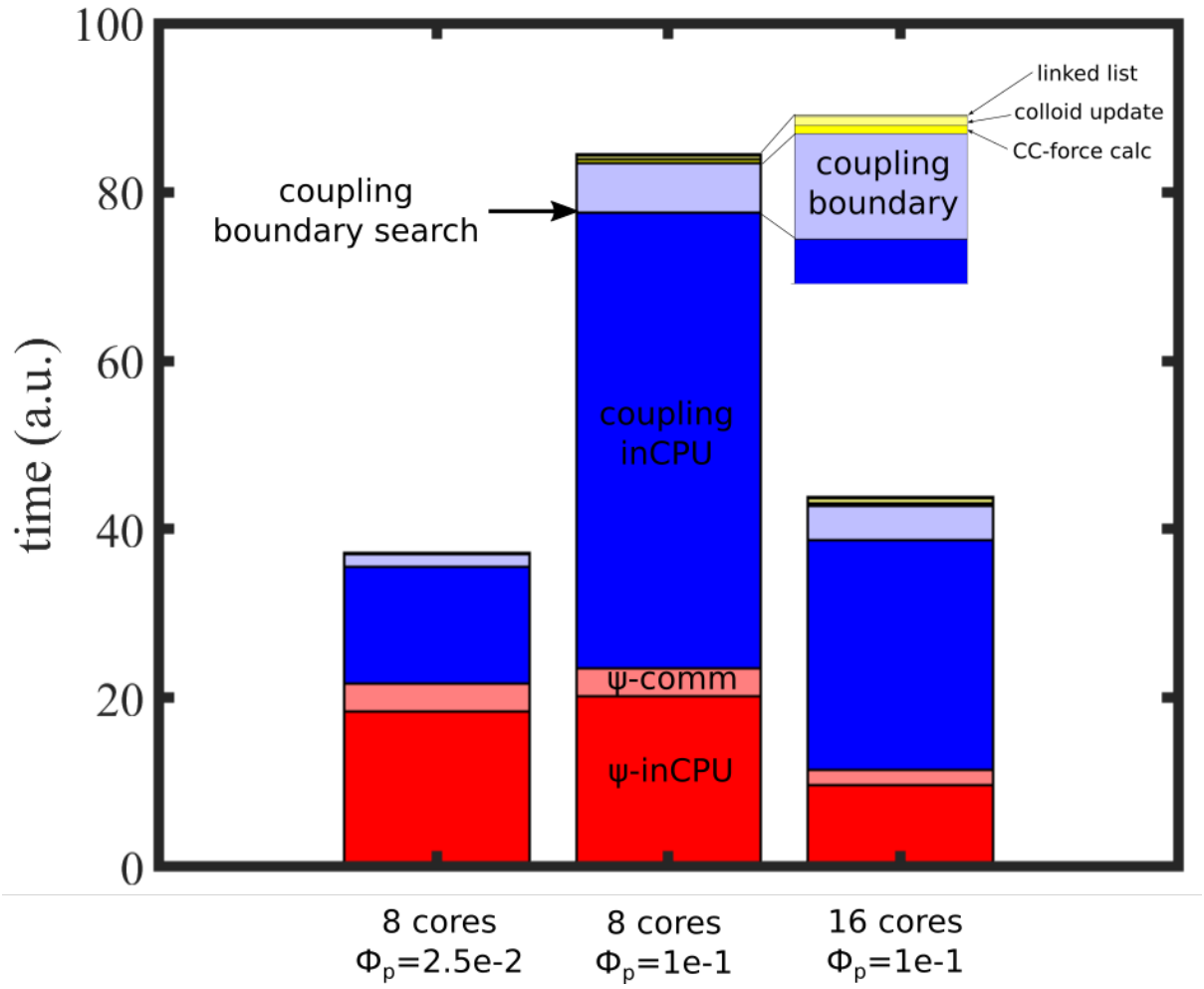

Figure S7: Bar graph of the computational time for three different systems in a $V = 256^3$ box: (left) 8 cores with concentration $\phi_p = 2.5 \times 10^{-2}$ of NPs, (middle) 8 codes with $\phi_p = 1 \times 10^{-1}$ and (right) 16 codes with $\phi_p = 1 \times 10^{-1}$. The bar height is the time spent in each section of the code, in arbitrary units. The colouring reflects the section type: Dark red for the in-processor calculation and transparent red for the halo exchanges. Blue for coupling calculations, dark for coupling within local processor and two transparent for searching and calculation of boundary particles. Finally, yellow for colloidal calculations.

- Polymer
  - **In-processor** - Laplacian calculations (for ψ and μ), adding chemical potential and update of ψ
  - ψ **communication** - halo exchange for ψ and μ.
- Coupling
  - **In-processor** - calculating forces and torques for particles within the processor's domain.
  - **Boundary search** - Search for particles within the boundary of local processor, *ie*, looking for particles in neighboring processors that may be overlapping local processor.
  - **Boundary particles calculation** - calculating coupling for particles which are in the boundary
- Colloid
  - **Force Calculation** - calculating colloid-colloid forces
  - **Update colloid** - Update colloidal positions and communicate colloidal information between processors.
  - **Update linked list** - Update the linked list.

Figure S7 further confirms the main conclusions in figure 3 and 4 in the main text: the computational effort to perform coupling calculations can be comparable and much higher than the purely polymeric part. This is not the case for the colloid part (colloid-colloid forces calculations and updates), which are always subdominant. It should be noted that the simulation parameters are particularly chosen to reflect a high effort in the colloidal part, with 400000 particles in the system for the $\phi_p = 1 \times 10^{-1}$ case. The smaller concentration $\phi_p = 2.5 \times 10^{-2}$ leads to a smaller coupling contribution and a totally negligible colloidal part, in the left bar. One can notice that the polymeric part remains mostly equal in the left and middle bar as the polymer conditions are equal. On the other hand, in the right bar the number of cores is doubled, which reflects in an approximate reduction in half in the computational time. The computational time to perform the halo exchanges in the polymeric part remains the largest communication effort in the simulation, which again should be emphasised is considering a particularly large number of particles in the middle and right cases. The coupling boundary search is found to be completely negligible compared with the two actual coupling calculations: in-processor and boundary particle calculations.

The bottleneck of the computational time clearly depends on the concentration of particles in the system. If the concentration is low, the calculations of the Laplacians are the main bottleneck of the simulation, as the halo exchange is shown to be subdominant. If higher number of processors are used, the performance of the code will decrease and the use of additional processors will not be beneficial, as the ψ and μ communications become significant, as shown in figure 1 in the main text (this is clearly dependent on the system size). For considerable concentrations the coupling contribution is not negligible compared to the polymeric part, which can become the main bottleneck for the computational effort of the simulation. The coupling calculation (both in-processor and boundary particles) are clearly the heaviest part of the coupling part.

References

[1] A. Fanfarillo, T. Burnus, V. Cardellini, S. Filippone, D. Nagle and D. Rouson, Proceedings of the 8th International Conference on Partitioned Global Address Space Programming Models, 2014, pp. 1–11.

[2] X. Guo, M. Pinna and A. V. Zvelindovsky, *Macromolecular Theory and Simulations*, 2007, **16**, 779–784.