VI International Conference on Particle-based Methods – Fundamentals and Applications
PARTICLES 2019
E. Oñate, M. Bischoff, D.R.J. Owen, P. Wriggers & T. Zohdi (Eds)

MULTI-LEVEL LOAD BALANCING FOR PARALLEL PARTICLE SIMULATIONS

Godehard Sutmann[1,2]

[1] Jülich Supercomputing Centre (JSC), Institute for Advanced Simulation (IAS),
Forschungszentrum Jülich (JSC), D-52425 Jülich, Germany
e-mail: g.sutmann@fz-juelich.de http://www.fz-juelich.de/ias/jsc/staff/g_sutmann

[2] ICAMS, Ruhr-University Bochum, D-44801 Bochum, Germany

Key words: Parallel Computing, Particle Methods, Load Balancing, Multi-Level Methods, Multigrid

Abstract. Ideas from multi-level relaxation methods are combined with load balancing techniques to achieve a convergence acceleration for a homogeneous work load distribution over a given set of processors when the underlying work function is inhomogeneously distributed in space. The algorithm is based on an orthogonal recursive bisection approach which is evaluated via a hierarchically refined coarse integration. The method only requires a minimal information transfer across processors during the tree traversal steps. It is described of how to partition the system of processors to geometrical space, when global information is needed for the spatial tesselation.

1 INTRODUCTION

Load imbalance is a common problem for parallel applications, which often arises when work load distributions are inhomogeneously distributed in the global system setup, or may occur when local inhomogeneities in the work density show up when increasing the number of processors. In fact, for many parallel applications which rely on domain decomposition as a parallel strategy, the processors define a spatial discretisation. Increasing the processor count, the spatial resolution is increased, which resolves density differences (which are often related to differences in work distribution) on a finer scale which consequently lead to runtime differences on the individual processors. Since differences in work load do lead to reduced parallel efficiency, various methods for an improved load balance between the processors have been proposed [1, 2, 3, 4]. In the present article, problems related to particle distributions or mesh vertices are considered. Since vertices might be considered as a kind of abstract particle in the sense that they have properties and relations to their neighbourhood we will use in the present article the notion of *particle*, which might be understood in a more general context.

The problem of load balancing has to consider redistribution of work between processors in such a way that all processors are actively working for a period of time and reaching

either the end of the program or a synchronization point at almost the same time, so that overhead- and waiting-times are minimized. This either includes active redistribution of work by sending tasks over the network, which reside on the other processors, or by, e.g., work stealing procedures, which includes forth and back communication between processors for both task distribution and gathering of results. In the present article we focus on redistribution of work load such that degrees of freedom (e.g. particles, vertices, etc.) are communicated between processors as a result of redefining the geometry of the domains. The basic principle is therefore a modification of the shape and size of domains such that local work $W_i^{(n+1)}$ is increased, if $W_i^{(n)} < \langle W \rangle_P$, where $W_i^{(n)}$ is the total work on processor $i \in [0, P-1]$, P the number of processors and $\langle W \rangle_P$ the average work of a processor. In analogy, $W_j^{(n+1)}$ is reduced, if $W_j^{(n)} > \langle W \rangle_P$. The goal is that shapes and sizes of domains are adjusted in such a way that $W_i^{(n)} = \langle W \rangle_P \pm \delta W_i$, where δW_i is a tolerable difference on each processor from the optimal load, for which holds $\sum_{i=0}^{P-1} \delta W_i = 0$.

In that sense, the problem of balancing the load between the processors has different levels of complexity: (i) how to properly define the load; (ii) which is the proper shape and size of a domain; (iii) how to control or minimise additional costs, e.g. increased number of communication partners; (iv) how to minimise the procedure of work redistribution, domain size and shape, e.g., with minimal communication costs between the global processor grid. In the present article we will mainly focus on issue (iv), which is related to the computational costs introduced by the chosen load balancing procedure and which should, of course, be much smaller than the computational overhead which is related to the work imbalance, i.e. the overhead which exists without any application of load balancing. Concerning issue (i) we will assume a properly chosen function, which characterises the work and which is properly measurable. In practice, this could be, e.g., the number of particles on each processor, the number of interactions, the total time of interactions or the wall clock time of one full time step. For convenience we choose for discrete systems the number of particles, N_i, on each processor. Concerning issue (ii) there were a number of different approaches discussed in the literature, including (a) orthogonal shapes in the case of orthogonal recursive bisection method [5] or tensor product method [6]; (b) irregular cells in the case of, e.g., Voronoi tesselation [7] or graph partitioning [8]; (c) distorted meshes with conserved topology [9, 10]. These methods differ mainly in how the work is redistributed and which constraints for the individual domains are considered. For most proposed methods the computational costs on the domain are considered, but communication between neighbour domains is neglected or, at least, not minimised. The latter issue usually leads to a coupled problem, which on the one hand increases complexity of the minimisation procedure and, on the other hand, leads sometimes to a non-smooth objective function since communication partners, and therefore also the communication overhead, may change discontinuously when increasing or reducing communication partners during minimisation. This communication overhead is related to the work on each domain *after* the load balancing step is finished. But the load balancing itself needs information which invokes communication between processors and in the worst case the required information has to be communicated between *all* processors,

which especially leads to a big overhead for a large number of processors (i.e. those cases where load balancing is often crucial for a good parallel efficiency). This overhead might get crucial, especially when the load balancing has to be performed frequently in the simulation. This is a common problem for dynamic systems with high density contrasts or regions, which temporally does not contain any load. In such cases, the domain sizes and shapes might be reconstructed frequently, e.g. as it is for graph partitioning methods, or might be smoothly adjusted to a new (smoothly varying in time) work load change, which allows for iterative schemes, following the work load in time. This, however, has an important requirement, namely that the change in work load has a slower relaxation time than the load balancing scheme. Otherwise the load balancing method would lag behind the work load, not reaching an equal distribution of load.

The current work does not focus on a new formulation of an objective function for minimising computational work and communication overhead, but on an approach which minimises the communication volume during the load balancing step for the case when global information exchange is necessary within the processor mesh. Necessary global communication in load balancing steps is performed on a hierarchical tree with minimal information exchange. We will focus on the class of *Orthogonal Recursive Bisection* (ORB), which is by itself a hierarchical scheme.

Since the work load on a domain is determined by the local degrees of freedom on each domain, also information about the distribution of work on the domain is necessary, if it cannot be mapped to a simple mask, which could be communicated over the network. Therefore, we consider here an approach, which adapts elements from multigrid method [11], which uses the property that inhomogeneities on small scales are smoothed on a coarser scale. Since the number of degrees of freedom might get large (e.g. $N > 10^9$) on high processor counts (e.g. $P > 10^5$), the present approach is formulated in terms of reduced properties, i.e. only local work densities and domain coordinates are required to balance the load quite efficiently, even for very inhomogeneous work distributions, i.e., computational degrees of freedom, e.g. particles, are only exchanged / redistributed on the highest tree level and not explicitly communicated along the tree.

The article is organised as follows: In Sec. 2 the method is introduced from a formal point of view. In Sec. 3 results are shown for various test cases, which include exact function descriptions (in order to consider convergence properties) and sample systems, consisting of discrete particle distributions.

2 METHOD

2.1 Multi-Level Description of Workload

The goal of the formal characterization of work load is to describe it as hierarchical subdivision, where domains on level $l+1$ in the hierarchical description are constructed by a bisection of domains of a given level l. In this sense, the coarsest level, $l = 0$, consists of the whole system, whereas the finest level $l = L$ consists of 2^L sub-domains. The original computational domains, which are administered by 2^L processes are the target regions for balancing the work load, such that each process has to fulfil the same amount

of computational work. Therefore, the target work load for each process can be written as the average work on a given level, i.e.

$$\langle W\rangle_l = 2^{-l}\, W \tag{1}$$

For the highest level L the target distribution of work is

$$\begin{aligned}\langle W\rangle_L &= 2^{-L}\, W &(2)\\ &= \frac{1}{P}\sum_{m=0}^{2^L-1} W_{L,m}^{(n)} &(3)\\ &= \frac{1}{P}\, W &(4)\end{aligned}$$

where $W_{L,m}^{(n)}$ is the work on level L on preocess m during multi-level iteration n. Since this represents the total sum over all partial work distributions, this relation holds for each iteration n. Splitting a domain into two sub-domains can therefore be described as

$$W_{l+1,2k}^{(n)} = \langle W\rangle_l + \delta W_{l+1,2k}^{(n)} \tag{5}$$

$$W_{l+1,2k+1}^{(n)} = \langle W\rangle_l + \delta W_{l+1,2k+1}^{(n)} \tag{6}$$

where

$$\begin{aligned}\delta W_{l+1,2k}^{(n)} &= \int_0^{x_{1/2}} dx\, \rho(x\,|\,x\in\Omega_{l+1,2k}^{(n)}) - \int_0^{\delta x_{l+1,2k}^{(n)}} dx\, \hat\rho_{l+1,2k}^{(n)}(x) &(7)\\ &= \frac{1}{2}\left[W_{l+1,2k}^{(n)} + W_{l+1,2k+1}^{(n)}\right] - \sum_{\{\ell\}} V_{l+1,\ell}^{(n)}(\{\ell\})\,\hat\rho_{l+1,\ell}^{(n)}(\{\ell\}) &(8)\end{aligned}$$

with the set

$$\{\ell\}_{l+1,2k}^{(n)} = \{\ell\,|\,\Omega_{L,\ell}^{(n)} \cap \Omega_{l+1,2k}^{(n)} \notin \{\emptyset\}\} \tag{9}$$

meaning that all volume and density contributions from the highest level L are sampled onto the geometric region $2k$ with finite intersection. $\delta x_{l+1,2k}^{(n)}$ is an approximation for the location of the division point (in 2d division line, in 3d division plane) which subdivides a domain on level l into two sub-domains on level $l+1$ with approximately equal work load. In this stage, $\delta x_{l+1,2k}^{(n)}$, can only be computed approximately, since it is based on an average underlying work density, which does not contain any information about inhomogeneities in work distribution on a given process, i.e., the density $\hat\rho_{l+1,2k}^{(n)}(x)$ is constructed as a coarse grain approximation on the highest level L, which is iteratively refined during the multi-level procedure. Within multi-level cycle n, a density map of the system on level l is constructed by concatenating the average densities within the domain geometries, $\Omega_{l,k}^{(n)}$. On the lowest level $l=0$ this can be expressed as the complete map

$$\hat\rho_{0,0}^{(n)}(x) = \bigcup_{i=0}^{2^L-1} \left\langle \frac{W_{L,i}^{(n)}}{V_{L,i}^{(n)}} \right\rangle_{\Omega_{L,i}^{(n)}} \tag{10}$$

This results in a step or plateau function, where constant values are assigned to the density within the domain boarders of each domain on level $l = L$.

This approach offers a low communication approach to the implementation, since only the domain geometries and the local work has to be transferred along the tree traversal. The communication part is described in more detail in the next sections.

2.2 Tree Down-Traversal: Merging domains between two levels

The amount of communicated data down the tree is reduced to $\#_{l,i}^{(n)} = (2 \times d + 1) \times 8 \times |\{\ell\}_{l,2k}^{(n)}|$ bytes, where d is the dimension of the parallel sub-decomposition and which essentially contains the lower-left- and upper-right-corner of the domain (given that it is an orthogonal decomposition) and the work function value. To describe the basic procedure, we consider for the initial geometry a non-staggered, orthogonal decomposition of domains in 3-dimensional space on level L, which means that each domain has 6 neighbours, separated by domain interfaces. Here, we consider tilings in each cartesian direction which contain domains of multiples of 2, i.e. $\Omega = \mathcal{D}_x \otimes \mathcal{D}_y \otimes \mathcal{D}_z$, where $|\mathcal{D}_\alpha| = 2^{l_\alpha}$ (note that other tilings of arbitrary number of domains, including prime numbers, are possible and will be discussed elsewhere [12]). The procedure of merging is then straightforward. We describe each level l as combination of levels in cartesian directions, i.e.

$$l = l_x + l_y + l_z \tag{11}$$

If we consider a given level l, the merging of two adjacent sub-domains is performed in one of the cartesian directions α, which reduces the next level by one through

$$l - 1 = (l_\alpha - 1) + l_\beta + l_\gamma \tag{12}$$

where α, β, γ is any permutation of x, y, z. If we denote 2 adjacent index pairs in the multi-level decomposition as $([i_l]_\alpha, [i_l + 1]_\alpha)$, which are subject to be merged in direction α, then we can write

$$\begin{aligned}
i_x \in [0, 2^{l_x - 1} - 1] \quad i_y \in [0, 2^{l_y} - 1] \quad i_z \in [0, 2^{l_z} - 1] & \\
[i_l]_x &= 2^{L_x - l_x}\, 2 i_x + 2^{L_x + L_y - l_y}\, i_y + 2^{L_x + L_y + L_z - l_z}\, i_z && (13) \\
[i_l + 1]_x &= 2^{L_x - l_x}\, (2 i_x + 1) + 2^{L_x + L_y - l_y}\, i_{l_y} + 2^{L_x + L_y + L_z - l_z}\, i_z && (14)
\end{aligned}$$

$$\begin{aligned}
i_x \in [0, 2^{l_x} - 1] \quad i_y \in [0, 2^{l_y - 1} - 1] \quad i_z \in [0, 2^{l_z} - 1] & \\
[i_l]_y &= 2^{L_x - l_x}\, i_x + 2^{L_x + L_y - l_y}\, 2 i_y + 2^{L_x + L_y + L_z - l_z}\, i_z && (15) \\
[i_l + 1]_y &= 2^{L_x - l_x} + 2^{L_x + L_y - l_y}\, (2 i_{l_y} + 1) + 2^{L_x + L_y + L_z - l_z}\, i_z && (16)
\end{aligned}$$

$$\begin{aligned}
i_x \in [0, 2^{l_x} - 1] \quad i_y \in [0, 2^{l_y} - 1] \quad i_z \in [0, 2^{l_z - 1} - 1] & \\
[i_l]_z &= 2^{L_x - l_x}\, i_x + 2^{L_x + L_y - l_y}\, i_y + 2^{L_x + L_y + L_z - l_z}\, 2 i_z && (17) \\
[i_l + 1]_z &= 2^{L_x - l_x} + 2^{L_x + L_y - l_y}\, i_{l_y} + 2^{L_x + L_y + L_z - l_z}\, (2 i_z + 1) && (18)
\end{aligned}$$

The enumeration of the next lower level is then accordingly $[i_{l-1}]_\alpha = [i_l]_\alpha$. Moving down the tree and merging domains in direction α, the separating surface between domains is removed and it is $\Omega_{l-1,i} = \Omega_{l,i_1} \cup \Omega_{l,i_2}$, with $i = [i_{l-1}]_\alpha$, $i_1 = [i_l]_\alpha$, $i_2 = [i_l + 1]_\alpha$.

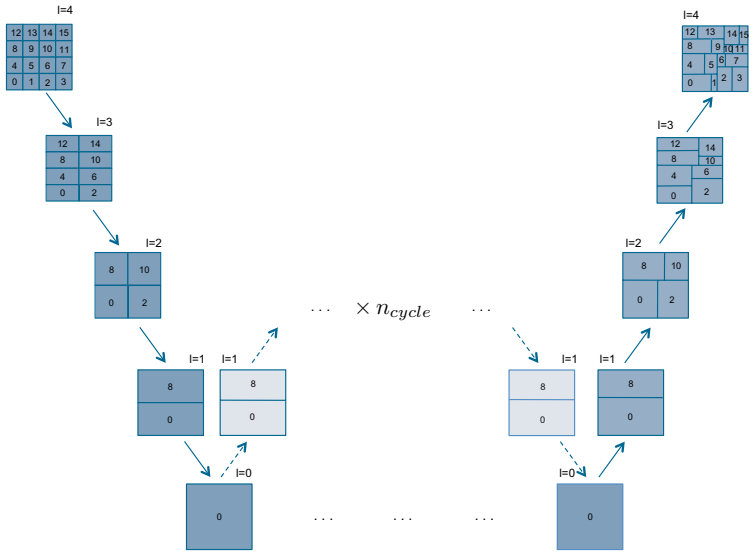

Figure 1: Schematics of the parallel implementation of the multi-level load balancing scheme. On lower levels, only processors with even indices are active.

Merging implies that the domain with index $[i_l]_\alpha$ is the *master domain*, which combines information of two sub-domains and takes part in the merging process on the next lower level. To have all information available, a communication from $[i_l+1]_\alpha$ to $[i_l]_\alpha$ has to be performed which contains the lower left, upper right coordinates of the domain and its total work. On this stage, also the coarse density field of domains on level L, which overlaps with domain $[i_l+1]_\alpha$ is communicated to ensure that on each level in the up-traversal later on the computation of the separating surface can be accomplished.

2.3 Tree Up-Traversal: Splitting domains on a level

The density field of work which has to be integrated to find the proper coordinate for domain splitting on level l, is fixed on the highest level L in the tree, i.e., it contains the average work within the geometry of individual domains in a given iteration step k. For simplicity we first consider a 1-dimensional case. This means that at lower levels, several sub-volumes from the highest level contribute to the work density field, which means that several integrals have to be evaluated to find the location of $x_{1/2} = \{x : P(x) = 1/2\}$, where $P(x)$ is the cumulative distribution function of work on a domain. From Fig. 1 it is understood that $x_{1/2}$ can be found by the geometric consideration

$$x_{1/2} = x_{k-1} + \frac{\frac{1}{2} - P(x_{k-1})}{W_k}(x_k - x_{k-1}) \tag{19}$$

$$= x_{k-1} + \frac{1}{2}\frac{1 - 2P(x_{k-1})}{P(x_k) - P(x_k - 1)}(x_k - x_{k-1}) \tag{20}$$

Here k is the domain, for which holds

$$k = \left\{ k : P(x_{k-1}) < \frac{1}{2} \wedge P(x_k) > \frac{1}{2} \right\} \tag{21}$$

In a multi-dimensional setting this is a bit more involved. The splitting of a domain is performed in a given direction $\mathbf{n}_\alpha$ by an intersection plane, which is orthogonal to the splitting direction. The splitting plane is then introduced at the relative position $r_\alpha(1/2)$, for which the integral holds

$$\int_{r_{\alpha,0}}^{r_\alpha(1/2)} dr_\alpha \int_{r_{\beta,0}}^{r_{\beta,L}} dr_\beta \int_{r_{\gamma,0}}^{r_{\gamma,L}} dr_\gamma \, \rho(\mathbf{r}) = \frac{1}{2} W \tag{22}$$

In a discrete set of domains, where the density might change abruptly, depending on the work distribution over the processors, one can write for the total work on the domain

$$W = \sum_{i_\alpha=0}^{p_\alpha-1} \int_{r_{\alpha,I[i_\alpha]}}^{r_{\alpha,I[i_\alpha+1]}} dr_\alpha \sum_{i_\beta=0}^{p_\beta-1} \int_{r_{\beta,i_\beta}}^{r_{\beta,i_\beta+1}} dr_\beta \int_{r_{\gamma,0}}^{r_{\gamma,L}} dr_\gamma \sum_{i_\gamma=0}^{p_\gamma-1} \int_{r_{\gamma,i_\gamma}}^{r_{\gamma,i_\gamma+1}} dr_\gamma \, \rho_{i_\alpha,i_\beta,i_\gamma} \tag{23}$$

In this formulation, $I[i]$ is an ordered set as such that the domains of the highest level, intersecting with the domain on the coarse level l, are sorted according to their lower boarder in direction $\mathbf{n}_\alpha$. Since the densities are approximated as constant over the domains at the highest level, the integrals can be computed exactly. Selecting for the splitting e.g. the z-direction can be formulated as a sum of sub-domains, $W_l^<$, located completely below $z_{1/2}$ and those, $\delta W_l^{1/2}$, which are cut by the xy-plane cutting the z-axis at $z_{1/2}$

$$W_l^{1/2} = W_l^< + \delta W_l^{1/2} \tag{24}$$

where

$$W_l^< = \sum_{p \in \mathcal{P}_z} (x_{p,1} - x_{p,0}) (y_{p,1} - y_{p,0}) (z_{p,1} - z_{p,0}) \rho_p \tag{25}$$

and

$$\delta W_l^{1/2} = \sum_{p \in \mathcal{P}_{1/2}} (x_{p,1} - x_{p,0}) (y_{p,1} - y_{p,0}) (z_{1/2} - z_{p,0}) \rho_p \tag{26}$$

which leads to the following expression for $z_{1/2}$

$$z_{1/2} = \frac{W_l^{1/2} - W_l^< + \sum_{p \in \mathcal{P}_{1/2}} (x_{p,1} - x_{p,0}) (y_{p,1} - y_{p,0}) z_{p,0} \rho_p}{\sum_{p \in \mathcal{P}_{1/2}} (x_{p,1} - x_{p,0}) (y_{p,1} - y_{p,0}) \rho_p} \tag{27}$$

Here, the sets $\mathcal{P}_z$ and $\mathcal{P}_{1/2}$ contains all subdomains p on the highest level L for which

$$\mathcal{P}_z = \left\{ \{p\} \,|\, \Omega_{L,p} \cup \Omega_l \notin \{\emptyset\} \wedge z_{p,1} > z_{1/2} \wedge z_{p,1} \leq z_{1/2} \right\} \tag{28}$$

$$\mathcal{P}_{1/2} = \left\{ \{p\} \,|\, \Omega_{L,p} \cup \Omega_l \notin \{\emptyset\} \wedge z_{p,0} < z_{1/2} \wedge z_{p,1} > z_{1/2} \right\} \tag{29}$$

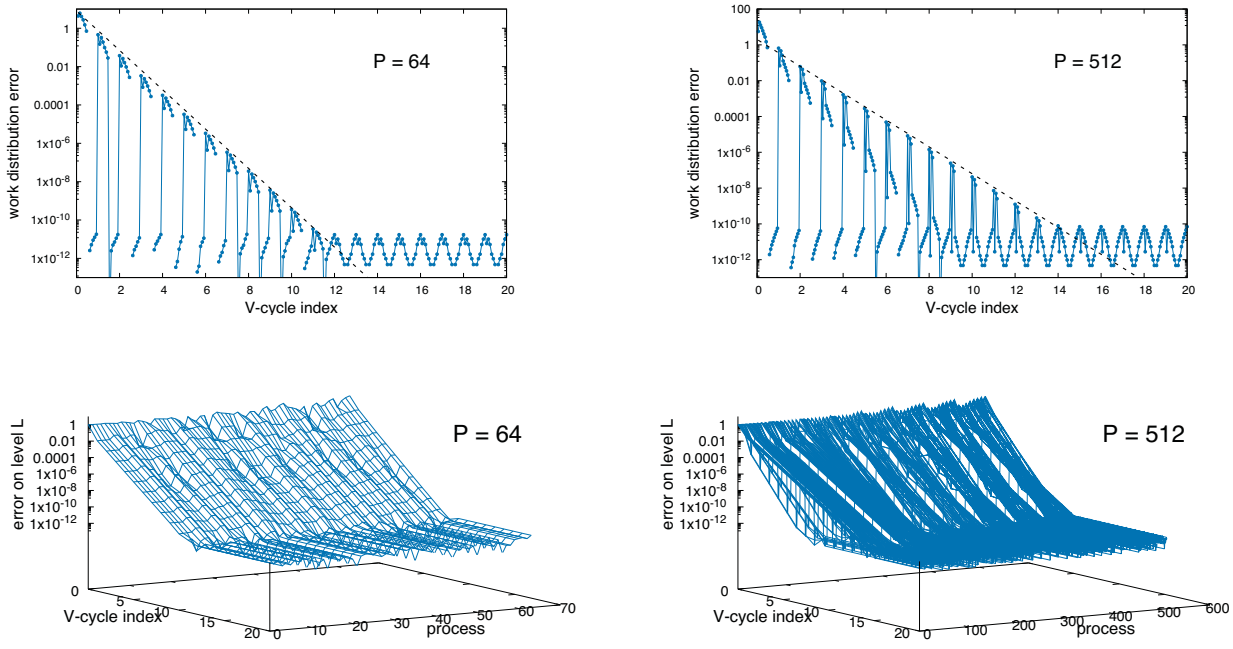

Figure 2: Results for Model 1, the trilinear model with an error threshold $\epsilon_{1/2} = 10^{-12}$ for the detection of the intersection plane during the up-traversal step of the tree walk. Compared are results for $P = 64$ and $P = 512$ domains, which are partitioned applying a multi-level V-cycle. The longer relaxation time for larger number of domains is in part due to a more inhomogeneous convergence behaviour of some domains.

Therefore, in order to split the domain Ω_l, requires a map of the underlying Ω_L distribution. Once this map is known on the coarse domain Ω_l, the cutting plane for the generation of two sub-domains on level Ω_{l+1} can be done exactly according to the underlying density distribution.

It is noted that between multi-level iteration cycles the exact work distribution does not change, since the sources of work, i.e. particles or mesh points are not moved or re-weighted. However, after each multi-level cycle, the domain geometries on level L are adjusted and accordingly the work in each domain is changed, i.e., also the work density for each domain is modified in general. As a consequence, the underlying domain map and also the density tesselation is changed, leading to an iteratively converging partitioning of the domains on level L.

2.4 Generation of the underlying density map

In each multi-level iteration step, the density map has to be known by all levels $l \in \{0, L\}$. The multi-level cycle starts from the finest level L, corresponding to the computational domain of each processor. On that level, the work per domain is explicitly known and transferred as a characteristic quantity to the load balancing method. Going down the tree to the coarsest level $l = 0$, means to merge two adjacent domains and com-

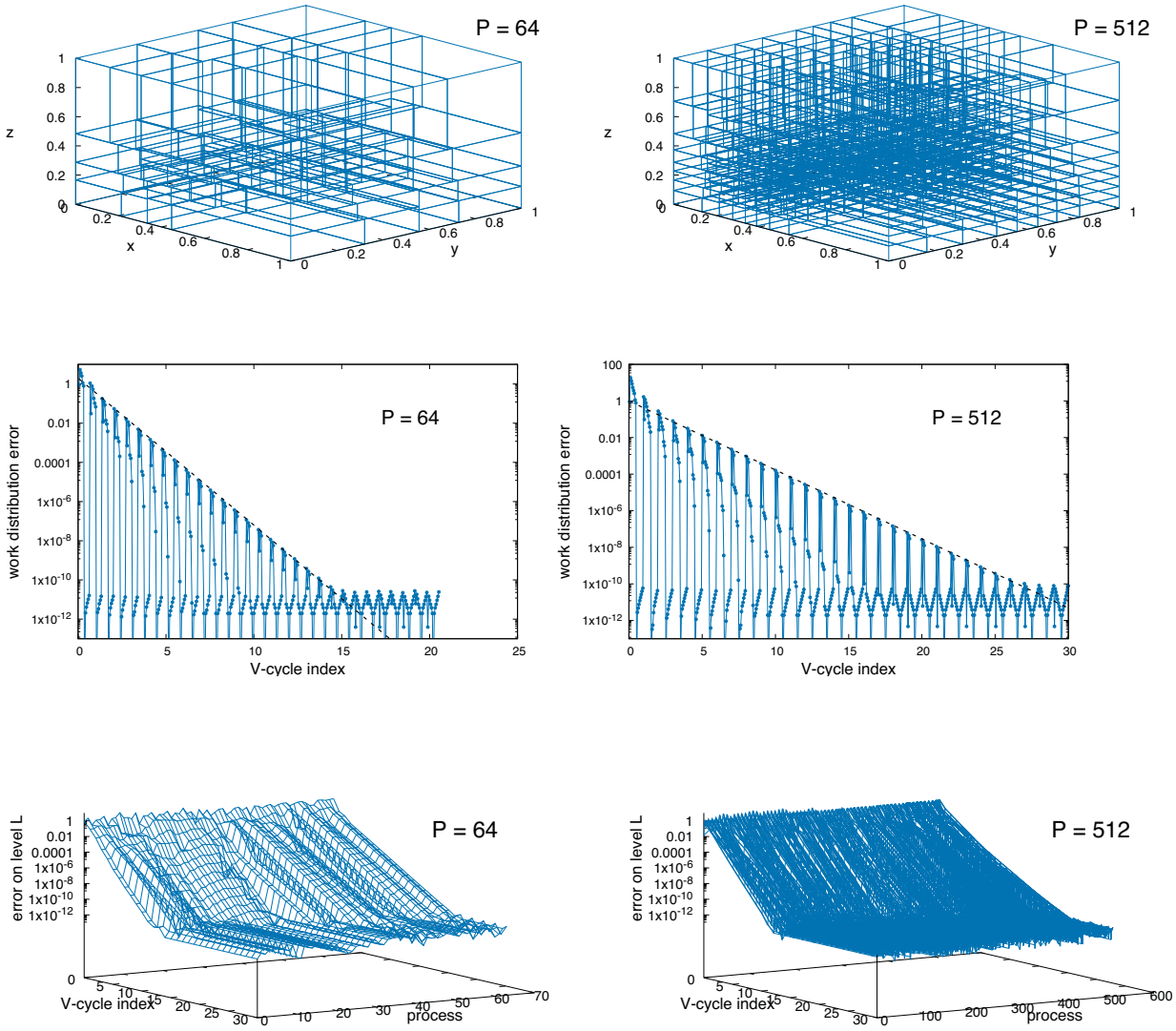

Figure 3: Results for Model 2, the Gaussian superposition model with an error threshold $\epsilon_{1/2} = 10^{-12}$ for the detection of the intersection plane during the up-traversal step of the tree walk. As in Fig. 2, results are compared for $P = 64$ and $P = 512$ domains, which are partitioned applying a multi-level V-cycle. For this case, there is also some inhomogeneous convergence behaviour observed for some domains. As an example for the result of the partitioning, the domain structure on the highest level, L, is shown (top).

bine their individual work, i.e. $W_{p,l} = W_{2p,l+1} + W_{2p+1,l+1}$. Since on the lower level, we do not want to introduce coarser density descriptions, we keep the context of the density distribution, i.e. $L-1$, there will be two distinct regions in the boarder of Ω_L domains, which have different densities. To distribute the density distribution over the tree, the processor containing the region $\Omega_{2p+1,l+1}$ communicates its density to processor $2p$, which will contain $\Omega_{p,l}$ on the next coarser level. This can be continued along the walk down the tree, so that on the lowest level the remaining processor p knows about the complete density map of the system. Having this information available, it is possible to use the formalism described above to construct a numerically exact subdivision with domains of equal work load.

3 RESULTS

Model 1: Trilinear function: The density field is defined as

$$\rho_w(x,y,z) = x\,y\,z \tag{30}$$

resulting in the total work on level L on domain k, $\Omega_{L,k}$

$$W_{\Omega_{L,k}} = \frac{1}{8}\,(x_{1,k}^2 - x_{0,k}^2)\,(y_{1,k}^2 - y_{0,k}^2)\,(z_{1,k}^2 - z_{0,k}^2) \tag{31}$$

Model 2: Gaussian superposition: The density field is defined as

$$\rho_w(x,y,z) = \frac{1}{n_g}\sum_{j=1}^{n_g}\prod_{\alpha=x,y,z}\frac{1}{\sqrt{2\pi}\sigma_{\alpha,j}}\,e^{-(\alpha-\mu_{\alpha,j})^2/2\sigma_{\alpha,j}^2} \tag{32}$$

resulting in the total work on level L on domain k, $\Omega_{L,k}$

$$W_{\Omega_{L,k}} = \frac{1}{n_g}\sum_{j=1}^{n_g}\prod_{\alpha=x,y,z}\mathrm{erf}\left(-\frac{\alpha_{0,k}-\mu_{\alpha,j}}{\sqrt{2}\sigma_{\alpha,j}}\right) - \mathrm{erf}\left(-\frac{\alpha_{1,k}-\mu_{\alpha,j}}{\sqrt{2}\sigma_{\alpha,j}}\right) \tag{33}$$

Model 3: Gaussian particle distribution: The probability for finding a particle in the volume $dx\,dy\,dz$ is given by

$$p(x,y,z)dxdydz = p_g(x)p_g(y)p_g(z)dxdydz \tag{34}$$

with

$$p_g(\alpha) = \frac{e^{(\alpha-\mu_\alpha)^2/2\sigma_\alpha^2}}{\int_{L_\alpha} d\alpha\, e^{(\alpha-\mu_\alpha)^2/2\sigma_\alpha^2}} \tag{35}$$

The work function is the result of the underlying discrete distribution of N particles, so that the total work on the highest level L, i.e., on each process is given by

$$W_{\Omega_{L,k}} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}(\{i\,|\,\alpha_{0,k} < \alpha_i < \alpha_{1,k}, \alpha = x,y,z\}) \tag{36}$$

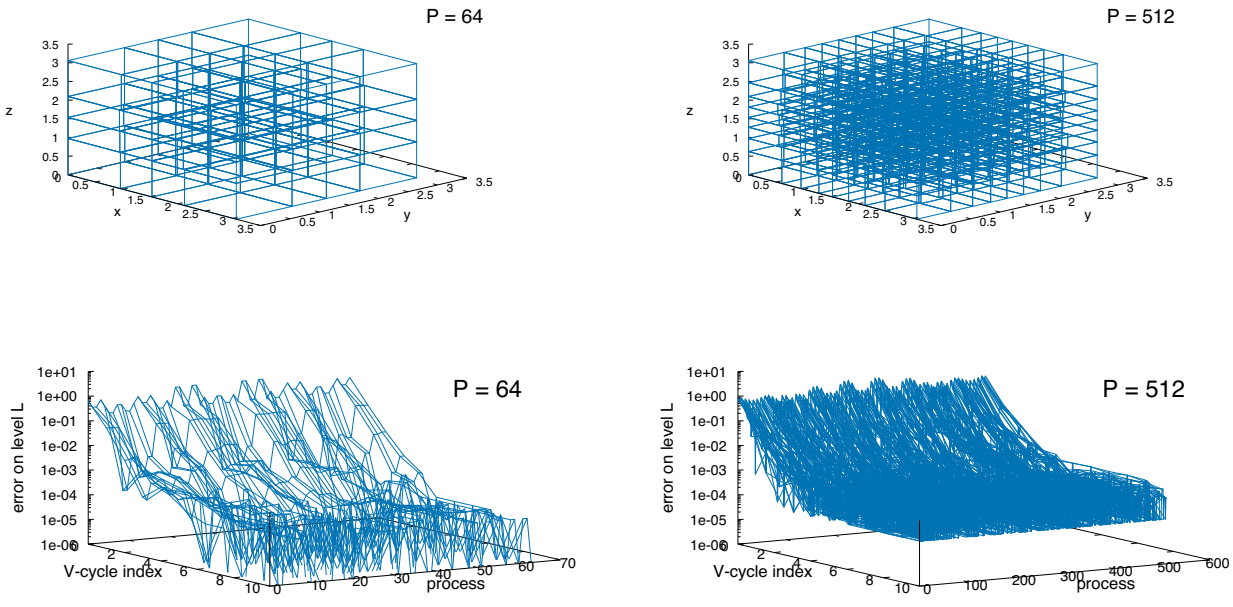

Figure 4: Results for Model 3, the Gaussian density for particle distribution function with an error threshold $\epsilon_{1/2} = 10^{-12}$ for the detection of the intersection plane during the up-traversal step of the tree walk. Compared are results for $P = 64$ and $P = 512$ domains, which are partitioned applying a multi-level V-cycle. Obviously, due to discreteness of particles, the error of work distribution does not decrease to the prescribed error threshold.

where $\mathbb{1}(\{i \,|\, .\})$ is the indicator function, which is 1 if the criteria for α_i are met and 0 otherwise. Therefore, we consider the number of particles on each process as a measure for the work. In other scenarios, the number of interactions between particles or the wall clock time of a time step could be chosen as work function on a process.

Each load balancing experiment starts on a regular cartesian mesh. The errors are computed as local errors of process i

$$\epsilon_i = \left| 1 - \frac{W_i}{\langle W \rangle} \right| \tag{37}$$

and global errors as

$$\epsilon = \sqrt{\frac{1}{P} \sum_{i=0}^{P-1} \epsilon_i^2} \tag{38}$$

All three models have been tested for $P = 64$ and $P = 512$ processes. Model 1 shows a quite smooth relaxation behaviour to the prescribed error threshold for $P = 4^3 = 64$ processes. For the case of $P = 8^3 = 512$ processes, Fig. 2(right) shows that there are different modes present with different relaxation times. It is not yet clear whether this is an inherent system property, or whether a numerical condition is responsible for this

behaviour which might be lifted by a correction technique. Qualitatively, the same behaviour is observed for Model 2, where for both numbers of processes this inhomogeneous relaxation can be observed. Nevertheless, although there might be a delay in relaxation (which might be possible to minimise) the system is converging to a stable final configuration of domain distribution, which is shown in Fig. 3(top). A stronger effect of this behaviour is observed for the discrete particle system. It is understood that the exact work distribution can ideally only be found if the number of particles is a multiple of the number of processes. If this is not the case there is an inherent discretisation error, which brings the error of work distribution $\delta\epsilon = \mathcal{O}(P/N)$, which is likely to reach the percentage range. For the cases studied, the observed local errors in the relaxed state are in the range of $\epsilon_i < 10^{-3}$ for $P = 64$ and for $P = 512$ they are found to be in the range of $\epsilon_i < 5 \times 10^{-3}$, which is still below the percentage threshold.

4 CONCLUSIONS

We have developed a mathematical description of a multi-level orthogonal recursive bisection method for the balanced work distribution on parallel processes in particle systems. The method uses a minimal basis for information exchange between processors, i.e. communicating information about local domain geometries and a scalar work descriptor of the underlying finest domain splitting, i.e. the average work on each processor on the highest level L in the tree. This density field is iteratively refined by correcting the domain geometries on each intermediate level l during the up-traversal step, resulting in an improved work density description on the highest level, which in turn is applied to the domain-splitting step in the next up-traversal step. This procedure resembles the classical geometric multigrid scheme and is implemented in analogy as a sequence of V-cycles [11]. For three test cases it has been shown that a domain partitioning can be efficiently computed, which could reduce the work imbalance error in a few steps by several orders of magnitude. For continuous test cases it could be shown that the global error can even be reduced to $\epsilon < 10^{-10}$. For discrete systems, smoothness of the problem is lacking and the error is saturating at a level of $\epsilon < 10^{-3}$, which is certainly satisfying for any realistic simulation scenario, since dynamics of the underlying particle system is likely to increase load imbalance quickly into the low percentage range. Analysing the relaxation behaviour of each domain, it could be observed that the final accuracy of work imbalance is nearly the same for all processes, but that the relaxation is non-homogeneous, i.e. relaxation time is not unique for all processes. This might be related to the inhomogeneous work density distribution, or the convergence of the coarse density field to the *true* density map. To investigate the origin, will be a matter of investigation. Also it could be observed that the final error of the work distribution shows a slight oscillatory behaviour for the case of the discrete particle system. This is a signature of oscillations of the domain boarder geometries, i.e. no fix-point has been found for the domain geometry. Whether the system is trapped in a local minimum, or an incommensurability of equal work distribution has been occurred due to the discrete nature and, consequently, possible discretisation error, will be investigated in a future work. Furthermore, a more detailed analysis of the convergence behaviour will be conducted and possible criteria for not finding the absolute

minimum of the work distribution error will be discussed. It is a matter of the underlying system dynamics which contains the information of how often the load balancing procedure has to be called by the program. Experimental studies will be conducted in future to understand the interplay between system dynamics, gain in load balancing and penalty costs of calling the load balancing routine.

REFERENCES

[1] G. Karypis and V. Kumar. Multilevel algorithms for multi-constraint graph partitioning. In *Proceedings of the 1998 ACM/IEEE Conference on Supercomputing (SC ?98)*, pages 1–13, Washington, DC, USA, 1998. IEEE Computer Society.

[2] G. Karagiorgos and N.M. Missirlis. Accelerated diffusion algorithms for dynamic load balancing. *J. Inform. Proc. Lett.*, 84:61–67, 2002.

[3] E.G. Boman, U.V. Catalyurek, C. Chevalier, and K.D. Devine. The Zoltan and Isorropia Parallel Toolkits for Combinatorial Scientific Computing: Partitioning, Ordering, and Coloring. *Sci. Program.*, 20:129–150, 2012.

[4] F. Schornbaum and U. Rüde. Extreme-Scale Block-Structured Adaptive Mesh Refinement. *SIAM J. Sci. Comput.*, 40:C358–C387, 2018.

[5] H.D. Simon and S.H. Teng. How good is recursive bisection? *SIAM J. Sci. Comp.*, 18:1436–1445, 1997.

[6] J.E. Boillat, F. Brugè, and P.G. Kropf. A dynamic load balancing algorithm for molecular dynamics simulation on multi-processor systems. 96:1–14, 1991.

[7] J.-L. Fattebert, D.F. Richards, and J.N. Glosli. Dynamic load balancing algorithm for molecular dynamics based on Voronoi cells domain decompositions. *Comp. Phys. Comm.*, 183:2608–2615, 2012.

[8] Karen D. Devine, Erik G. Boman, Robert T. Heaphy, Rob H. Bisseling, and Umit V. Catalyurek. Parallel hypergraph partitioning for scientific computing. IEEE, 2006.

[9] A. Nakano and T. Campbell. An adaptive curvilinear-coordinate approach to dynamic load balancing of parallel multiresolution molecular dynamics. *Parallel Comp.*, 23:1461–1478, 1997.

[10] C. Begau and G. Sutmann. Adaptive Dynamic Load-Balancing Method for Particle Simulations. *Comp. Phys. Comm.*, 190:51–61, 2015.

[11] U. Trottenberg, C. W. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, San Diego, 2001.

[12] G. Sutmann. Load Balancing with Generalized Multi-Level Orthogonal Recursive Bisection, 2019. (in preperation).