III International Conference on Particle-based Methods – Fundamentals and Applications
PARTICLES 2013
M. Bischoff, E. Oñate, D.R.J. Owen, E. Ramm & P. Wriggers (Eds)

PARALLEL IMPLEMENTATION OF THE NON-SMOOTH CONTACT DYNAMICS METHOD FOR LARGE PARTICLE SYSTEMS

Matthias Balzer[1], Jan Kleinert[1] and Martin Obermayr[1]

[1] Fraunhofer Institute for Industrial Mathematics (ITWM)
67663 Kaiserslautern, Germany
{balzer,kleinert,obermayr}@itwm.fraunhofer.de, http://www.itwm.fraunhofer.de

Key words: Granular Material, NSCD, HPC, Parallel Implementation

Abstract. In numerous industrial applications there is the need to realistically model granular material. For instance, simulating the interaction of vehicles and tools with soil is of great importance for the design of earth moving machinery. The Discrete Element Method (DEM) has been successfully applied to this task [1, 2]. Large scale problems require a lot of computational resources. Hence, for the application in the industrial engineering process, the computational effort is an issue.

In DEM parallelization is straight forward, since each contact between adjacent particles is resolved locally without regard of the other contacts. However, modelling a contact as a stiff spring imposes strong limitations on the time step size to maintain a stable simulation. The Non–Smooth Contact Dynamics Method (NSCD), on the other hand, models contacts globally as a set of inequality constraints on a system of perfectly rigid bodies [3]. At the end of every time step, all inequality constraints must be satisfied simultaneously, which can be achieved by solving a complementarity problem. This leads to a numerically stable method that is robust with respect to much larger time steps in comparison to DEM. Since a global problem must be solved, parallelization now strongly depends on the numerical solver that is used for the complementarity problem.

We present our first massively parallel implementation of NSCD based on the projected Gauß-Jacobi (PGJ) iterative scheme presented in [4]. Focusing on one-sided asynchronous communication patterns with double buffering for data exchange, global synchronizations can be avoided. Only weak synchronization due to data dependencies of neighboring domains remains. The implementation is based on the Global address space Programming Interface (GPI), supplemented by the Multi Core Threading Package (MCTP) [5] on the processor level. This allows to efficiently overlap calculation and communication between processors.

1 INTRODUCTION

Realistic modelling of granular material is of great interest in numerous industrial applications. Hence, time consuming calculations of large scale systems demand for efficient parallelization of the underlying model. This becomes more and more relevant with an increasing number of cores available on today's compute architectures. Thus, reducing data dependencies in the algorithms is an issue.

In this work we investigate the Non–Smooth Contact Dynamics Method (NSCD). The method constitutes a set of inequality constraints on a system of perfectly rigid bodies, which has to be solved at every time step [3]. Formulating this task as a cone complementarity problem a solution can be found by applying the projected Gauß-Jacobi (PGJ) iterative solver [4]. The application of NSCD on a granular material simulation has shown, that system forces can be predicted in agreement with the Discrete Element Method (DEM). On the other hand, since NSCD is robust with respect to much larger time steps, calculations can be done faster with a reduced number of iterations within the solver, if only the material flow is of concern [6].

We have implemented a parallel version of NSCD with one-sided asynchronous communication patterns. Our first test cases show good scaling behavior on a shared memory system. In addition, the extension to distributed memory architectures allows for large scale simulations. Here, work on the dynamic load balancing is under progress.

The paper is organized as follows: In the next section we introduce NSCD as the underlying model of our implementation. Subsequently, the implementation of the algorithm is described. We discuss details of the domain decomposition and communication concept, starting with the multi-threaded case for the shared memory system. Parallelization for the distributed memory environment is built on top. Finally, we discuss the impact of data dependencies on a simple model calculation.

2 MODEL

The basis of NSCD is a model of perfectly rigid bodies for the particles and unilateral contacts for their interaction. Unilaterality means that only repulsive reaction forces at the contact are valid. The contact force separates particles if they would penetrate otherwise, but it cannot exert an attractive cohesive or adhesive force. This is formulated using a complementarity condition: Either two soil particles are in contact, then a repulsive reaction force is required that keeps them from penetrating, or the particles are separated and no reaction force is needed. If ϕ is the signed distance between two particles, and $\tilde{\lambda}_n$ the reaction force in normal direction of the contact, the complementarity condition amounts to

$$0 \leq \phi \quad \perp \quad \tilde{\lambda}_n \geq 0, \tag{1}$$

i. e. one of the two non–negative values is zero whenever the other is non–zero. So far, only the frictionless case is considered.

The Coulomb Friction model states, that the tangential reaction force $\tilde{\boldsymbol{\lambda}}_t \in \mathbb{R}^2$ at a contact of two particles is restricted by a multitude of the normal reaction force $\tilde{\lambda}_n$ from Eqn. (1). In other words, the total reaction force $\tilde{\boldsymbol{\lambda}} \in \mathbb{R}^3$ is restricted to the Coulomb Friction Cone

$$\mathcal{C} = \left\{ \begin{bmatrix} \lambda_n \\ \boldsymbol{\lambda}_t \end{bmatrix} \in \mathbb{R} \times \mathbb{R}^2 \;\middle|\; \|\boldsymbol{\lambda}_t\| \leq \mu \lambda_n \right\}. \tag{2}$$

The frictional coefficient μ usally takes a value between zero and one. DeSaxcé and Feng derived in [7], that the Coulomb Friction model can be cast into a complementarity condition

$$\mathcal{C}^* \ni \tilde{\mathbf{u}} \quad \perp \quad \tilde{\boldsymbol{\lambda}} \in \mathcal{C} \tag{3}$$

where

$$\tilde{\mathbf{u}} = \begin{bmatrix} \phi + \mu \|\mathbf{v}_{t,\mathrm{rel}}\| \\ \mathbf{v}_{t,\mathrm{rel}} \end{bmatrix} \in \mathbb{R}^3$$

and

$$\mathcal{C}^* = \left\{ \mathbf{u} \in \mathbb{R}^3 \;\middle|\; \mathbf{u}^T \boldsymbol{\lambda} \geq 0 \text{ for all } \boldsymbol{\lambda} \in \mathcal{C} \right\}$$

is the dual cone of $\mathcal{C}$. Here, $\mathbf{v}_{t,\mathrm{rel}}$ denotes the relative tangential velocity at the contact point. Whenever the frictional coefficient and the relative velocity at contact are small, a reasonable approximation is given by using $\mathbf{u} = \begin{bmatrix} \phi & \mathbf{v}_{t,\mathrm{rel}}^T \end{bmatrix}^T \approx \tilde{\mathbf{u}}$, refer to [8] for details regarding this approximation.

Let there be m particles and let $M \in \mathbb{R}^{6m \times 6m}$ be the mass matrix of the system. Let $\mathbf{q} \in \mathbb{R}^{6m}$ denote the generalized coordinates of the particles, $\mathbf{v} \in \mathbb{R}^{6m}$ the translational and angular velocities and let $\mathbf{f}_{\mathrm{ext}} \in \mathbb{R}^{6m}$ be external forces acting on the particles. Due to the perfect rigidity of the particles, the velocity of the particles may be discontinuous, and thus accelerations $\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t}$ do not exist in a classical sense [9]. Nevertheless, the motion of all particles can be described by a measure differential equation

$$M \, \mathrm{d}\mathbf{v} = \mathbf{f}_{\mathrm{ext}} \, \mathrm{d}t. \tag{4}$$

In a numerical simulation, the task is to find the configuration of the system at time t_{j+1}, given its configuration at time t_j. Let $h = t_{j+1} - t_j$ denote the time step size and $\mathbf{v}_{j+1}$ an approximation of $\mathbf{v}(t_{j+1})$. Let there be n potentially active contacts at time t_j, i. e. contacts with distances $\phi_k(\mathbf{q}(t_j))$, $k = 1, ..., n$ smaller than a certain threshold. The equations of motion (4) in discrete time, coupled with the complementarity conditions

$\mathcal{C}_k^* \ni \mathbf{u}_k \perp \tilde{\boldsymbol{\lambda}}_k \in \mathcal{C}_k$ for $k = 1, ..., n$, are

$$
\begin{aligned}
& M\left(\mathbf{v}_{j+1} - \mathbf{v}(t_j)\right) = h\mathbf{f}_{\text{ext}}(t_j) + \sum_{j=1}^{n} \nabla \mathbf{u}_j^T \boldsymbol{\lambda}_j, \\
& \mathcal{C}_k^* \ni \mathbf{u}_k(\mathbf{q}(t_{j+1})) = \begin{bmatrix} \frac{1}{h}\phi_k(\mathbf{q}(t_{j+1})) \\ \mathbf{v}_{k,t,\text{rel}}(t_{j+1}) \end{bmatrix}, \\
& \mathcal{C}_k \ni \boldsymbol{\lambda}_k, \\
& 0 = \mathbf{u}_k^T \boldsymbol{\lambda}_k \quad \text{for all } k = 1, ..., n.
\end{aligned}
$$

The new unknowns $\boldsymbol{\lambda}_k = \int_{t_j}^{t_{j+1}} \mathrm{d}\tilde{\boldsymbol{\lambda}}_k$ are integrals of a force over time with physical unit of momentum.

For the contact with index k, let $D_k = \begin{bmatrix} D_{kn} & D_{kt} \end{bmatrix} \in \mathbb{R}^{6m \times 3}$ be the constraint Jacobian as described in [4, 10]. Here, $D_{kn} = \nabla\phi_k(\mathbf{q}) \in \mathbb{R}^{6m}$ and $D_{kt} \in \mathbb{R}^{6m \times 2}$. We can write

$$
\mathbf{u}_k(\mathbf{q}(t_{j+1})) = D_k^T \mathbf{v}_{j+1} + \begin{bmatrix} \frac{1}{h}\phi_k(\mathbf{q}(t_j)) \\ 0 \\ 0 \end{bmatrix} + \mathcal{O}(h^2)
$$

and thus $\nabla \mathbf{u}_k \approx D_k^T$. The new velocity $\mathbf{v}(t_{j+1})$ can be approximated by solving the non-linear Cone Complementarity Problem (CCP)

$$
\begin{aligned}
\mathbf{v}_{j+1} =& \mathbf{v}(t_j) + hM^{-1}\mathbf{f}_{\text{ext}}(t_j) + \sum_{j=1}^{n} M^{-1} D_j \boldsymbol{\lambda}_j \\
\mathcal{C}_k^* \ni& \mathbf{u}_k = D_k^T \mathbf{v}_{j+1} + \begin{bmatrix} \frac{1}{h}\phi_k(\mathbf{q}(t_j)) \\ 0 \\ 0 \end{bmatrix} \\
\mathcal{C}_k \ni& \boldsymbol{\lambda}_k \\
0 =& \boldsymbol{\lambda}_k^T \mathbf{u}_k \quad \text{for all } k = 1, ..., n.
\end{aligned}
\tag{5}
$$

Finally, implicit Euler integration[1] is used to obtain the new positions of the particles,

$$
\mathbf{q}_{j+1} = \mathbf{q}(t_j) + h\mathbf{v}_{j+1} \approx \mathbf{q}(t_{j+1}).
$$

A popular iterative scheme to solve problems of this kind is the projected Gauß–Jacobi method (PGJ) [3, 4]. Given an estimate $(\mathbf{v}_{j+1}^{(i)}, \boldsymbol{\lambda}^{(i)})$ of a solution $(\mathbf{v}_{j+1}, \boldsymbol{\lambda})$ to Eqn. (5),

[1] In practice, implicit Euler time integration is sufficient for large systems with many non-smooth events. As a consequence, all collisions will be perfectly inelastic.

an improved estimate $(\mathbf{v}_{j+1}^{(i+1)}, \boldsymbol{\lambda}^{(i+1)})$ is constructed via

$$\boldsymbol{\lambda}_k^{(i+1)} = \mathcal{P}_{\mathcal{C}_k}\left[\boldsymbol{\lambda}_k^{(i)} - \omega\eta_k\left(D_k^T\mathbf{v}_{j+1}^{(i)} + \begin{bmatrix} \frac{1}{h}\phi_k(\mathbf{q}(t_j)) \\ 0 \\ 0 \end{bmatrix}\right)\right] \tag{6}$$

$$\mathbf{v}_{j+1}^{(i+1)} = \mathbf{v}(t_j) + hM^{-1}\mathbf{f}_{\text{ext}}(t_j) + \sum_{j=1}^{n} M^{-1}D_j\boldsymbol{\lambda}_j^{(i+1)} \tag{7}$$

where $\mathcal{P}_{\mathcal{C}_k}$ denotes a projection onto the friction cone $\mathcal{C}_k$, η_k is a scalar approximation of $\left(D_k^T M^{-1} D_k\right)^{-1}$ and ω is a relaxation parameter that controls the convergence rate. Eqn. (6) only depends on the velocity $\mathbf{v}_{j+1}^{(i)}$ of the two particles involved in the k–th contact and the momentum $\boldsymbol{\lambda}_k^{(i)}$ from the previous iteration. After Eqn. (6) has been evaluated for all contacts independently, the complementarity conditions

$$\mathcal{C}_k^* \ni \mathbf{u}_k(\mathbf{v}_{j+1}^{(i)}) \quad \perp \quad \boldsymbol{\lambda}_k^{(i)} \in \mathcal{C}_k$$

are satisfied locally for all $k = 1, ..., n$. Then the velocities are updated according to Eqn. (7). This update can cause violations of complementarity conditions, and the entire process has to be repeated until the violations are smaller than a predefined threshold or a maximum number of iterations is exceeded.

3 IMPLEMENTATION

As described in the previous section, in NSCD a non-linear system of equations has to be solved at every time step. Subsequently, with the updated particles' velocities the new particle positions are calculated using an implicit Euler step.

As input data for the PGJ iterative solver all active contacts, i. e. pairs of particles that are likely to collide within the next time step, need to be identified. The distance between particles as well as the contact normal has to be calculated. This is currently done using the collision detection provided by the Bullet Physics Library [11]. To account for the dynamics within the time step, particles are artificially enlarged before collision detection. Thus, all possible collisions of a particle within a fixed range (corresponding to a maximum particle velocity) are guaranteed to be detected.

In the following, we first describe the thread parallel implementation of the PGJ solver on a shared memory system (e. g. compute node). The extension to a distributed memory system (e. g. compute cluster) is built on top.

3.1 Multi-threading

By construction of the PGJ iterative solver, all contacts can be processed in parallel within one iteration step (Eqn. (6)). However, care has to be taken at the velocity update (Eqn. (7)) of the corresponding particles to preclude concurrent read-modify-write access

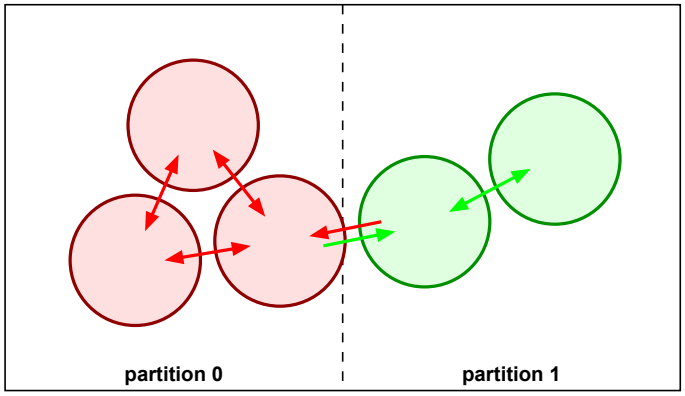

Figure 1: Data access pattern at a partition boundary. Particles are uniquely assigned to a partition (indicated by color). Active contacts are shown as lines connecting the particles, write access is marked by an arrow.

on the particle data. During the iteration, two arrays for the particle velocities are used (double buffering). At each iteration step, the arrays provide the previous state of a particle's velocity (which is read only), and its current value (which is modified by all respective contacts). After the completion of the current iteration step, i. e. all active contacts are processed, the role of previous and current velocities is inverted.

Decomposing the simulation volume into smaller partitions, a particle can be uniquely assigned to a certain partition according to its position. Thus, all contacts including only particles of a given domain can be processed by a single thread without further restrictions. Contacts at the boundaries, connecting neighboring partitions, need to be treated differently. To circumvent explicit synchronization primitives these contacts are processed twice, once for each partition. Here, only the corresponding particle velocities of the processed partition are updated (see Fig. 1).This guaranties exclusive write access to the particle data by each partition.

Though double buffering of the velocities allows for concurrent read and write access on the particle data at the partition boundaries, the iteration scheme of the PGJ solver implies dependencies between neighboring partitions. To procede with the PGJ iteration from a completed step i to the next step $i + 1$, all read velocities need to be given at iteration step i. In other words, two neighboring partitions can not differ by more than one iteration step (weak synchronization). Hence, the current iteration step (or more generally: time stamp) needs to be known for each partition.

For optimal dynamic load balancing the number of partitions needs to exceed the number of simultaneous threads. This reduces idle time due to the aforementioned dependencies at the cost of an increased number of boundary contacts to be processed.

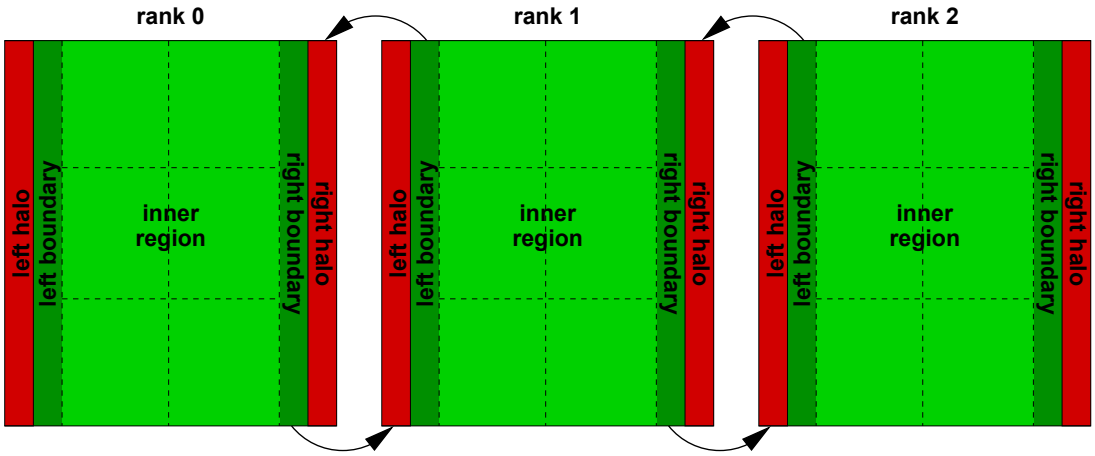

Figure 2: Domain decomposition with boundary exchange. Domains are labeled by the rank of the respective compute resource. Each domain (local volume) is extended by the boundaries of its neighbors (halo). A domain is further divided into partitions (indicated by the dashed lines), including the boundary partition(s).

3.2 Distributed memory architectures

So far, only a shared memory system has been considered in the multi-threaded case. On a compute cluster we additionally have to deal with distributed memory and expensive access of remote data. The parallelization scheme basically remains the same, using domain decomposition and double buffering of particle data. The essential extension is the implementation of a so called halo to speed up access on remote data. This will be described in the following.

Alike the multi-threaded case, a domain decomposition of the simulation volume is implemented, with one domain (local volume) created for each compute resource.[2] Particles are distributed to the compute resources accordingly. We denote the volume assigned to a compute resource *domain* to distiguish from the *partitions* created for the multi-threading environment. Once again, inner contacts and boundary contacts (here connecting neighboring domains) are discriminated. Consistently, the boundary contacts are processed twice, updating local particles only. While in the shared memory environment all particle data is stored only once, limited access times of remote memory on a compute cluster render this storage scheme impracticle for the boundary contacts.

To overcome these limitations each domain is extended by the boundaries of its neighbors (halo), see Fig. 2. The halo provides the necessary data of remote particles to process the domain boundary. Double buffering of the halo data (identical to the double buffering of velocities in the multi-threaded case) allows for one-sided asynchronous communication between compute resources. In other words, the boundary data is sent to neighboring resources without synchronization of the communication process via remote

[2]The term compute resource indicates a single socket or a compute node.

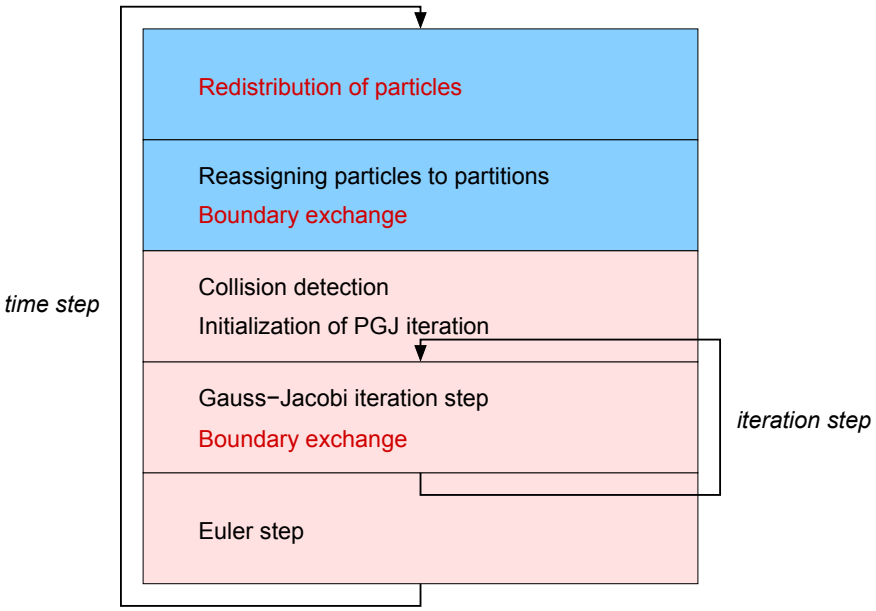

Figure 3: Program flow. Blue boxes indicate pure data management, red boxes contain the actual NSCD algorithm. Tasks involving data exchange with remote compute resources are higlighted in red.

direct memory access (RDMA). The weak synchronization, introduced through the time stamp of each domain, guarantees correct non-conflicting read and write access of the halo data. Furthermore, since the communication is done by the interconnect hardware without involving the CPU, computation and communication can be overlapped. We used the Global Address Space Interface (GPI) in combination with the Multi-Core Threading Package (MCTP) for our purposes [5].

Implementing a two-dimensional decomposition of the simulation volume on the cluster level, the boundary data can be arranged in such a way allowing for zero-copy communication. Calculating boundary contacts first, boundary data can be sent while processing the inner contacts. Thus, in the optimal case the communication is completed before the computation has finished.

3.3 Program flow

The combined domain decomposition for the cluster, including the additional partitions of each domain for the multi-threaded environment is already illustrated in Fig. 2. The boundary of each domain is defined as a separate partition.

The complete program flow on a compute cluster can be summarized as follows (Fig. 3): At the beginning of a time step, particles are reassigned to the domains, i. e. some particles are moved to different compute resources. Second, each domain is divided further into partitions for the multi-threading environment. A boundary partition is introduced for the data exchange between neighboring domains (halo). With the received halo data of all

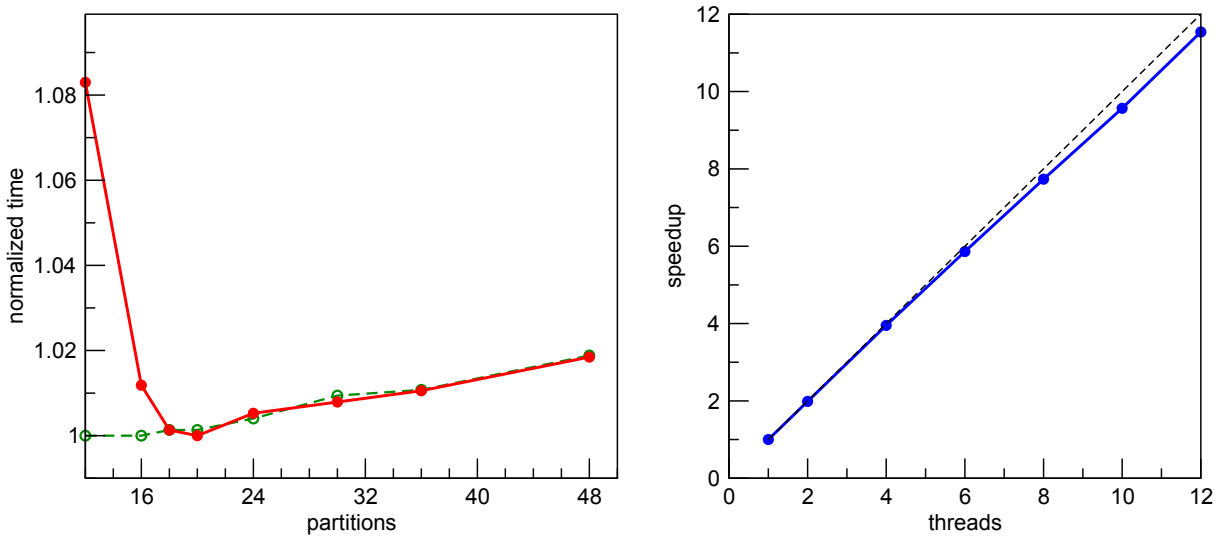

Figure 4: (left) The solid red line shows the normalized runtime for 12 cores depending on the number of partitions, the dashed green line represents the normalized calculation time. (right) Speedup for the shared memory system, see text for details.

neighboring domains, the collision detection is done. Following, the PGJ iterative solver is initialized with the identified active contacts. Then the PGJ iteration is repeated up to a maximum iteration number, involving exchange of the boundary data after every iteration step of the solver. Finally, the particle positions are updated with the Euler step and the next time step starts with redistribution of the particles.

4 RESULTS

As a first test for the multi-threading environment we set up a simulation of $2 \cdot 10^4$ spheres confined in a box (length = 88, width = 22). The radii are randomly distributed between 0.25 and 0.5. Particles are densely packed and uniformly spread over the simulation volume. Runtimes are meassured for 10 time steps with each 2000 iterations. The test is done on a compute node with 2×6 cores (Intel Xeon X5660).

The influence of the volume decomposition is meassured running 12 threads using different numbers of partitions. Results are illustrated in Fig. 4 (left). The solid red curve shows the runtime T_{run} of the job and the dashed green curve the total calculation time T_{calc}, each normalized by their respective minimum values. Data dependencies can be reduced at the cost of increased computational effort, i. e. increased number of partitions causing additional contacts at boundaries. This tradeoff reveals an optimal number of partitons of about twice the number of threads in this example. In the following, we will take this as guideline for the speedup meassurement.

Fig. 4 (right) shows the speedup results for our test case. The ideal (linear) speedup is shown as dashed line. The number of contacts is increased by about 6% at most, the cumulated runtime $T_{\text{run}} \times N_{\text{threads}}$ by about 5%.

Multiple calculation of boundary contacts can be avoided, if neighboring partitions are globally excluded from calculation until the respective partition is processed. While this scheme works well for a large simulation data per thread ratio, data dependencies finally become dominating with an increased number of cores.

5 CONCLUSIONS

We presented our first implementation of NSCD focussing on double buffering and asynchronous communication. For the multi-threaded case we demonstrated with a test model the influence of data dependencies on the speedup with good scaling behavior on a compute node with 2×6 cores. The extension to distributed memory architectures has been implemented and allows for simulations of essentially larger systems. Here, work on the dynamic load balancing is currently under progress.

REFERENCES

[1] M. Obermayr, C. Vrettos, J. Kleinert, and P. Eberhard *A discrete element method for assessing reaction forces in excavation tools.* Proceedings of the Congress on Numerical Methods in Engineering - CNM 2013, Bilbao, Spain, June 2013.

[2] D.A.Horner, J.F. Peters, and A. Carrillo *Large scale discrete element modeling of vehicle-soil interaction.* Journal of engineering mechanics, 127 (10), 1027-1032, (2001).

[3] V. Acary and B. Brogliato *Numerical Methods for Nonsmooth Dynamical Systems.* Applications in Mechanics and Electronics, Springer, 2008

[4] A. Tasora and M. Anitescu *A matrix-free cone complementarity approach for solving large-scale, nonsmooth, rigid body dynamics.* Computer Methods in Applied Mechanics and Engineering, 200(5-8):439 C 453, 2011

[5] R. Machado and C. Lojewski *The Fraunhofer virtual machine: a communication library and runtime system based on the RDMA model.* Computer Science – Research and Development, vol. 23, pp. 125-132, 2009; Fraunhofer ITWM *GPI – Global Address Space Programming Interface.* http://www.gpi-site.com

[6] J. Kleinert , M. Obermayr, and M. Balzer *Modeling of Large Scale Granular Systems using the Discrete Element Method and the Non–Smooth Contact Dynamics Method: A Comparison.* To be published as contribution for the ECCOMAS Multibody Dynamics 2013 Conference Proceedings

[7] G. De Saxcé and Z.Q. Feng *The bipotential method: a constructive approach to design the complete contact law with friction and improved numerical algorithms.* Mathematical and Computer Modelling, 28(4):225–245, 1998.

[8] M. Anitescu *Optimization-based simulation of nonsmooth rigid multibody dynamics.* Mathematical Programming, 105(1):113–143, April 2005.

[9] D. E. Stewart *Rigid-body dynamics with friction and impact.* SIAM Review, 42(1):pp. 3–39, 2000.

[10] M. Anitescu, J. F. Cremer, and F. A. Potra *Formulating 3D Contact Dynamics Problems.* Technical report, The University of Iowa, Iowa, 1995.

[11] *Bullet Physics Library*, www.bulletphysics.org