Powder Technology 278 (2015) 234–247

Contents lists available at ScienceDirect

Powder Technology

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

Hybrid parallelization of the LIGGGHTS open-source DEM code

R. Berger [a,*], C. Kloss [c], A. Kohlmeyer [b], S. Pirker [a]

[a] Johannes Kepler University Linz, Department on Particulate Flow Modelling, Altenbergerstrasse 69, 4040 Linz, Austria
[b] College of Science & Technology, Temple University, Philadelphia, PA, USA
[c] DCS Computing GmbH, Altenbergerstr. 66a - Science Park, 4040 Linz, Austria

ARTICLE INFO

Article history:
Received 7 November 2014
Received in revised form 10 February 2015
Accepted 14 March 2015
Available online 24 March 2015

Keywords:
LIGGGHTS
Discrete Element Method
Hybrid parallelization
MPI
OpenMP

ABSTRACT

This work presents our efforts to implement an MPI/OpenMP hybrid parallelization of the LIGGGHTS open-source software package for Discrete Element Methods (DEM). We outline the problems encountered and the solutions implemented to achieve scalable performance using both parallelization models. Three case studies, including two real-world applications with up to 1.5 million particles, were evaluated and demonstrate the practicality of this approach. In these examples, better load balancing and reduced MPI communication led to speed increases of up to 44% compared to MPI-only simulations.

1. Introduction

LIGGGHTS is an open-source software package used for numerical simulation of granular materials and of heat transfer [1]. It is a DEM [2] code implemented on top of LAMMPS [3], a molecular dynamics code developed by Sandia National Laboratories. Technically, LIGGGHTS is a fork of LAMMPS, the main adaptations being the addition of mesh geometry support, granular models for particle–particle and particle–wall interactions, and particle–particle and particle–wall heat transfer. Numerous other features are added continuously to support special requirements for large industrial cases. Such cases include simulations of bulk solids needed in the chemical industry and steel industry. Examples are the coating process of pills in the pharmaceutical industry and the optimization of dust filters in industrial plants.

LIGGGHTS operates on macroscopic particles and tracks the trajectory of each. It is designed around an integration loop which integrates Newton's second law and resolves particle–particle and particle–wall collisions using a soft-sphere approach. Spring-dashpot models are used to compute forces caused by particle–particle interactions (pair forces) and particle–wall interactions. Additionally, volume forces such as gravity are applied. The total amount of computational work during collisions is cut in half by utilizing Newton's third law, thus avoiding recomputation of the same force with a different orientation.

Fig. 1 illustrates the flow of control for Velocity-Verlet time integration as implemented in LAMMPS and LIGGGHTS. The two time integration steps, which update particle positions and velocities, are bracketed by system modification hooks that allow the simulated system to be manipulated in various ways. Binning of particles and Verlet lists [4] are used to improve the efficiency during collision detection. Periodic spatial sorting of particles ensures that those in close proximity are in nearby memory locations, thereby increasing cache utilization.

Because of the common code base, many positive performance characteristics are inherited from LAMMPS. Both codes can be used in a parallel environment through message passing (MPI) [5]. The original MPI code of LAMMPS used a static domain decomposition [3,6,7] which partitions space such that the area of communication between MPI ranks is minimized. A similar approach was taken by Kacianauskas et al. [6], but they observed an increase in computational effort for simulations of polydisperse material compared to monodisperse material. Gopalakrishnan and Tafti [7] reported an almost ideal speed increase in the DEM portion of a CFD-DEM fluidized bed simulation on up to 64 processors, and a parallel efficiency of 81% on 256 processors.

Static domain decomposition works well for homogeneous condensed matter simulations, but in DEM the typically inhomogeneous and changing distribution of particles across subdomains result in performance-limiting load imbalances. This motivated the development of a load-balancing scheme in LIGGGHTS through which domain boundaries are dynamically adjusted at runtime. This has since been backported into LAMMPS [8] and is described in Section 2.1.

The idea of moving domain boundaries is not new. It has previously been implemented by Srinivasan et al. [9] in the context of molecular dynamics. They generate equally loaded rectangular regions by moving

* Corresponding author.
E-mail addresses: richard.berger@jku.at (R. Berger), christoph.kloss@dcs-computing.com (C. Kloss), akohlmey@gmail.com (A. Kohlmeyer), stefan.pirker@jku.at (S. Pirker).

http://dx.doi.org/10.1016/j.powtec.2015.03.019
0032-5910/

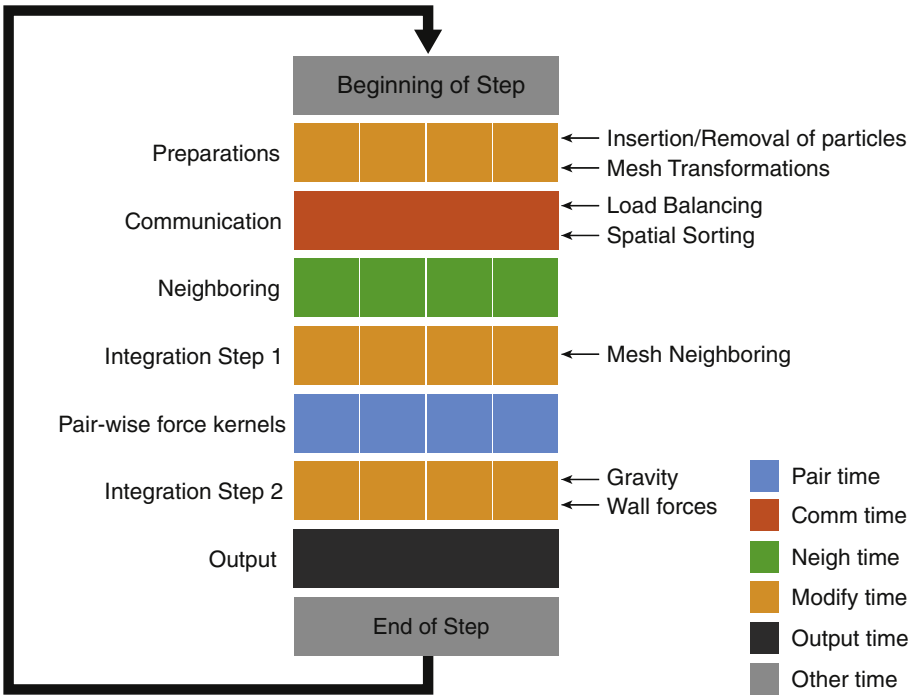

Fig. 1. LIGGGHTS integration loop (simplified). At the beginning of each step, particles may be inserted and meshes transformed. Particles are exchanged between MPI processes, and load balancing may occur. After communication, neighbor lists are built. Forces are computed between integration steps 1 and 2. The parts of the loop that can be parallelized are shown as 4 separate squares. The color codes indicate how these sections contribute to the total timing breakdown.

domain boundaries in increments of a load-discretizing (LD) grid that is coarser than the computation domain (CD) grid. Load balancing occurs on multiple levels by adjusting cuts in the x, y and z directions. The load balancing is based on particle density or number of pairs in each subdomain. In their simulations, a reduction in computation time by as much as 50% was achieved.

More sophisticated load-balancing techniques were employed by Plimpton et al. [10], who applied three different decompositions in a joint finite-element (FE) and smoothed particle hydrodynamics (SPH) code. They employed static FE decomposition of mesh elements, while SPH-decomposition and contact-decomposition of contact-nodes and SPH-particles were performed dynamically using Recursive Coordinate Bisection (RCB) [11]. This geometric algorithm was chosen because it produced well-shaped subdomains and exhibited linear scaling to the problem size.

RCB and many other dynamic load-balancing algorithms were evaluated by Hendrickson and Devine [12]. They listed geometric methods like RCB as being well suited to geometric problems such as particle simulations. RCB in particular is considered to be one of the fastest and easiest to implement in this group of algorithms. It also has the valuable property of being incremental, meaning that small changes in a domain only lead to small changes in the decomposition. This property minimizes expensive communication between processors.

Recently, LAMMPS has also gained the ability to use RCB directly for its MPI decomposition [13]. However, this implementation only balances the number of particles in each subdomain, not the actual workload. It is also incompatible with the current MPI parallelization of meshes in LIGGGHTS.

A different parallelization strategy, known as particle subset method, was employed by Kafui et al. [14] in their CFD-DEM code. They applied a "mincut" graph-partitioning algorithm to a graph of particles and their contacts that generates partitions of particles with a minimal number of contacts with particles in another partition. Based on these partitions, for each MPI process working on a single partition, particles are assigned and halo regions defined. Partitions are recomputed at regular intervals for dynamic balancing.

The desire for better load balancing and better utilization of available compute resources motivated many groups to experiment with the particle subset method using shared-memory parallelizations in their codes. Since fall 2011, LAMMPS has included an add-on package called USER-OMP [15], which provides multi-threaded and thread-safe variants of selected modules and subroutines, in particular force kernels, neighbor-list builds and a few selected other modules. In 2013, our exploration of using these OpenMP modifications for LIGGGHTS started [16].

Concurrently, Amritkar et al. [17] developed an OpenMP parallelization for MFIX DEM code which uses the particle subset method. They argued that for the N-body particulate phase of their CFD-DEM simulation, a parallelization over the number of particles is more suitable. To support their claim, they presented measurements of a fluidized bed simulation (uniform particle distribution) and a rotary kiln heat transfer simulation (non-uniform distribution). Despite higher overheads for fetching non-local data, the better load balancing makes the OpenMP parallelization 50–90% faster than MPI-only. To achieve optimal speed increases, they ensured that data was stored locally by following the first-touch policy, and that thread/process affinity was set using placement tools.

Finally, Liu et al. [18] further expanded on existing MPI and OpenMP work and described a hybrid MPI/OpenMP parallelization of their MFIX-DEM solver. They emphasized that data locality and thread placement policies play a critical role in scaling OpenMP to large core counts. They also mentioned the necessity of distinguishing between private and global data in multi-threaded code, which avoids race conditions by each thread using its own copy of data and reductions. Due to reduced MPI communication, their hybrid becomes faster. Scaling was presented using a coupled CFD-DEM run of a 3D fluidized bed simulation. They reported speed increases of 185× on 256 cores (72% efficiency) of their hybrid parallelization compared to 138× using a standalone MPI computation with 5.12 million particles.