Hybrid OpenMP - MPI Simulation for Large-Scale Granular Dynamics in Chrono

Nicholas Olsen[1], Radu Serban[1], Alessandro Tasora[2] and Dan Negrut[1]

[1]*Department of Mechanical Engineering, University of Wisconsin - Madison, [nicholas.olsen, serban, negrut]@wisc.edu*
[2]*Dipartimento di Ingegneria Industriale, Universitá degli Studi di Parma, alessandro.tasora@unipr.it*

Motivation. Chrono is an open-source, BSD3 available, multi-physics simulation engine developed through a joint university effort [1]. Until recently, the highest level of support for simulating granular dynamics in Chrono has been provided by Chrono::Parallel, a module which uses OpenMP acceleration to expedite computation of large many-body systems. This implementation is limited by (*i*) cache coherency issues that prevent Chrono::Parallel from achieving a more than 15-fold speedup despite using workstations that sport 64 parallel cores; and, (*ii*) constraints on memory size available on a workstation.

Depending on grain size, one cubic meter of sand contains well over a billion elements. This problem size renders futile any Chrono::Parallel attempt for a fully resolved analysis of vehicle mobility on sand, the drifting of a sand dune, etc. As pointed out in [2], more than 50% of the materials processed in industry come in granular form and understanding their dynamics is relevant in a range of practical applications such as additive manufacturing, terramechanics, nanoparticle self-assembly, composite materials, pyroclastic flows, formation of asteroids and planets, meteorite cratering; and also in industries such as pharmaceuticals, chemical and biological engineering, food processing, farming, manufacturing, construction, and mining.

Against this backdrop, we have developed a new Chrono module called Chrono::Distributed, which has been designed to harness the compute power and RAM available on multiple nodes to tackle problem sizes that reach into billions. The new module implements a distributed memory parallel computing framework that uses the Message Passing Interface (MPI) to wrap the existing OpenMP multi-threading implementation in Chrono::Parallel, thus allowing for a modular, stacked software deck: Chrono::Distributed leverages Chrono::Parallel, which relies on the Chrono::Engine.

Overview of Hybrid OpenMP - MPI Solution. The numerical solution methodology adopted formulates the equations governing the time evolution of a mechanical system as a collection of index three differential algebraic equations of motion that account for friction and contact via external forces. The numerical solution of the equations of motion draws on a first order, semi-implicit symplectic Euler method. The current Chrono::Distributed solution models friction and contact using a penalty-based approach. Unlike the differential-variational inequality approach also available in Chrono, the penalty approach leads to a decoupled numerical solution that is more amenable to parallel computing via MPI.

Chrono::Distributed implements a distributed-multicore hybrid solution to leverage parallelism over multiple nodes, each with multiple cores. Multicore parallelism is provided by the Chrono::Parallel module via OpenMP. This is obtained by organizing the underlying data in flat structures of arrays (SoA), which optimizes shared memory access and cache usage, and by using OpenMP parallel for loops and algorithms provided by the Thrust parallel library [3] through its OpenMP back-end. Chrono::Parallel implements a custom parallel binning algorithm for its broad-phase collision detection. Resolution of actual collisions is performed in parallel over all candidate pairs in a narrow-phase based on a hybrid analytical - MPR (Minkovski Portal Refinement) algorithm.

Support for distributed memory simulation in Chrono::Distributed creates multiple processes, each with an instance of a Chrono::Parallel system. At the onset of the simulation, the user-defined spatial domain is statically divided into sub-domains, each of which is assigned to a single MPI process, or rank. For the duration of the simulation, each rank is responsible for computations only in its designated sub-domain and a thin sharing region between it and the neighboring sub-domains. In these sharing regions, each body is owned by one rank and viewed as a proxy body on the other. Each time step, the pair of ranks use a standard Chrono::Parallel time step separately, viewing their own bodies and proxies from their neighbors. After each time step, each rank sends out MPI point-to-point communications to its neighbors to update the proxy bodies that mirror its owned bodies. Pairs of ranks synchronize body data using non-blocking point-to-point communications to minimize latency. This minimizes both the number of messages sent and their sizes in addition to preventing long chains of communication hops within large clusters, largely eliminating the high overhead associated with collective communications over a very large set of ranks. Non-blocking communications allow each rank to overlap its time spent packing data for sends with previous communication times, hiding some of the cost of high-latency MPI communication. In order to maintain a consistent view of the system at all times, each body across the distributed simulation is given a unique global identifier. Each rank is then responsible for mapping this global ID to its own local index for the body. This allows fast lookup of a body across a large distributed system.

Scaling Results for Chrono::Distributed. A preliminary scaling analysis was performed on a settling simulation. Parallel efficiency results for strong and weak scaling runs on a Cray XC30 system are shown in Fig. 1. The compute nodes are equipped with 12-core Intel® Xeon® E5-2697 v2 processors and are interconnected with a dedicated Cray Aries high-speed network. Each Chrono::Parallel instance used 24 OpenMP threads.

Strong scaling yields some efficiencies above 1.0; these are due to the slightly nonlinear scaling of the broad-phase collision detection phase in Chrono::Parallel, which performs better on small problem sizes, and possibly to better cache utilization on smaller local problem sizes. Results from the weak scaling show parallel efficiencies of nearly 1.0, indicating relatively minimal communication overhead for larger communicators.

MPI Ranks	Particles	Problem Size Ratio	Wall Time (s)	Parallel Efficiency
1	1,236,372	1	46,978.9	N/A
2	1,236,372	1	23,653.3	0.993
4	1,236,372	1	11,908.6	0.986
8	1,236,372	1	5,544.9	1.059
16	1,236,372	1	2,879.0	1.020
32	1,236,372	1	1,424.6	1.031

(a) Strong scaling data. Parallel Efficiency: $E(n) = \frac{T(1)}{nT(n)}$

MPI Ranks	Particles	Problem Size Ratio	Wall Time (s)	Parallel Efficiency
1	1,236,372	1	46,978.9	N/A
2	2,472,744	2	47,801.9	0.983
4	4,944,700	4	47,995.1	0.979
8	9,889,400	8	48,198.7	0.975
16	19,778,012	16	48,812.9	0.962
32	39,555,236	32	48,961.5	0.960

(b) Weak scaling data. Parallel Efficiency: $E(n) = \frac{T(1)}{T(n)}$

Fig. 1: Chrono::Distributed scaling results for a granular material settling problem.

Conclusions and Future Work. Initial scaling results for Chrono::Distributed show the potential to allow for granular simulations with numbers of bodies in the billions. With this resolution, we plan to investigate the impact of various element shapes on the behavior of aggregate granular material. We plan to integrate this hybrid implementation into an existing framework such that various other Chrono solvers can interface with the distributed large-scale granular framework via co-simulation [4]. Finally, we plan to upgrade Chrono::Distributed to also support a non-smooth, complementarity-based frictional contact model. This is already supported in Chrono::Parallel, but poses implementation challenges in a distributed fashion since it requires, at each simulation time step, solving a global problem involving all solution variables.

References

[1] A. Tasora, R. Serban, H. Mazhar, A. Pazouki, D. Melanz, J. Fleischmann, M. Taylor, H. Sugiyama, and D. Negrut, "Chrono: An open source multi-physics dynamics engine," in *High Performance Computing in Science and Engineering – Lecture Notes in Computer Science* (T. Kozubek, ed.), pp. 19–49, Springer, 2016.

[2] P. Richard, M. Nicodemi, R. Delannay, P. Ribiere, and D. Bideau, "Slow relaxation and compaction of granular systems," *Nature Materials*, vol. 4, no. 2, pp. 121–128, 2005.

[3] J. Hoberock and N. Bell, "Thrust: A parallel template library," 2010. Version 1.7.0.

[4] R. Serban, D. Negrut, A. M. Recuero, and P. Jayakumar, "A co-simulation framework for high-performance, high-fidelity simulation of ground vehicle-terrain interaction," *Intl. J. Vehicle Performance*, vol. in press, 2017.