Improving Gadget3 Tree Build And Vectorisation

A. Ragagnin

Alma Mater Studiorum Università di Bologna, via Gobetti 93/2, I-40129 Bologna, Italy

Introduction

OpenGadget3 is an OpenMP/MPI parallelized version of the widely used cosmological N-body and adaptive-timestep code P-Gadget3 [1,2]. It follows a large number of physical processes (e.g. gravity, hydrodynamics) and proved to successfully simulate the formation of cosmological structures. Gadget3 particles are organised in an octree and the domain decomposition builds a so-called top-tree, which stores the amount of mass inside a tree-node down to a predefined refinement level; most of its parts are not OpenMP parallelised. Additionally at each time-step the hydrodynamic solver loops over all particles. These two parts are some of the bottlenecks for high-resolution runs.

Methods

To find the bottlenecks in our zoom-in runs (~4e8 particles over 16 nodes, each with 2 MPI tasks, each with 48 OpenMP threads), and found four bottlenecks: (I) the building top-tree that is then exchanged between MPI nodes; (II) the process of merging top-trees exchanged between MPI ranks; (III) the recursive update of the top-tree (masses inside each node) after the domain decomposition; (IV) and the process of adding particles to the final tree.

The tree-build and merge bottlenecks (points I,II, III) are all implemented as a recursive function that contains a for loop iterating over the 8 node-children of the tree-node passed as argument. To parallelise this function we first check if the number of available OpenMP threads is > 8, if so, the above-mentioned loop is executed in parallel. Otherwise it is executed in serial. Figure 1 reports an artistic representation of this refactoring, where the red line on the right-panel shows a final serial part where we stitched results from various OpenMP threads.

For bottleneck IV) we created an array of OpenMP locks, each for each possible tree-node. Then, we parallelised the loop over all particles to be inserted and, for every tree-walk iteration we checked if the OpenMP lock corresponding to the current tree-node is free, and, whenever we found that a thread was going to modify a tree node, we lock/unlock the node.

We tackled the problem of Gadget3 performing a for loop over all particles even on small time bins in order to set a given flag to zero. To this end, we extracted the flag from the AoS layout and put it in a separate array.

Results

We found that the bottleneck (I) went from running in 1.8s to 1.4s; (II) went from 0.9s to 0.6s; (III) went from 2.0s to 1.1s; (IV) went from 3.6s to 1.2s. The total

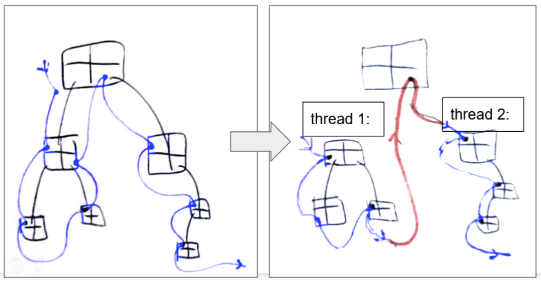

Figure 1 - Visual deception of the octree-build algorithm in serial (left panel) and the OpenMP porting (right panel). Blue line shows the tree-walk of a given thread (main thread for the serial version and an OpenMP thread for the parallel version). Red line in the right panel shows the serial tree-stitching performed after the parallel tree build.

speedup time of domain decomposition and the tree-build over several timesteps went from ~11000s to ~10000s, with a speedup of 1.1. For what concerns the vectorisation of flag-zeroing, for time-bins with less than 5000 particles, we start having a speedup. In particular, these time-bins (which are the ones executed most frequently) we get a speedup from 0.47s to 0.23s for each iteration.

Conclusions

This project focused on improving some bottlenecks for the widely used N-body code for cosmological simulations Gadget3. In particular, previous tests showed that high-resolution simulations suffer from bottlenecks in small adaptive timebins and in some serial parts of the tree-build.

We set up some realistic high-resolution cosmological simulations, profiled its internal parts, vectorised and parallelised them with OpenMP, and obtained a final speedup on tree build and domain decomposition of ~10% and vectorisation made us we save 0.2s for each of the smallest timebins.

References

[1] Hammer N. J. et al., arXiv, 1609.01507
[2] Ragagnin A. et al., Advances in Parallel Computing, Volume 27: Parallel Computing: On the Road to Exascale, OP Ebook, ISBN: 978-1-61499-621-7, pages 411-420

Acknowledgement

The work has been performed under the Project HPC-EUROPA3 (INFRAIA-2016-1-730897), with the support of the EC Research Innovation Action under the H2020 Programme; in particular, the author gratefully acknowledges the support of LMU/USM, PRIN-MIUR 2017 WSCC32 ZOOMING and the computer resources and technical support provided by **LRZ**.