Modernization of the Gasoline code at LRZ

Jonathan Coles [1,2], Salvatore Cielo [1], Aura Obreja [3], Tobias Buck [4]

[1] *Leibniz Supercomputing Centre, Garching by Munich, Germany*
[2] *CSCS - Swiss National Supercomputing Centre, Lugano, Switzerland*
[3] *Universitaets-Sternwarte Muenchen, LMU, Germany*
[4] *Leibniz-Institut fuer Astrophysik, Potsdam, Germany*

Abstract

The simulation code Gasoline2_LAC (Gasoline2 with LPF, AGN and Chemistry updates) aims to push forward the field of galaxy formation and evolution by including novel but computationally demanding physical processes. We, developers of Gasoline2_LAC and the LRZ AstroLab, engaged a joint code modernization project with three initial goals: 1) ensure that the code compiles and runs without errors, 2) lower the memory footprint, and 3) improve the code scaling. Optimizations of particle data structure (for AHF and MPI), Chemistry memory allocation, a more correct usage of the MDL layer and a new and improved domain decomposition now make sure that the code compiles and run correctly.

Mid-project we also improved on the tree and gravity computation, adding also higher-order radiation multipoles in the process, which had larger impact on the performance. We implemented a better integration of tree structure and multipoles, included multipole code from PKDGRAV3, added of r^{-2} multipole moments for orders $p = 3$, 4, and 5, and made the code more SIMD-friendly by using arrays for multipole moments and radiation fields. On small-scale tests, the time-to-solution is reduced by more than 25%, with peaks of $> 80\%$ on specific kernel, which were previously the main bottlenecks when running at large scale. Medium- and large- scale runs are to be executed soon in the framework of the NIHAO2 project, which is partly run on LRZ resources.

1. The Gasoline code

`Gasoline` (Wadsley et al. 18) is a smoothed particle hydrodynamics (SPH) code build on top of the N-body code PKDGRAV (Stadel 14). The SPH part has been updated to deal with the cold blob problem (Agertz et al. 1), and has been publicly released in 2017 as `Gasoline2` (https://github.com/N-BodyShop/gasoline). The implementation of the hydro equations in `Gasoline2` are extensively discussed by [17], while the particular supernova feedback called 'superbubble feedback' which is the default in the public version can be found in [6]. Instead of Keller's model, the base code for the LRZ project is `Gasoline2` with the supernova feedback implementation following the blast-wave model of [15], and pre-heating of the nova environment following the 'early stellar feedback' model of [16]. This base code version, which we call `Gasoline2-NIHAO`, was used to run the Numerical Investigation of Hundred Astrophysical Objects simulation suite (Wang et al. 19) and their higher resolution Milky Way-like and Andromeda-like counterparts (Buck et al. 3).

On the backbone of `Gasoline2-NIHAO`, we have made three independent updates:

- the default cooling module of the code (Shen et al. 13) has been modified to include the effects of photoionization and photoheating from five types of local radiation sources: young stars, old stars, and hot gas emitting Bremsstrahlung in three distinct temperature bins, on top of a redshift dependent but isotropic UltraViolet Background (Obreja et al. 9). We refer to this update as 'Local Photoionization Feedback' `LPF`.

- the independent halo finder C-code `Amiga Halo Finder` (`AHF`, Knollmann and Knebe 7) has been merged with `Gasoline2-NIHAO` to act as halo finder on-the-fly. This functionality was needed to implement black hole seeding, accretion and feedback (Blank et al. 2). We refer to this update as 'Active Galactic Nucleus' `AGN`.

- the chemical enrichment implementation has been updated to run detailed models pre-constructed with `Chempy` (Rybizki et al. 11). This functionality allows to self-consistently trace on-the-fly all elements in the periodic table (Buck et al. 4). We refer to this update as 'Chempy' `Chempy`.

`Gasoline2` has been used in many studies focusing on planet and galaxy formation. The scientific aim of `Gasoline2-NIHAO` and the three updates describe above is to push forward the field of galaxy formation

and evolution by including physical processes thought to be important. E.g. AGN feedback is currently considered the most likely process responsible for quenching star formation in the dark matter halos more massive than $\sim 10^{12} M_{\odot}$. Also, in the vicinity of radiation sources emitting strongly at wavelengths larger than 1 Ryd, the gas cooling time can be significantly increased by the high fluxes of photoionizing photons, compared to the case where only the UVB provides these photons. This latter effect has been shown to decrease the star formation rates, particularly in the central region of galaxies hosted by $\sim 10^{12} M_{\odot}$ dark matter halos, thus producing flatter circular velocity profiles which are in better agreement with observations. Finally, self-consistent chemical enrichment models, which follow the creation of all elements channel by channel – core collapse supernovae (SNe II), SN I, Asymptotic Giant Branch (AGB) stars – provide the most refined tool to investigate the cosmic origin of the chemical elements and study galactic archaeology using simulations.

1.1. Initial code structure

Both the SPH and the N-body parts of the code are written in C, and parallelized using pure MPI. PKDGRAV is a N-body treecode, which uses multipole expansion to fourth (hexadecapole) order for the gravitational force/potential produced by distant masses. PKDGRAV uses a spatially binary tree for both gravity and nearest neighbors finding. The minimum rectangle containing the full simulation volume represents the root-node (root-cell). The tree is constructed in a top down fashion, starting from spatially bisecting the root-node along its longest axis. In practice, this is done by a quick–sort algorithm which partitions the particles about the bisector of the longest axis of the root cell. Next, the code computes the bounding boxes for each of the two children cells, and spatially bisects them to create the children of children cells. The algorithm proceeds in this fashion until the leaf cells (buckets) are reached. The code stops building the tree when all leaf cells have no more than a maximum b number of particles (b=8 in in all cases presented here). Thus, buckets can have between 1 and b particles. The reason for choosing this binary tree structure is that it can be naturally extended to a parallel, distributed tree structure over an arbitrary number of processors. The time cost of this first phase of tree building is dominated by the continuous re-shuffling of pointers to the particle structures. At the end of this phase, the code has constructed the basic tree structure, calculated the bounding boxes of all cells, and ordered the particles in memory. To complete the tree building, the code walks once the tree bottom-up to calculate the center of mass (CM), reduced multipole moments (RMM), and opening radius $r_{\rm open}$ around its CM[1] for all the cells. For the buckets this operation is trivial, as the code uses directly the particle structures associated with each bucket. For all other cells, the code uses the parallel axis theorem to compute the CM and RMM from the CMs and RMMs of the two children. In this second phase of the tree building, the code also calculates the 'luminosities' for the five types of radiation sources of LPF for each cell.

To compute the accelerations (and the five types of LPF radiation fields) at all particle positions, `Gasoline2` walks the tree to create for each particle two interaction lists: a particle–particle one (p–list) and a particle–cell one (c–list). Given that the interaction lists for nearby particles are almost identical, the code optimizes the tree walk by constructing single c- and p-lists for all the particles in a given bucket. To create the interaction lists for a bucket of particles, the code begins from the root–cell and proceeds in a top–down manner, in a recursive procedure. If a cell's opening ball ($r_{\rm open}$, CM) intersects the bucket, this cell is opened and both its children cells are checked against the intersection criteria. If, instead, the opening ball of a cell does not intersect the bucket, its multipole moments are added to the c-list for this bucket. Once this recursive procedure reaches the leaves (buckets), each leaf is checked against the intersection criteria, and if this is met, then this leaf's particles are placed on the p-list of the bucket. Once the c- and p-lists for all the buckets are constructed, the code computes the accelerations and LPF radiation 'fluxes' at all particle positions.

1.2. Time integration

`Gasoline` uses the Kick-Drift-Kick 2^{nd} order leapfrog integration with hierarchical powers of two time stepping [14, 18]. This approach is symplectic only if time steps never change. When particles change time steps (by factors of two each time), secular drift in otherwise conserved quantities can be introduced though this is not systematic and the code conserves energy well overall as shown in the original paper and confirmed in the tests. `Gasoline2` applies time step criteria in a pairwise fashion. For example, the Courant condition is applied as,

$$\Delta t|_{\rm Courant,i} = 0.4 \min_j \frac{\bar{h}_{ij}}{1.25\bar{c}_{ij} + 0.75\left[\alpha_{ij}\bar{c}_{ij} + \beta\mu_{ij}\right]}, \quad (1)$$

[1] $r_{\rm open} = \frac{2}{\sqrt{3}} \frac{r_{\rm max}}{\theta}$, where $r_{\rm max}$ is the distance from the CM to the farthest corner of the cell, and θ has approximately the same meaning as the opening angle of the Barnes-Hut algorithm. θ is a parameter set at run-time.

which is symmetric between the particles. The term in square brackets arises where artificial viscosity is on. This form is standard [e.g. 8]. We also employ the acceleration criterion $\Delta t|_{\rm Acc} = 0.2\sqrt{\frac{h}{a}}$.

[12] showed that large differences in time steps between nearby particle can cause disastrous energy non-conservation. Following their suggestion we never let a particle time step exceed 4 times the current time step estimate of any neighbour. In addition, if high speed events insert new neighbours with significantly shorter time steps, we wake up the long time step neighbours and cut short their current time step to adjust it downwards as much as necessary. This makes the scheme temporarily 1^{st} order but closely limits the overall error. It is critical for strong shocks such as the Sedov-Taylor blast, which has negligible initial temperatures.

We integrate heating, cooling and chemical networks using sub-cycling (arbitrarily many sub-steps, smaller than the hydro time step). These equations are commonly stiff with very stringent stability requirements. We apply a time step limiter to the hydrodynamics based on overall changes to the internal energy, $\Delta t|_{\rm u} = 0.25\ u/(\frac{du}{dt})$. The form of the energy equation ensures that energies can never go negative due to errors in PdV estimates.

The overall code architecture is organized in four layers which we detail below:

- **The Master** layer is essentially a serial code that orchestrates the overall progress of the simulation, handles restarting and checkpointing of the code.
- **The processor set tree (PST)** layer distributes the work and collects the output from parallel operations in an architecture-independent way using the machine-dependent layer (MDL).
- **The machine-dependent layer (MDL)** is a relatively short library of code that implements remote procedure calls, effective memory sharing and parallel diagnostics. MDL has been implemented in using various parallelization protocols: MPI, PVM, pthreads, shmem and CHARM. Localizing the communication in the MDL layer has made it fairly easy to port the code between different architectures.
- **Parallel KD (PKD) layer** manages local trees for gravity and particle data. At this level all the physics is implemented. This modular design enables new physics to be coded at the PKD level without requiring detailed knowledge of the parallel framework.

All processors in the PST level are waiting for directions from the single process executing the Master loop. Work directions are passed down the PST in a tree based $O\log_2(N_P)$ procedure, where N_P is the number of processors, that ends with access to the fundamental bulk particle data on every node at the PKD level. The PKD layer is almost entirely serial acting on local particle data except for a few calls to MDL to access remote data.

1.3. Initial code status and limitations

At the beginning of the LRZ project, `Gasoline2` with the three updates (LPF, AGN and Chempy), which we call `Gasoline2_LAC`, was not running properly, especially on high resolution simulations.

The one big problem of the initial `Gasoline2_LAC` was the large memory footprint which was caused by a combination of two factors. (1) LPF was adding 5 new double-precision variables ("doubles") to the particle structure to hold the radiation field fluxes at each particle position, and 5 new doubles to the cell structure to hold the luminosities of the five types of radiation sources. (2) Chempy was adding at least 30 new doubles to the particle structure to hold the fractions of elements (default 10, but can be up to ~100) produced by three different channels (core-collapse SNe, SN Ia, AGB stars). Additionally, the new cooling module that performed the multi-linear interpolation of the cooling and heating functions in a 7-dimensional parameter space (density, temperature, plus the five LPF radiation fields) was saving the two tables in memory using nested pointers.

Another initial issue lead to random segmentation faults associated with improper usage of the MDL library. In the smooth routine, an `mdlAcquire` retrieves a cache line of particles from a remote rank. This is `MDL_CACHELINE_ELTS=8` particles. It then calls the `init*` function for each of the particles as defined in `smooth.c`. In some rare cases the cache line is not full, but the `init*` function is still called on the full cache line regardless of whether there is valid data there or not. In the case we tracked down here, in `initDistFBEnergy()`, the bounds of an array were read from the structure. These bounds are then incorrect due to a non-full cache and the initialization loop writes to random places in memory. This has been fixed early on in the improvement of the code. Another problem related to the MDL library was found in `smFindRemoteCenter`. The problem was that the cache was running out of free cache lines because particles were acquired but never released. This was fixed by adding an `mdlRelease` in the continue statements.

Another strong limitation of `Gasoline2` is that the code does not scale on more than ~600 CPUs. With the help of LRZ we found one cause of this to be the domain decomposition of the code. The code updates now include a flag for domain decomposition based on an estimate of expected the work (number of active particles) as well as simple particle number count.

2. Goals for the modernization project

The modernization project of `Gasoline2_LAC` had three initial goals: 1) ensure that the code compiles and runs without errors, 2) lower the memory footprint, and 3) improve the code scaling. After starting the project, we also decided to work on improving the accuracy of the radiation fields needed by LPF. As the radiation fluxes were computing on the tree at the same time with the accelerations, the most natural step forward was to use a higher order (the initial code was only zero-order) for the radiation fields' multipoles.

2.1. Test cases

We used two test cases for the code: i) a low resolution, low mass (halo mass of $\sim 2 \times 10^1 1$ $M_\odot$ at redshift 0) galaxy g2.19e11.2x4, and ii) a higher resolution, higher mass ($10^{12.5} M_\odot$ dark matter host halo at redshift 3) galaxy g3.16e12. The problem size of the first galaxy contains $\sim 1.3 \times 10^6$ particles and has a spatial resolution of $\sim$ 800 pc. The second galaxy contains about thirty times as many particles ($\sim 30 \times 10^6$) and has about three times better spatial resolution ($\sim$ 280 pc).

2.2. Changes to the code

Below we first list the most important minor changes which ensured that the updated code can be compiled and run correctly:

- Fixed use of not properly initialized variables.
- Added an alternative AHFGatherParticles in `halofind.c` which creates a new MPI type for particles. This ensures that the item count stays below 2^{31}, regardless of the size of an individual particle. This is similar to the existing code, but without assuming the size of the particle structure is exactly divisible by the size of a long long int.
- Replaced the nested pointers in the LPF cooling module inline indexing functions into flatten arrays. This reduced the memory footprint associated with the cooling and heating tables by $\sim$1/3, and speed up the access to the tables.
- Changed the use of bSplitWork to only control whether domain decomposition is determined by a measure of work or particle number. Added a new parameter bActiveDomainDecomp that controls whether the weight (or number) of active particles is used in domain decomposition instead of from all particles.
- Fixed some bugs in the MDL library to prevent accessing non-initialized memory in the cache and properly release cached particles in the smoothing routine to prevent segmentation faults.

Major changes have been made to the gravity and tree parts of the code in an effort to optimize the most time consuming parts of the code, and to increase the precision of the radiation field calculation. These improvements are partially based on the newer versions of the PKDGRAV [PKDGRAV3, 10], and on the Python code MOSAIC (Coles and Bieri 5), which has been developed to generate optimized fast multipole method operators for arbitrary scalar fields using symbolic algebra. `Gasoline2_LAC` now uses local field tensors from the fast multipole method within the Barnes-Hut gravity solver to avoid multiple loops over the newtonian cell-bucket interaction list. Multipole functions were taken from PKDGRAV3 with slight modifications. Specifically, these changes are:

- The local expansion calculation has been moved into the tree walking routine to avoid building an interaction list. This saves copying overhead which can be substantial.
- The shift-and-add of multipoles has been incorporated into the tree build which significantly reduces the build time. Multipoles are computed for buckets and then combined in pkdCombineMultipoles up the tree.
- Added `moments.c` and `moments.h` from the public release of PKDGRAV3 (`http://www.pkdgrav.org/`).
- Multipole routines have been placed in `moments.h` as static inline routines which allows some compilers to vectorize over calls in loops.
- Newer routines have been moved to the *_lrz files: `cooling_metal_rad_hothalo_lrz.c`, `cooling_metal_rad_hothalo_lrz.h`, `grav_lrz.c`, and `walk_lrz.c`. Older routines have been commented out in the original `grav.c` and `walk.c`.

- Added a tiling structure for computing moments in `walk_lrz.c`. Moments are added to a small array which can fit in larger cache. Once the cache is full the moments are evaluated in a vectorizable loop.
- `radfield.c` and `radfield.h` have been created to separate the new radiation field code. Signatures have been modified to take an array of radiation fields instead of explicitly named ones. RDFActive-Class can now be used to ignore disabled radiation classes in the cooling routines by setting values to zero.
- Added $1/r^2$ multipole moment functions for expansion orders p = 3, 4, and 5 (`momr2p3.h`, `momr2p4.h`, `momr2p5.h`). The code can switch among the three options by including the corresponding header file in `moments.h`.

2.3. *Final performance*

Speedups for all the kernels interested in the optimization are graphed in Figure 1: the *Gravity* and *GravTree* components present the most significant speedups (~ 20% and > 80%, respectively).

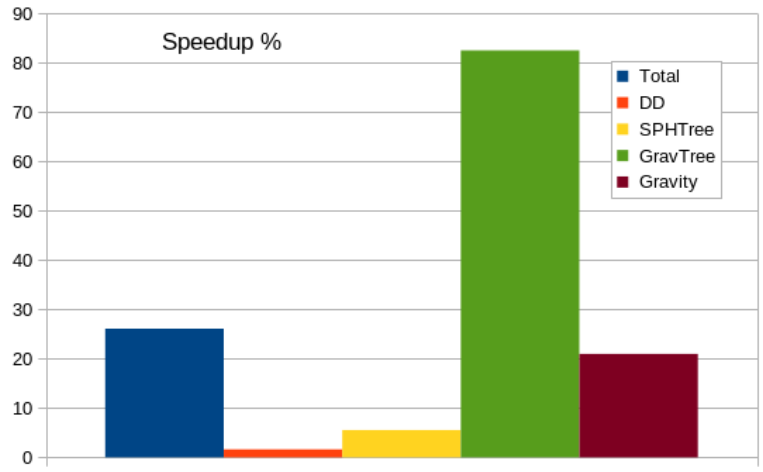

Figure 1: Percent speedup for all the optimized kernels, and for the total execution time. This is measured on the g2.19e11 test, for a single node using the full 48 MPI tasks.

In Figure 2 we show instead a complete single-node scaling test of g2.19e11 case, featuring the time to solution of individual kernels. While the scaling, on this test case, deviates still early from the ideal case

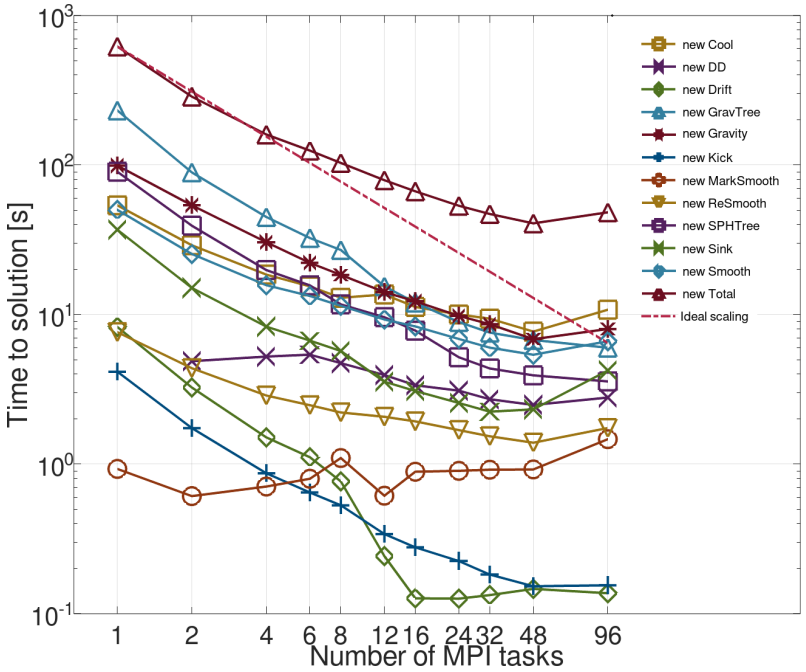

Figure 2: Full strong scaling (time to solution vs. number of tasks) for the g2.19e11 test case, in the optimized (*new*) case, for all the different code kernels.

(likely due to load imbalance), the optimized performance is still > 25% better than the initial. The figure is also a guide for finding bottlenecks in view of future optimization projects: due to its improved scaling, GravTree is no more the top-pressure bottleneck. Work could be dedicated to improve the performance of the Cooling unit. Another consideration to draw is that running in the Hyper-threading (in this case, with 2 MPI tasks per physical core) is never beneficial.

Finally, in Figure 3 we show a detailed communication pattern produced by Intel Application Performance Snapshot (APS) which, besides giving a general, lightweight but indicative profiling, is also able to analyze local communication.

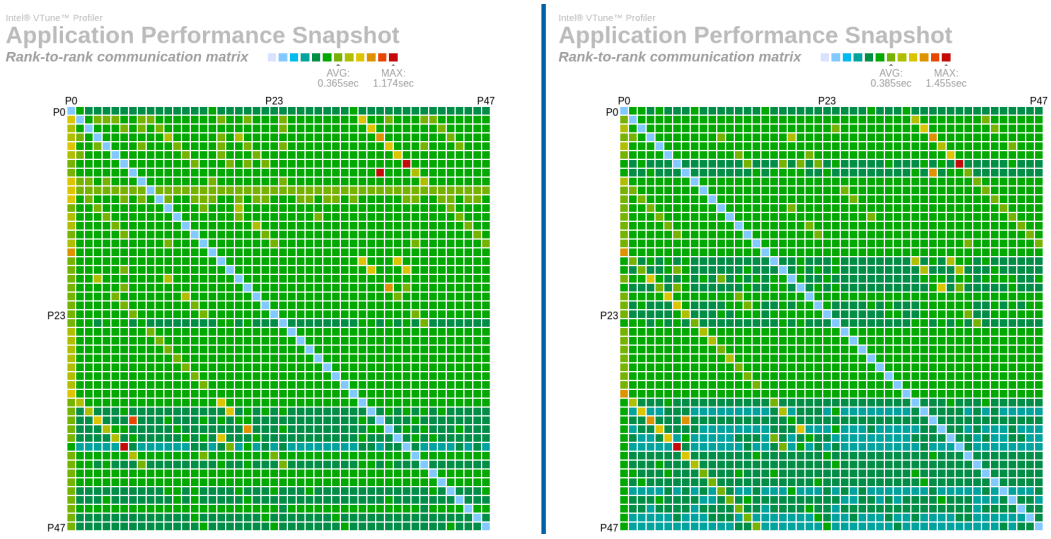

Figure 3: Full rank-to-rank communication matrix for the g2.19e11 test case, as produced by APS. Although overall the pattern is very similar, as a result of the optimization, the average communication time is reduced (see legend) in the optimized version (left) with respect to the baseline (right).

The pattern of communication, with many features parallel to the first diagonal, is typical of neighbor-to-neighbor communication pattern on a Cartesian (logical) topology. The hotspots of higher communication time (once again, indicative of load imbalance) stand clearly out, in both the optimized (left panel) and initial (right panel) case. A low-communication background (in green) is also observed in both cases, as APS includes both point-to-point and global communication in the computed value. This is however not a concern, as it is not the limiting factor in the communication. We also gladly observe that while the hotspots have remained tied to the same tasks, the overall communication times have decreased (see scale in legend).

As a conclusion, after the optimization we are now able to run simulations more reliably, use more MPI ranks per node with less performance loss, and reach increased accuracy in the radiation fields thanks to the new physics modules. This opens to the possibility of running higher-resolution galaxies with the same resources, or to increase the number of galaxies we can simulate with a given computational budget, for both our LRZ allocation, and in the framework of NIHAO2.

3. Outlook

`Gasoline2_LAC` will be used to complete the LRZ project under project number pn29mo with a total of 20 Mio-CPUh allocated, and run the new generation of NIHAO galaxies. NIHAO2 aims to construct a large sample of high resolution galaxies for $z > 3$, covering from massive dwarfs at that redshift to hosts of high-z quasars. These simulations will help answering some of the open questions in galaxy formation, in the light of the new observational facilities like the James Webb Space Telescope, or the Athena X-ray observatory.

The changes made to the LPF part of the code (high precision of radiation fields, generic functions to compute the radiation fields' multipoles, possibility to easily enable/disable various components) are extremely useful for the on-going DFG project of extending the LPF module with the photoionization and photoheating effects from accreting super massive black holes ("The impact of AGN-radiation on the evolution of galaxies and their circumgalactic medium", https://gepris.dfg.de/gepris/projekt/443044596).

From a purely computational point of view, there is definitely still space for improvement on the scalability of `Gasoline2_LAC`. Given the structure of the code, further gains in scalability can be made by switching from pure MPI to hybrid (e.g. OpenMP+MPI).

References

[1] Agertz, O., Moore, B., Stadel, J., Potter, D., Miniati, F., Read, J., Mayer, L., Gawryszczak, A., Kravtsov, A., Nordlund, Å., Pearce, F., Quilis, V., Rudd, D., Springel, V., Stone, J., Tasker, E., Teyssier, R., Wadsley, J., and Walder, R. (2007). Fundamental differences between SPH and grid methods. *MNRAS*, 380(3):963–978.

[2] Blank, M., Macciò, A. V., Dutton, A. A., and Obreja, A. (2019). NIHAO - XXII. Introducing black hole formation, accretion, and feedback into the NIHAO simulation suite. *MNRAS*, 487(4):5476–5489.

[3] Buck, T., Obreja, A., Macciò, A. V., Minchev, I., Dutton, A. A., and Ostriker, J. P. (2020). NIHAO-UHD: the properties of MW-like stellar discs in high-resolution cosmological simulations. *MNRAS*, 491(3):3461–3478.

[4] Buck, T., Rybizki, J., Buder, S., Obreja, A., Macciò, A. V., Pfrommer, C., Steinmetz, M., and Ness, M. (2021). The challenge of simultaneously matching the observed diversity of chemical abundance patterns in cosmological hydrodynamical simulations. *MNRAS*, 508(3):3365–3387.

[5] Coles, J. P. and Bieri, R. (2020). An optimizing symbolic algebra approach for generating fast multipole method operators. *Computer Physics Communications*, 251:107081.

[6] Keller, B. W., Wadsley, J., Benincasa, S. M., and Couchman, H. M. P. (2014). A superbubble feedback model for galaxy simulations. *MNRAS*, 442(4):3013–3025.

[7] Knollmann, S. R. and Knebe, A. (2009). AHF: Amiga's Halo Finder. *ApJS*, 182(2):608–624.

[8] Monaghan, J. J. (1992). Smoothed particle hydrodynamics. *ARA&A*, 30:543–574.

[9] Obreja, A., Macciò, A. V., Moster, B., Udrescu, S. M., Buck, T., Kannan, R., Dutton, A. A., and Blank, M. (2019). Local photoionization feedback effects on galaxies. *MNRAS*, 490(2):1518–1538.

[10] Potter, D., Stadel, J., and Teyssier, R. (2017). PKDGRAV3: beyond trillion particle cosmological simulations for the next era of galaxy surveys. *Computational Astrophysics and Cosmology*, 4(1):2.

[11] Rybizki, J., Just, A., and Rix, H.-W. (2017). Chempy: A flexible chemical evolution model for abundance fitting. Do the Sun's abundances alone constrain chemical evolution models? *A&A*, 605:A59.

[12] Saitoh, T. R. and Makino, J. (2009). A Necessary Condition for Individual Time Steps in SPH Simulations. *ApJL*, 697:L99–L102.

[13] Shen, S., Wadsley, J., and Stinson, G. (2010). The enrichment of the intergalactic medium with adiabatic feedback - I. Metal cooling and metal diffusion. *MNRAS*, 407(3):1581–1596.

[14] Stadel, J. G. (2001). *Cosmological N-body simulations and their analysis*. PhD thesis, University of Washington, Seattle.

[15] Stinson, G., Seth, A., Katz, N., Wadsley, J., Governato, F., and Quinn, T. (2006). Star formation and feedback in smoothed particle hydrodynamic simulations - I. Isolated galaxies. *MNRAS*, 373(3):1074–1090.

[16] Stinson, G. S., Brook, C., Macciò, A. V., Wadsley, J., Quinn, T. R., and Couchman, H. M. P. (2013). Making Galaxies In a Cosmological Context: the need for early stellar feedback. *MNRAS*, 428(1):129–140.

[17] Wadsley, J. W., Keller, B. W., and Quinn, T. R. (2017). Gasoline2: a modern smoothed particle hydrodynamics code. *MNRAS*, 471(2):2357–2369.

[18] Wadsley, J. W., Stadel, J., and Quinn, T. (2004). Gasoline: a flexible, parallel implementation of TreeSPH. , 9(2):137–158.

[19] Wang, L., Dutton, A. A., Stinson, G. S., Macciò, A. V., Penzo, C., Kang, X., Keller, B. W., and Wadsley, J. (2015). NIHAO project - I. Reproducing the inefficiency of galaxy formation across cosmic time with a large sample of cosmological hydrodynamical simulations. *MNRAS*, 454(1):83–94.