Available on-line at www.prace-ri.eu

Partnership for Advanced Computing in Europe

Large Scale Simulations of the Non-Thermal Universe

Claudio Gheller[a,*], Graziella Ferini[a], Maciej Cytowski[b], Franco Vazza[c]

[a]CINECA, Via Magnanelli 6/3, Casalecchio di Reno, 40033, Italy
[b]ICM, University of Warsaw, ul.Pawinskiego 5a, 02-106 Warsaw, Poland
[c]Jacobs University,Campus Ring 1, 28759 Bremen, Germany

Abstract

In this paper we present the work performed in order to build and optimize the cosmological simulation code ENZO on the Jugene, Blue Gene/P system available at the Forschungszentrum Juelich in Germany. The work allowed us to define the optimal setup to perform high resolution simulations finalized to the description of non thermal phenomena (e.g. the acceleration of relativistic particles at shock waves) active in massive galaxy clusters during their cosmological evolution. These simulations will be the subject of a proposal in a future call for projects of the PRACE EU funded project (http://www.prace-ri.eu/).

Project ID: PRA2IC

1. Overview and project goals

There is nowadays evidence that the phenomenology of large scale diffuse radio emissions from galaxy clusters (hereafter GC, e.g. Ferrari et al.2008 for a review) is connected to a perturbed dynamical state of host systems and to processes of acceleration and re-acceleration of relativistic particles in a weakly magnetized intra-cluster plasma (e.g. Brunetti & Lazarian 2010 and references therein). Even if the basic processes leading to the development of a sizable pool of relativistic particles are rather well understood (Fermi I acceleration at shocks, Fermi II stochastic acceleration in turbulent magnetized flow, high energy collisions between relativistic particles and thermal particles), the relative efficiency of such processes in the various phases of astrophysical plasmas found in GC environments are still very uncertain, due to the large technical difficulties in obtaining good spatially and energy resolved data at the wavelength related to such emissions (i.e., radio, gamma, hard-X). Cosmological numerical simulations are a powerful tool to study the time evolution of GC and matter accretions driving shocks and turbulent motions in the intra cluster medium (e.g. Borgani & Kravtsov 2009 for a review). However the realistic treatment of non-thermal processes achieved in present cosmological simulations is not yet satisfactory (e.g. Dolag et al.2008 and references therein). Currently, numerical simulations have to face a number of major issues, related both to the maximum achievable (mass and/or spatial) resolution (at the same time, we need to model with high enough accuracy the cosmological large scale associated to the growth of cosmic structures, and the "small" scales associated to plasma processes and CR processes) and to the physical problems (i.e. the run-time implementation of some physical processes can be very expensive or numerical challenging: for instance, the spectral evolution of accelerated CR require a run-time treatment of Fokker-Planck equations, an extremely demanding operation). According to preliminary studies performed by our group, and focusing on turbulence and shocks in Large Scale Structures (Vazza, Brunetti & Gheller 2009; Vazza et al.2009,10, a,b), our goal is to apply several updated techniques to model the most likely physical scenario(s) leading the production and evolution of CR particles in the Universe. The ensemble of simulated data will provide an unprecedented tool to guide the theoretical interpretation of existing and short-coming radio (LOFAR, LWA, SKA) and gamma (FERMI) and X-ray (ASTRO-H) observations, and will help in discriminating among different theoretical models of nonthernal mechanisms in the GC plasma.

The overall objective of the project is to employ the cosmological Adaptive Mesh Refinement code ENZO (Bryan et al. 1995; O'Shea et al. 2004; Norman et al. 2007), extended by our group with some "custom" functions (see section 2.2.), to study in details the non thermal phenomena (shocks, turbulence, Cosmic Rays acceleration, magnetic fields and Active Galactic Nuclei feedback) active in massive galaxy clusters during their cosmological evolution. This will represent a big step forward for the achievement of a holistic view of non-thermal phenomena related to galaxy clusters in the evolving Universe. For this purpose we need to perform

*Corresponding author.
tel. +39 051 6171560 fax. +39 051 6137273 e-mail. c.gheller@cineca.it

simulations of a cosmological box of $\approx$ 100 Megaparsec (Mpc) with a resolution of the order of a few tens of kpc. This can be reached by:

- using a uniform computational mesh (UNIGRID approach) of linear size of about 2000-4000 cells;
- using an AMR grid, with base grid of about 500 cells and 2 to 4 refinement levels.

The total computing time should not exceed a few millions of CPU hours and the number of computational nodes should be such that: a) enough memory is available, b) we get an acceptable efficiency ($\geq$0.5).

The corresponding computational resources are available only on large HPC systems, such as the Tier-0 systems available in the framework of the PRACE project. This work is a preparatory phase aiming at enabling the ENZO code to run efficiently and effectively on the Jugene, Blue Gene/P system available at the Forschungszentrum Juelich in Germany. Such preparatory work will enable our group to submit a scientific proposal in one of the next calls for large scale projects in the PRACE framework.

2. Code setup

The first part of the work was dedicated to the porting and optimization of ENZO and the associated initial conditions generator INITS (see section 2.4.) on the Jugene platform. The work on INITS proved to be particularly challenging, since only a serial version of the code was available. The full parallelization of the initial conditions generator was accomplished.

2.1. The HPC platform: Jugene

Jugene (Juelich Blue Gene) is a supercomputer built by IBM for Forschungszentrum Juelich in Germany and based on Blue Gene P architecture. In its configuration, Jugene consists of 72 racks, each containing 1024 compute nodes. A single compute node has a quadcore PowerPC450, with a clock frequency of 850 MHz and a total memory of 2 GB. Blue Gene P architecture allows to choose among three different execution modes:

- SMP or Shared Memory mode, in which one core runs one MPI process (and up to 3 threads on the other cores in the node) and may access the full node memory;
- DUAL or Coprocessor mode, in which two cores run one MPI process each (and up to 1 thread each on core not used by the other process) and may access only half of the full node memory;
- VN or Virtual Node mode, in which each core runs one MPI process (and no threads can be spawn) and may access only 1/4 of the full node memory.

For a pure MPI parallel application, such as the one used in the present project (ENZO-2.0), this means that it is possible to choose between single-MPI-process or multiple-MPI-processes per core. Improved performance can of course be gained in VN mode, but this choice imposes a quite strict limitation on the amount of available memory, since, in such a case, each core may access only 512 MB of memory. As we will deeply discuss in the following, this represents the main hurdle in running ENZO-2.0 on Jugene, due to the great number of physical variables demanding a large amount of memory.

2.2. The ENZO code

ENZO is an adaptive mesh refinement (AMR) cosmological hybrid code highly optimized for supercomputing (Bryan et al. 1995; O'Shea et al.2004; Norman et al. 2007). ENZO couples an N-body particle-mesh solver describing the dark matter, with an adaptive mesh method for ideal fluid-dynamics (Berger & Colella, 1989), that follows the evolution of the baryonic component. The two components are coupled via gravitational field, calculated by means of a FFT-multigrid based approach. The fluid dynamics is solved adopting an Eulerian hydrodynamical solver based on the the Piecewise Parabolic Method (PPM, Woodward & Colella, 1984), that is a higher order extension of Godunov's shock capturing method (Godunov 1959). The PPM algorithm belongs to a class of schemes in which an accurate representation of flow discontinuities is made possible by building into the numerical method the calculation of the propagation and interaction of non-linear waves. It is at least second-order accurate in space (up to the fourth-order, in the case of smooth flows and small time-steps) and second-order accurate in time. The PPM method describes shocks with high accuracy and has no need of artificial viscosity, leading to an optimal treatment of energy conversion processes, to the minimization of errors due to the finite size of the cells of the grid and to a spatial resolution close to the nominal one. In the cosmological framework, the basic PPM technique has been modified to include the gravitational interaction and the expansion of the Universe.

The AMR approach adopts an adaptive hierarchy of grid patches at varying levels of resolution. Each rectangular grid patch (referred to as a "sub-grid" or "sub-box") covers some region of space in its parent grid which requires higher resolution, and can itself become the parent grid to an even more highly resolved child grid. ENZO's implementation of structured AMR poses no fundamental restrictions on the number of grids at a given level of refinement or on the number of levels of refinement. However, owing to limited computational resources it is practical to institute a maximum level of refinement. Additionally, the ENZO AMR implementation allows arbitrary integer ratios of parent and child grid resolution, though in general for cosmological simulations (including the work described in this paper) a refinement ratio of 2 is used.

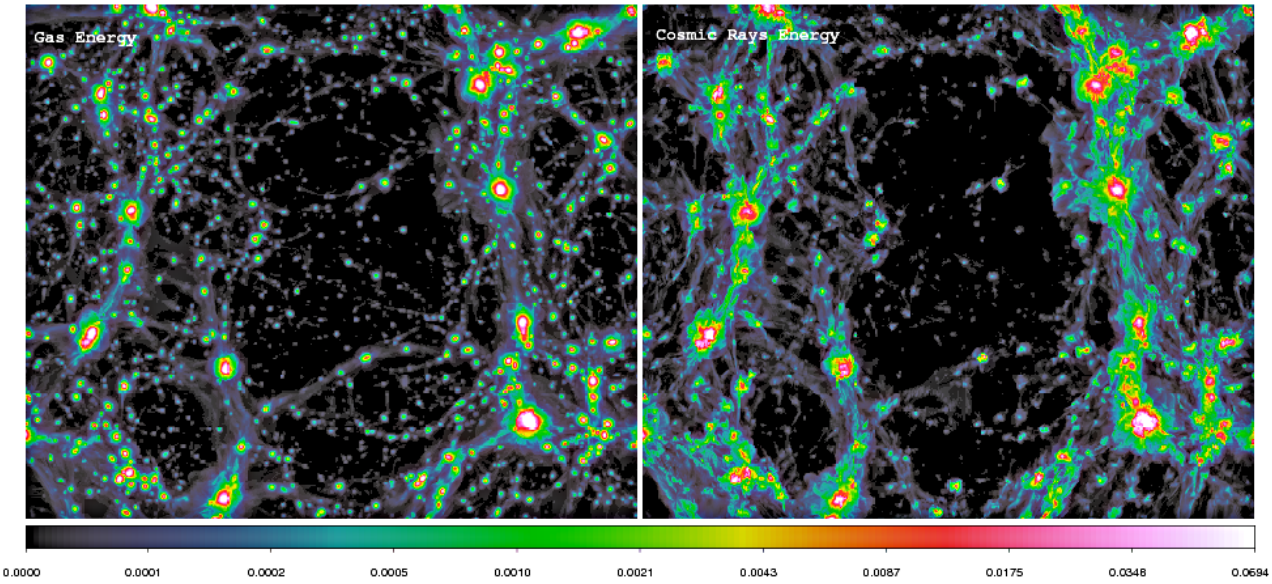

Fig. 1. Projected gas energy (left) and Cosmic Rays energy (right) across a simulated volume of cubic 100 Mpc/h, and a uniform resolution 512^3, the energies are in arbitrary units.

The code is parallelized by domain decomposition into rectangular sub-grids, including the top/root grid (which is the only level in a non-AMR run). Message passing paradigm is adopted and implemented by means of the MPI library (see http://www.mpi-forum.org/), while I/O makes use of the HDF5 data format (see http://www.hdfgroup.org/HDF5/).

The public ENZO code has been extended by our group with a number of ad hoc techniques and algorithms, which were successfully applied to various ENZO runs. For instance we developed:

- a mesh refinement criterion tailored to track shock waves at run time (Vazza et al.2009);
- an algorithm to inject and follow the propagation of tracer (massless) particles in simulations (Vazza, Gheller & Brunetti 2010);
- simplified algorithms to model at run-time a) re-ionization in the early Universe; b) pre-heating of the baryon gas due to the effect of galactic activity(Vazza 2010, Vazza et al.2010a,b);
- a model of thermal energy feedback ("quasar mode") from AGN (Vazza 2010).

Finally, a new module is being tested, which allows to follow at run-time the injection of CR particles at shocks, their spatial advection and dynamical feedback on the thermal gas (following a two-fluid approximation).

2.3. Building and optimizing ENZO on Jugene

Due to the particular features of the Blue Gene/P architecture, a few issues needed to be fixed to successfully port ENZO on Jugene. This goal has been mainly achieved through the creation of a proper make.Mach* file, which usually proceeds through the following basic steps: i) the choice of the suitable cross-compilers and libraries; ii) the specification of the proper compiling and linking flags; iii) the definition of some preprocessor variables to execute pieces of code specific for the architecture. Concerning the first point, the thread safe versions for compilers, both for Fortran and C++ languages, have to be used (namely, the mpixlcxx_r and mpixlf90_r wrappers), while particular attention has to be paid for linking. Indeed, in addition to essl and xlf90_r libraries (and, of course, HDF5 and zlib), which are required also for enabling ENZO on other platforms (e.g. IBM SP Power6 or Linux clusters), further libraries such as xlfmath and pthread are needed for the Blue Gene P architecture. Just for sake of completeness, we finally mention some other minor tricks: i) optimized code for the Blue Gene/P architecture can be built by specifing the -qtune=450, -qarch=450 options; ii) as usually necessary with IBM xl-compilers, the flags for fixed form (-qfixed) and f90 suffix (-qsuffix=f=f90) fortran source files, together with the -qlanglvl=oldmath for C source files, have to be included among compiler settings.

Further effort has been necessary to improve the application performance, which in its default configuration exhibits only a limted scalability. A detailed profiling of the application has been obtained using both gprof and SCALASCA profilers. Such an analysis allowed to detect the main sources of performance loss, which are:

- the default transposing of the grid when computing the FFT for the gravitational potential;
- the default mode for gravity calls.

The improvement of the performances was possible by properly configuring ENZO, by means of suitable setup options. This, however, resulted to be a challenging task, since such options are not documented and the optimal setting was found only analyzing the details of part of the source code.

The first issue could be fixed setting the parameter UnigridTranspose to the value of 2. Such a parameter actually determines the choice among 3 different implementations of the transposing method. The first one, corresponding to the default value 0, does not scale with the processor number. An improved version, scaling better because it decreases the number of MPI messages, can be used by setting UnigridTranspose=1 . However it uses a lot of memory when a number of processors ≥ 1024 is used. A more memory conservative method can be chosen if UnigridTranspose = 2. Such a parameter has to be set in the parameter file containing all the necessary input variables for the cosmological simulation.

Equally crucial for performances is the choice of the mode for gravity calls. ENZO-2.0 source code has two modes: i) bitwise-yes, which relies on fully blocking calls in the gravity and ensures bitwise identicality for different runs; ii) bitwise-no, which allows the non-blocking calls in gravity, so that runs are not guaranteed to be bitwise identical, as it can depend on the order of arrival for the gravity solver. We got a substantially worse performance in the blocking versus the non-blocking runs, while no appreciable difference on the results was found. To choose the non-blocking gravity calls it is necessary to build ENZO with the "make bitwise-no" option. A further limit on ENZO's scalability is intrinsic to the FFT solver. This is discussed in more details in section 3.

Further attempts to gain better performance have been made by checking the application behavior under different settings of many Blue Gene P environment variables (such as DCMF_EAGER, DCMF_RECFIFO, DCMF_OPTRVZ, DCMF_SSM, DCMF_INTERRUPT) which control buffer sizes, protocols and specific details of message passing implementation. However, no sizeable improvement has been noted with respect to the default settings.

A final remark concerns AMR simulations. Indeed the code is built by default with a maximum number of subgrids set to a quite small value (10^5), which for realistic cases needs to be increased. Also this parameter has to be modified at building-time, specifing "make max-subgrids-N", where N stands for the proper value.

2.4. Initial conditions

ENZO code generates initial conditions with through the INITS code, which is part of the distribution and is normally built together with ENZO. INITS is a sequential code, that produces 4 files, storing the initial density and velocity fields of the fluid and the initial positions and velocities of the particles. Using the sequential code, grids larger than 1024^3 cannot be generated on the currantly available computing platforms, since too much memory would be needed ($>$ 100 GB). Initial conditions for larger configurations can be obtained only with a parallel version of the INITS code, where data is properly distributed across memory of computational nodes available on the Blue Gene/P architecture. The work accomplished to parallelize INITS can be summarized as follows:

- Compilation, testing and performance analysis on a x86_64 node.
- Parallelization (described in details below).
- Analysis of the correctness of the results.
- Compilation and test run of the parallel code on x86_64 cluster.
- Compilation and test run on IBM Blue Gene/P system.
- Performance analysis on IBM Blue Gene/P system.

The first step represents just a set-up phase to get used to the code and its basic features on a standard computing environment. For the parallelization, we have adopted a domain decomposition approach. Furthermore, since according to the results of the set-up phase, most of the computational effort of the INITS algorithm is due to Fourier Transforms, we have selected the parallel FFT library that ensures good scalability on a large number of processors. The P3DFFT library (see http://www.sdsc.edu/us/resources/p3dfft/) proved to be the most suitable. In fact, apart from good scalability benchmarks, it supports a rectangular, rather than plane parallel, domain decomposition. This allows to overcome the typical limitation imposed by plane parallel FFTs, that scale with the linear size of the computational mesh. The P3DFFT library allows to scale with the square of the linear size, permitting to exploit a much larger number of processors (hence, much larger computational volumes). The parallelization encompassed the following steps:

- Basic MPI parallelization and P3DFFT initialization: data is divided between processors according to the basic P3DFFT domain decomposition.
- Parallel data representation for the preparation of the random field: One of the hardest parts of the project. Fortran 90 functions included in Rmake_field_kpreserving.src were almost totally rewritten.
- FFT computations with P3DFFT The P3DFFT was adopted dsc and proved to work properly and efficiently for parallel executions. It resulted to be faster than the original INITS FFT even for 1 MPI process run. The results are numerically very similar.
- Parallel I/O INITS data files are written using the HDF5 standard. We have exploited the HDF5 parallel interface, in order to support efficient parallel data writing.

For the generation of the initial conditions, the performance is not the main issue. Memory represents the main constraint, that has been solved as described above. Another critical point is represented by the quality of the results. The generated initial conditions must be close, if not identical, to those generated by the sequential

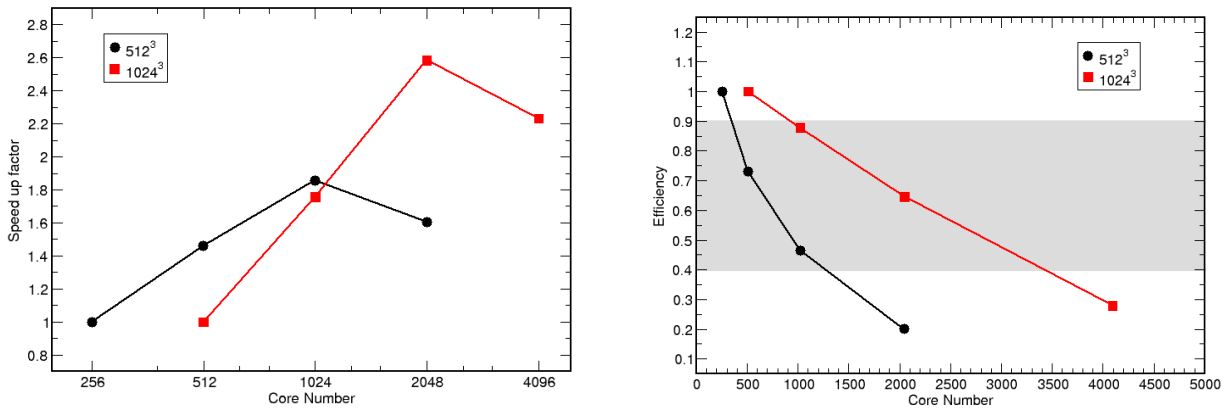

Fig. 2. Speed-up (left) and efficiency (right) of the 512^3 and the 1024^3 tests.

INITS code. This has been extensively tested running the parallel version of the generator starting from different configurations and running on various numbers of cores on Blue Gene/P. The results have been compared to those generated by the original INITS code, using the h5diff utility, that performs a bitwise comparison between the datasets in the HDF5 files. Only about 10% of the numbers differ, but the difference is always at the precision level of our double variables. This differences does not have any consequence on the corresponding simulations that gives exactly the same results in the two cases.

3. Tests and benchmarks

In a first stage, we have performed a number of tests using computational meshes with sizes that are currently adopted for our high-end production runs (512^3, 1024^3). In this way, we can easily analyze the performances that can be obtained on the Blue Gene/P architecture and compare them with the results achieved on different HPC systems. These results can be used to infer the behavior of the code on larger Jugene's configurations. The tests follow two different approaches. UNIGRID simulations are performed on a constant resolution cubic mesh, while AMR simulations adopt a grid refining method, increasing the spatial resolution where this is required, according to specific refining criteria (e.g. density jumps larger than a given threshold or in presence of strong shock waves). Both approaches can be used for the applications targeted by the present project, therefore we have analyzed their performances and suitability in detail before choosing a possible production set-up.

3.1. UNIGRID

In the UNIGRID approach a regular, constant resolution, computational mesh is used. On each cell of such mesh, fluid quantities are calculated by solving hydrodynamics conservation equations. This is a simple approach, that allows to predict exactly the total memory requested by the problem.

We have performed two series of benchmarks with the best choice of pre-compilation/compilation parameters, using a computational mesh of 512^3 or 1024^3 cells. The latter was the largest possible configuration for which the serial version of INITS (available since the beginning) could be run. Larger configurations exceed the memory of the Power6 node used for running INITS. Figure 2 shows the speed-up (left image) for the two configurations. In both cases, the performance improves with increasing the number of processors, even not linearly, giving the best result with a maximum number of processors twice the linear size of the computational mesh. This limit is intrinsic to the algorithm and depends mainly on the calculation of the gravitational field, that limits the scalability of the code to a number of processors at most 2-4 times the linear size of the mesh. The parallel FFT library used by ENZO, in fact, adopts a planar domain decomposition, that cannot scale above the 1D size of the computational grid. The performances, however, tend to improve for larger meshes (i.e. problem size), since the problem size grows much faster (depending on the cube of the linear size N_{1D}) than the number of processors, leading to an increasing per-processor workload. Furthermore, the FFT has a lower impact, its computational cost scaling with $N_{1D}\log(N_{1D})$. Such trend is confirmed by the analysis of the efficiency curves (figure 2, right image). Using a number of processors equal to the $1D$ size of the problem, efficiency for both 512^3 and 1024^3 is extremely good (≈ 0.75 the former, ≈ 0.9 the latter). It is still acceptable (≈ 0.46 and ≈ 0.65, respectively) when the number of processors is twice the linear size of the box. For larger numbers of processors the result is unacceptable. However, as can be noticed also for the speed-up, the situation tends to improve with increasing the size of the problem. Our conclusion is that, for larger configurations (e.g. 2048^3, 4096^3), an acceptable efficiency can be obtained even using a number of processors $4 \times N_{1D}$.

3.2. AMR

AMR allows to get high resolution only where this is required. AMR can achieve the same *effective* resolution of a UNIGRID run, requiring, for the physical variables, much less memory. The memory estimate, however, is

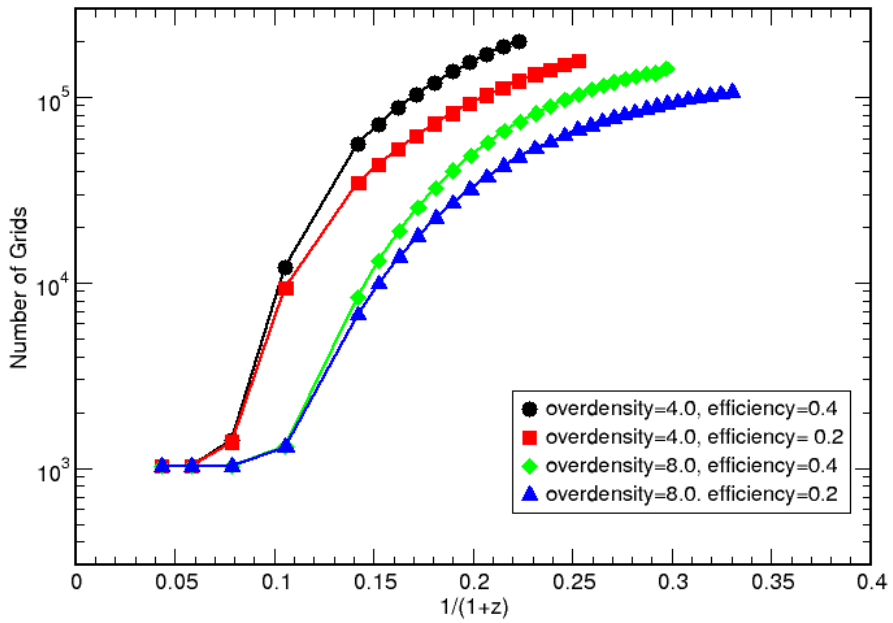

Fig. 3. Number of sub-grids generated in various AMR tests, performed changing the mass overdensity and the efficiency parameter used for the grid refinement criterion, as a fuction of the redshift, z. The redshift is a quantity used to define the cosmic time: z=0 is the current (end of the simulation) time. The curves end when the requested memory per processor overcomes the available one.

much more uncertain, since it depends on the evolution of the simulated system, on the maximum number of refinement levels allowed and on the refinement criteria.

The same holds for the computing time, that increases with the number of refined regions and with the complexity of the Hierarchical Tree (HT) structure. The latter is indeed responsible for the management of the refined regions (adding or removing regions, specifying the neighboring boxes at the same refinement level, the parent of each box, etc.). The HT is crucial in our analysis, being a data structure replicated on each processor. No parallelization is implemented on the HT, since its information is necessary to all processors for an efficient and scalable implementation of the code. This however leads to a large memory overhead, that limits the maximum size and resolution at which the simulation can be accomplished, especially when memory size of each computing node is small.

For this reason, we have focused on the growth of the HT as the simulation proceeds. Since, up to redshift $z \approx 1$, the number of AMR sub-regions rapidly increases, so does the size of the HT (starting from an initial configuration, at $z = 30$, where no sub-boxes are present). We estimated that the memory request necessary to characterize a sub-box in the HT is about 5kB.

A number of tests have been performed in order to check the size of the HT with the redshift. The different tests adopts different grid refining criteria (e.g. overdensity based, shock detection based etc.), and different parameters value for each criterion (e.g. density threshold for the overdensity criterion). Furthermore, a few choices of the maximum number of levels and of the efficiency parameter ϵ have been tested. The parameter ϵ is responsible for the size and number of the sub-boxes on a region to be refined. In figure 3, a few of these tests are presented, where the number of generated sub-box is shown as a function of the redshift, a parameter the measures the cosmic time. The number of boxes grows with redshift (hence with time) in all the cases, driven by the gravitational collapse of the cosmological structures. The different setup of the various tests leads to different HT growth rates and different balances between the number and sizes of the sub-boxes. However, in all cases, the size of the HT exceeds that of the node memory well before the end of the simulation ($1/(1+z) = 1$). We then conclude that the AMR configuration is not suitable for our application on the Jugene system.

4. Final setup: tests and benchmarks

The results of our work led to the detailed specification of the optimal case to be run on the Jugene platform, according both to the requirements of our scientific target and to the performances and features of the available computing resources.

The following conclusions can be stated:

Table 1. Memory requirements, minimum number of Jugene nodes, optimal number of cores, and wall clock time versus problem size. The minimum number of nodes is estimated such that enough memory is available for the corresponding problem and the smallest possible partition of the Blue Gene/P is allocated; the optimal number of cores is estimated considering the memory requirements and the best ENZO performance. Columns 5 and 6 show the wall clock time (in seconds) to complete a single time-step using different numbers of cores. For the 2048^3 and 4096^3 cases the values are linearly (lower value) and cubic (higher value) extrapolated from smaller cases.

N_{1D}	$\approx$ Mem (GB)	Min. Nodes	Opt. Core	$T_{\text{Opt.Cores}}$	$T_{2\times\text{Opt.Cores}}$
512	60	32	1024	21	25
1024	480	256	2048	60	70
1500	1510	1024	3000	120	130
2048	3900	2048	4096	189-217	199-220
4096	31000	16384	16384	NA	457-765

- The size of Jugene's node memory, prevents us to adopt an AMR approach, since the hierarchy tree of the whole computational box is replicated on each node. At most ≈ 150000 tree nodes (sub-grids) can be stored and managed, that are far too few to complete a cosmological run in any meaningful setup tested. UNIGRID configuration must be adopted.
- Scalability of the code in UNIGRID mode allows to use at most a number of processors $\approx$ four times the 1D size of the computational mesh.
- Memory requested by the code in UNIGRID configuration can be estimated as:

$$\text{Mem} \;=\; N_{1D} 3 \times 8 \times 15 \times 4/1024^3 \text{ GB}, \tag{1}$$

  where N_{1D} is the linear size of the mesh, the factor 8 is due to the 15 main arrays of type *double*, and 4 is a fiducial factor estimated on the base of our tests (related to auxiliary arrays replica, required by ENZO's algorithms).

The memory requirements associated to different grid sizes are presented in table 1. In the same table, we show also the minimum number of nodes necessary for each size. Nodes can be used in SMP, $DUAL$ or VN modes.

According to the scalability tests presented in section 3., the optimal configuration adopts a number of cores $\approx$ twice the linear size of the mesh. However, the usage of a number of cores four times N_{1D} is still acceptable, the efficiency improving with the size of the problem. Therefore, the 2048^3 configuration is the most effective, since it can use 2048 Jugene nodes in both DUAL and VN mode, hence exploiting half or even all the available 8192 cores, avoiding wasting of computing resources and achieving good performances. Unfortunately, in the time-frame and with the CPU resources available for the present project, we could not perform tests on this, or larger, configurations. We have estimated the CPU time needed by the 2048^3 and 4096^3 cases using both a linear and a cubic extrapolation from the smaller tests (see table 1). The former provides a sort of lower limit in the CPU time per time step per core, the latter is instead an upper bound. According to these estimates, in order to complete a run of about 1000 timesteps for a 2048^3 mesh, between 215000 and 250000 CPU hours are required, corresponding to a wall clock time between 52 and 60 hours on 4096 cores (DUAL mode). For the 4096^3 mesh only the 16384 nodes SMP mode configuration can be used. The usage of a larger number of cores, in fact, would lead to to a strong deterioration performances by ENZO. In such configuration, we estimate that between 2000000 and 3400000 CPU hours are necessary to complete a simulation. Therefore, a single run would take between 123 and 207 wall clock hours to finish.

5. Conclusions

In this work, the ENZO code has been successfully ported on the Jugene architecture. Its performances and its computational requirements have been analyzed and optimized. According to this analysis, the AMR configuration has been ruled out, being too memory demanding, due to the presence of the HT data structure that is replicated on each processor and that tends to grow as the evolution of the simulated system proceeds. The UNIGRID configuration has been accurately benchmarked and the best possible configuration for the simulations of interest identified. In particular, the proper number of nodes for computational meshes of different size was set, in order to ensure that both enough memory is available and the best performance is achieved. It has been found that such optimal number of cores is twice the linear size of the mesh. In order to generate initial conditions for the production-size runs the INITS code (initially sequential) has been parallelized. Finally we have identified the most suitable configurations to fulfil our scientific requirements matching the available HPC resources and estimating the corresponding necessary CPU time. This corresponds to a mesh size of 2048^3 cells, on 4096 cores and requires a CPU time of the order of 250000 hours.

Acknowledgements

This work was financially supported by the PRACE project funded in part by the EUs 7th Framework Programme (FP7/2007-2013) under grant agreement no. RI-211528 and FP7-261557. We thank Brian O'Shea

(Michigan State University) for the useful discussions and suggestions.

References

Borgani & Kravtsov(2009). Borgani, S., & Kravtsov, A. 2009, arXiv:0906.4370

Brunetti & Lazarian(2010). Brunetti, G., & Lazarian, A. 2010, MNRAS, 1371

Bryan & Norman (1998). Bryan G. L., Norman M. L., 1998, ApJ, 495, 80

Dolag et al.(2008). Dolag, K., Borgani, S., Schindler, S., Diaferio, A., & Bykov, A. M. 2008, Space Science Reviews, 134, 229

Ferrari et al. (2008). Ferrari, C., Govoni, F., Schindler, S., Bykov, A., & Rephaeli, Y., 2008, SSR

1. Godunov, S. K. 1959, Mat. Sbornik, 47, 271

Norman et al. (2007). Norman M. L., Bryan G. L., Harkness R., Bordner J., Reynolds D., O'Shea B., Wagner R., 2007, arXiv, 705, arXiv:0705.1556

O'Shea et al. (2004). O'Shea B. W., Bryan G., Bordner J., Norman M. L., Abel T., Harkness R., Kritsuk A., 2004, astro, arXiv:astro-ph/0403044

Vazza et al.(2009). Vazza, F., Brunetti, G., & Gheller, C. 2009, MNRAS, 395, 1333

Vazza et al.(2009). Vazza, F., Brunetti, G., Kritsuk, A., Wagner, R., Gheller, C., & Norman, M. 2009, A&A, 504, 33

Vazza et al.(2010). Vazza, F., Gheller, C., & Brunetti, G. 2010, A&A, 513, A32

Vazza et al.(2010). Vazza, F., Brunetti, G., Gheller, C., & Brunino, R. 2010, NewA, 15, 695

Vazza et al.(2011). Vazza, F., Brunetti, G., Gheller, C., Brunino, R., & Brüggen, M. 2011, AAP, 529, A17

Vazza(2011). Vazza, F. 2011, MNRAS, 410, 461

Woodward & Colella (1984). Woodward P., Colella P., 1984, JCoPh, 54, 115