Available online at www.prace-ri.eu

Partnership for Advanced Computing in Europe

Code Optimization and Scaling of the Astrophysics Software Gadget on Intel Xeon Phi

P. Borovska[a]*, D. Ivanova[a]

[a]*National Centre for Supercomputing Applications, Bulgaria*

Abstract

The whitepaper reports our investigation into the porting, optimization and subsequent performance of the astrophysics software package GADGET, on the Intel Xeon Phi. The GADGET code is intended for cosmological N-body/SPH simulations to solve a wide range of astrophysical tasks. The test cases within the project were simulations of galaxy systems. A performance analysis of the code was carried out and porting, tuning and scaling of the GADGET code were completed. As a result, the hybrid MPI/OpenMP parallelization of the code has been enabled and scalability tests on the Intel Xeon Phi processors, on the PRACE EURORA system are reported.

Keywords: Gadget Software, Galaxy Data Set, Hybrid MPI/OpenMP Parallelization, High-Performance Computing, Scalability Tests, Intel Xeon Phi

1. Introduction

Cosmic phenomena involve complex interactions between physical processes on a wide range of scales. Simulations are of great importance in this area of research. The development of such models is supported by rapid advances in computer technology, which provide a greater dynamic range of investigation in simulation models.

The software code, GADGET, is a freely available code, widely used for cosmological N-body/SPH simulations to solve a wide range of astrophysical tasks - colliding and merging galaxies, forming of large-scale structure in the space, studying the dynamics of the gaseous intergalactic medium, forming of the stars and its regulation, etc. The test case for within the project will be simulations of galaxy systems.

GADGET is written by Volker Springel to make possible the execution of cosmological N-body/SPH simulations on massively parallel computers with distributed memory.

GADGET uses an explicit communication model and is parallelized by a MPI communication interface. The cosmological code includes a tree-code module, a communication scheme for gravitational and SPH forces, a domain decomposition strategy, a novel smooth particle hydrodynamics (SPH) formulation based on entropy as an independent variable, and in addition of the “TreePM” functionality.

The objective of this project is to optimize the multithreading of GADGET along with developing the hybrid MPI/OpenMP parallelization of the code to allow the code to exploit new hybrid computing, and to perform scalability testing of the code on the Intel Xeon Phi processors on the EURORA system. This work is part of the PRACE-1IP WP7 Extension project.

* Corresponding author. *E-mail address*: pborovska@tu-sofia.bg
* Corresponding author. *E-mail address*: d_ivanova@tu-sofia.bg

2. GADGET Code Structure

The main structure of the GADGET software package is the TreeSPH code (Hernquist & Katz 1989) [2], where gravitational interactions are calculated by a hierarchical multipolar expansion and gas dynamics followed by a SPH computation. Gas and dark matter are represented by particles in this method. In the calculations for this method, collisions between the particles are not taken into account [1].

The Particle-Mesh (PM) methods, (eg Klypin & Shandarin 1983, White, Frenk & Davis 1983) is the fastest scheme for the calculation of gravitational forces, but for scales under one or two cells, the forces are strongly suppressed, as a result, this technique is not good at operations on data with a high spatial resolution. This resolution can be increased significantly by adding the direct calculations of power over short distances (Hockney & Eastwood 1981, Efstathiou et al., 1985), or using additional adaptive Fourier cells placed on areas of particular interest (Couchman 1991). The cell mesh can also be adaptively refined, with the potential found in real space using relaxation methods (Kravtsov, Klypin & Khokhlov 1997; Knebe, Green & Binney 2001) [2, 3, 4].

The Hierarchical Tree algorithm (Appel 1985, Barnes & Hut 1986, Dehnen 2000) is another method of execution and presents no limit when it comes to resolution, especially for mass distribution in which there are parts of the area with a low density of the tables, it can be considerably slower than the methods based on the Fourier transform algorithm. Developing the TreePM hybrid algorithm tries to combine the advantages of both methods by using tree algorithm calculations only at close distances and calculating forces over long distances through the PM algorithm [5, 6].

In the code of the software package GADGET2, both dark matter and gaseous fluid are represented by particles along for the two components to be calculated by the method of N-bodies. The basic method, which Gadget software package used to achieve spatial adaptively is hierarchically multipolar expansion, often called a tree algorithm. This method classifies remote particles in even bigger cells, allowing their gravity can be measured by a single multipolar power. Instead needed the N-1 interactions of particles - for each particle, as required in the method of direct summation gravitational forces on a particle can be calculated with only O (logN) interactions [6].

In practice, a hierarchical clustering, this forms the basis of a multi-extension, most often obtained by recursively partitioning the space. In the method used in hierarchical tree algorithm cubic node covering the entire mass of the problem, many times multiple of 8 knots subsidiaries each of which has its sides equal to half of the parental unit until they reach a "leaf" nodes each of which contains only one piece. The forces are calculated by browsing the tree, and deciding whether a multi-polar expansion of the unit will provide a sufficiently accurate force (which usually happens when the nodes are small and at a great distance from each other). If this is true, the power of a multi- use and crawling on the branch of the tree is terminated, and if this is not true, the daughter nodes are redrawn line by making the same check on each of them. We should pay attention to the fact that the end result of the tree algorithm will be approximately equal to the actual result. However, the error can be very easily reduced to acceptable levels by changing the criteria for crawling nodes of the tree, because the greater accuracy is achieved - it goes through the tree in greater depth.

3. GADGET Parallelization Strategy

The GADGET software package is a massively parallel simulation code. It uses MPI communication instructions. The code is written in the C programming language and uses the GSL and FFTW libraries, which are open source. As a result, this software package can be used on a wide variety of UNIX based machines, without having to use special features of proprietary compilers.

The software package, GADGET, uses a domain decomposition scheme ensuring that the results of forces will depend on the number of used processors, which is usually obtained by using orthogonal bisection in domain decomposition. It also addresses shortcomings of bisection. This scheme uses spatial filling fractal curve Peano-Hilbert to become three-dimensional space in the one-dimensional curve. This is then simply divided into parts that define the different domains. This scheme has several advantages, such as the fact that points that are close to one-dimensional curve usually close in three-dimensional space [6].

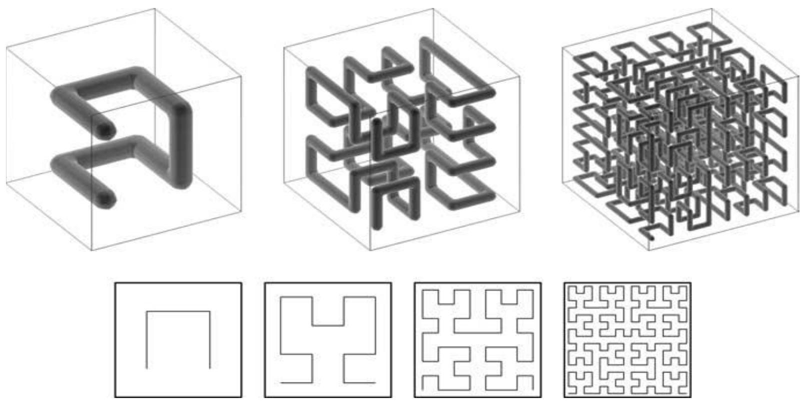

Figure 1: Space-filling curve Peano-Hilbert two (lower) and three (top) dimensions

4. Dynamic Analysis of the GADGET Code

A performance analysis of the code was necessary to determine the functions that needed to be optimized for Intel Xeon Phi. The software package chosen for the dynamic analysis of the GADGET code was the Tuning and Analysis Utilities (TAU) profiling tool [10]. The profiling tool, TAU, is a software package for analysis of high performance parallel and distributed computer systems. TAU provides a set of tools for static and dynamic analysis of programs written in C, C++, Fortran 77/90, High Performance Fortran and Java.

TAU implementation focuses on the requirements of the toolkit DOE ACTS and computer platforms ASCI. This forces the software package TAU to face the challenge of working with the constantly evolving standards of languages, large software frameworks and many different machine platforms, as well as libraries for real-time compilers. As a result, TAU provides one of the most portable and robust systems for the analysis of parallel scientific applications that currently exist. The heart of the software package TAU is its environment for profiling and tracking. It is an integrated set of tools for performance measurement and analysis of parallel multithreaded programs. The software package TAU supports integrated structure analysis, which can be expanded by a modular design of components, and also through the disclosed formats and programs for coupling to other tools. This software package allows be redirecting to a new language, development environment and system contexts and also extending with new features for analysis [10].

The dynamic analysis of the GADGET code with TAU requires some changes of the GADGET code Makefile. The first thing to do was to replace the compiler that is used with the appropriate TAU shell script. In the case of the GADGET code, which is entirely written in the C programming language with MPI for inter-processor communication, instead of the standard *mpicc* compiler we used *tau_cc.sh*. The compilation of GADGET code with TAU required to be set two variables of the Linux environment. The first environment variable is TAU_MAKEFILE. It is usually located in the lib subdirectory of the installation directory of the package and is there beginning with "Makefile." followed by the libraries that are included in the program. This file contains paths to dynamic libraries needed to compile software applications with the software package TAU. The second environment variable is TAU_OPTIONS. It is set options of the software package TAU. In this case, the following options are: -optVerbose, -optCompInst and -optMpi. The first shows that the package displays additional information when compiling. The second tells the package to instrument code using the compiler. The third package indicates that the code will use MPI communication.

The data set galaxy of the GADGET code is a simulation of a two galaxies collision. To obtain the best dynamic analysis results the code is necessary to be well optimized for standard processors. To get highly optimized code for the data set galaxy, the code was compiled with the following options:

- *#OPT += -DPERIODIC*
- *OPT += -DUNEQUALSOFTENINGS*
- *#OPT += -DPMGRID=128*
- *OPT += -DHAVE_HDF5*
- *#OPT += -DDOUBLEPRECISION*
- *#OPT += -DDOUBLEPRECISION_FFTW*

The simulation results can be visualized with the command *paraprog*, which invokes the GUI analyser of software package TAU.

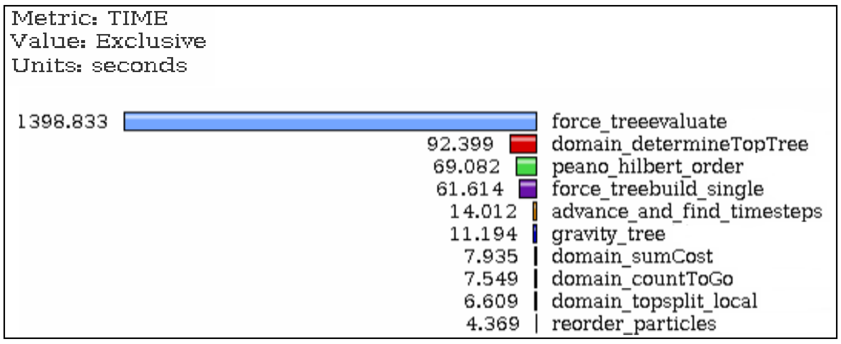

Figure 2: Dynamic analysis of the GADGET code with TAU

The dynamic analysis of the GADGET code shows that the most time-consuming function was *force_treeevaluate*, followed by the *domain_determineTopTree*, *peano_hilbert_order* and *force_treebuild_single* functions. The execution time of the *force_treeevaluate* function takes 1398 sec. and it is 15 times more than the next most time-consuming function. In order to execute the GADGET code on the EURORA supercomputer and to enable the GADGET code on Intel Xeon Phi processors we had to rewrite these parts of the code with hybrid MPI/OpenMP parallelization.

5. Experimental Framework

EURORA (**EUR**opean many integrated c**OR**e **A**rchitecture) is a heterogeneous cluster based system, located at the CINECA facility in Bologna, Italy. It was used to obtain the results presented in this paper.

The system is made up of 64 Intel compute nodes. Half of the compute nodes comprise two eight-core Intel Xeon E5-2658 processors the other half comprise two eight-core Intel Xeon E5-2678W processors [9].

58 of the nodes have 16 GB of memory but only 14 GB of this can be safely allocated by the application due to system overheads. The remaining 6 nodes have 32 GB of memory. 32 of the EURORA compute nodes have two NVIDIA Tesla K20 (Kepler) GPU cards attached with the remaining 32 compute nodes having two Intel Xeon Phi 5110P co-processors instead [9].

Each Xeon Phi card contains 60 physical cores running 4 virtual threads per core giving access to a total of 240 threads per card. For more details on the hardware specifications please see [8, 9].

For the case of scalability testing on Intel Xeon Phi processors, galaxy data set of the software package GADGET is used.

6. Hybrid Code Implementation on Intel Xeon Phi

The original GADGET code is written in standard ANSI C and uses an explicit communication model, parallelized by the MPI communication interface. The code execution required some external libraries [6, 7, 8]:

- GSL is a library provides a wide range of mathematical routines such as random number generators, special functions and least-squares fitting. There are over 1000 functions in total with an extensive test suite.

  ```
  CC=mpiicc CFLAGS="-O3 -mmic -mt_mpi" ./configure --host=x86_64 --
  prefix=/gpfs/scratch/userexternal/pborovsk/install/gsl_mic
  make
  make install
  ```

- ZLIB is a software library used for data compression. Zlib compressed data is typically written with a gzip or a zlib wrapper. The wrapper encapsulates the raw DEFLATE data by adding a header and trailer. This provides stream identification and error detection that are not provided by the raw DEFLATE data.

  ```
  CC=mpiicc CFLAGS="-O3 -mmic -mt_mpi" ./configure --
  prefix=/gpfs/scratch/userexternal/pborovsk/install/zlib_mic
  make
  ```

- SZIB is compression software, providing lossless compression of scientific data, has been provided with HDF software products.

```
CC=mpiicc CFLAGS="-O3 -mmic -mt_mpi" CXX=mpiicpc CXXFLAGS="-O3 -mmic -
mt_mpi" ./configure --host=x86_64 --
prefix=/gpfs/scratch/userexternal/pborovsk/install/szip_mic
make
make install
```

- HDF5 is a data model, library, and file format for storing and managing data. It supports an unlimited variety of data types, and is designed for flexible and efficient I/O and for high volume and complex data. HDF5 is portable and is extensible, allowing applications to evolve in their use of HDF5.
- FFTW is a C subroutine library for computing the discrete Fourier transform (DFT) in one or more dimensions, of arbitrary input size, and of both real and complex data.

```
CC=mpiicc CFLAGS="-O3 -mmic -mt_mpi" ./configure --host=x86_64 --disable-fortran --
enable-mpi --enable-type-prefix --prefix=/gpfs/scratch/userexternal/pborovsk/install/fftw_mic
make
make install
make clean
CC=mpiicc CFLAGS="-O3 -mmic -mt_mpi" ./configure --host=x86_64 --disable-fortran --
enable-mpi --enable-type-prefix --enable-float --
prefix=/gpfs/scratch/userexternal/pborovsk/install/fftw_mic
make
make install
```

The dynamic analysis of the GADGET code identified the most time-consuming function, *force_treeevaluate*. The code review of this function shows that the function receives an integer variable called `target` and calculates the gravitational forces of the particle index `target`, and depending on the variable mode - or the particle is local to the CPU or the communication buffer (the function is called twice in the function gravity_tree - once for mode 0 in which the calculations are made to the local processor for particles, and then to mode 1 in which the calculations are made of the particles in the communications buffer). The GADGET code implementation on Intel MIC requires hybrid granularity of parallelization.

7. Scalability Testing

The goal of this project is to optimize the multithreading of GADGET software with developing the hybrid MPI/OpenMP parallelization of the code and to perform the scalability testing of the code on the Intel Xeon Phi processors on the EURORA system.

The executions are performed on Intel MIC using different MPI/OpenMP configurations with maximum 240 threads on the MIC in native mode. The results are shown in Table 1.

Number	MIC	Total thread number per MIC	MPI tasks per MIC	Threads per MPI per MIC	Test (Data Set)	Total execution time [s]	Speed up
1	1	60	1	60	galaxy	1750	1
2	1	120	1	120	galaxy	1490	1.17
3	1	180	1	180	galaxy	1450	1.21
4	1	240	1	240	galaxy	1490	1.17
5	1	60	2	30	galaxy	1420	1.00
6	1	120	2	60	galaxy	1190	1.19
7	1	180	2	90	galaxy	1160	1.22
8	1	240	2	120	galaxy	1200	1.18
9	1	60	3	20	galaxy	1570	1.00

10	1	120	3	40	galaxy	1300	1.21
11	1	180	3	60	galaxy	1200	1.31
12	1	240	3	80	galaxy	1300	1.21
13	1	60	4	15	galaxy	1690	1.00
14	1	120	4	30	galaxy	1390	1.22
15	1	180	4	45	galaxy	1360	1.24
16	1	240	4	60	galaxy	1410	1.20
17	1	56	8	7	galaxy	1700	1.00
18	1	120	8	15	galaxy	1410	1.21
19	1	176	8	22	galaxy	1380	1.23
20	1	240	8	30	galaxy	1360	1.25
21	1	48	16	3	galaxy	1640	1.00
22	1	112	16	7	galaxy	1360	1.21
23	1	176	16	11	galaxy	1310	1.25
24	1	240	16	15	galaxy	1240	1.32
25	2	60	2	30	galaxy	1390	1.00
26	2	120	2	60	galaxy	1140	1.22
27	2	180	2	90	galaxy	1060	1.31
28	2	240	2	120	galaxy	1010	1.38

Table 1: Execution times in [s] for a galaxy data set of the software package GADGET with vary number of MPI tasks and threads per Intel MIC

In all the experimental tests, the thread scatter affinity is used. Scatter affinity means that threads are placed on cores in a round-robin fashion. The process continues until all threads are mapped to a core. This process results in a balanced amount of threads on each core which mostly removes the situation in which one or a small amount of cores are working while the rest are idle.

Some experiments have been carried out utilizing the hybrid parallel program implementation of GADGET code. Galaxy has been used as experimental data. The objective of the experiments was to measure parallel performance parameters of the specified parallel application for cosmological N-body/SPH simulations.

The speedup is evaluated as a ratio of the execution time running within 60 threads relative to the execution time on 120, 180 and 240 threats per MIC with different number of MPI processes respectively. The experimental results for the speedup of GADGET code on Intel MICs using various numbers of threads and MPI processes are shown in Table 1 and Figure 4.

In all the tests above the GADET code was compiled with the optimization level O3, which for MIC accelerators means the compiler’s auto vectorization is enabled. Better results might be achieved with manual adjustments to the code for better vectorization but the amount of work and necessary time to require for it compared to the possible gain in speedup did not warrant and attempt.

The speedup results are shown and illustrated in Table 1 and Figures 4 and 5.

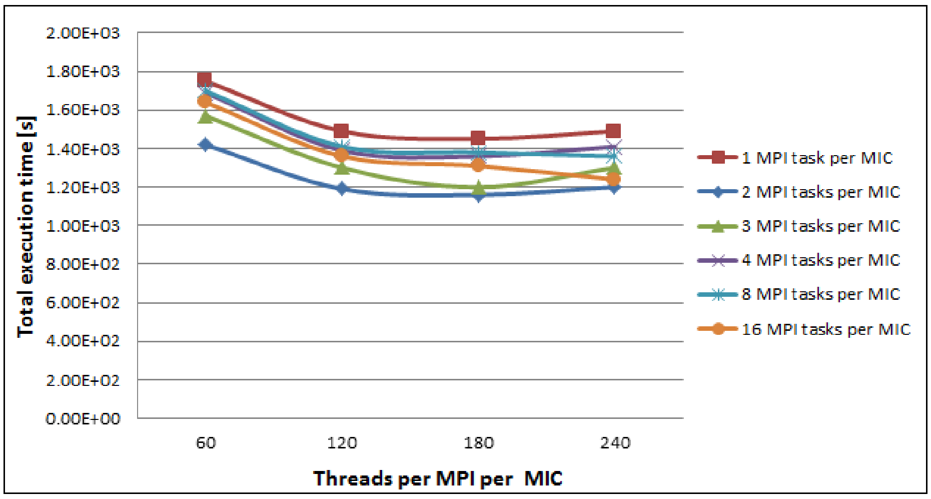

Figure 3: Execution time of GADGET code on Intel Xeon Phi

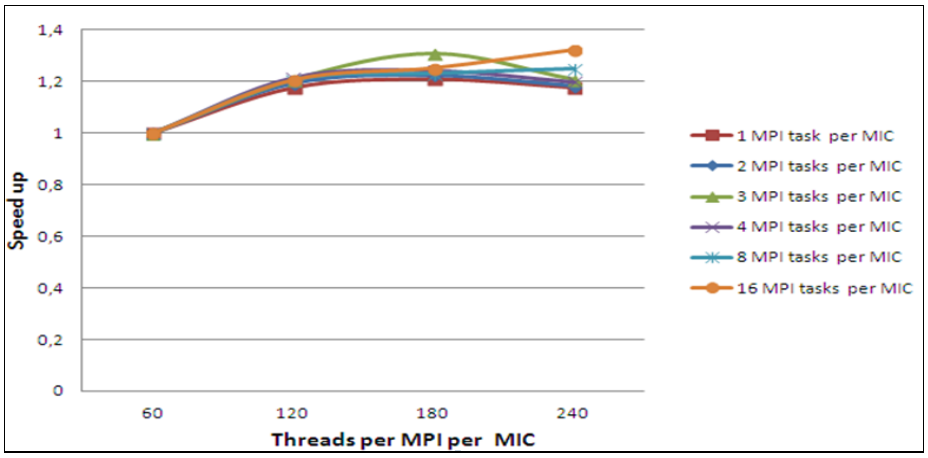

Figure 4: Speedup of GADGET code on Intel Xeon Phi

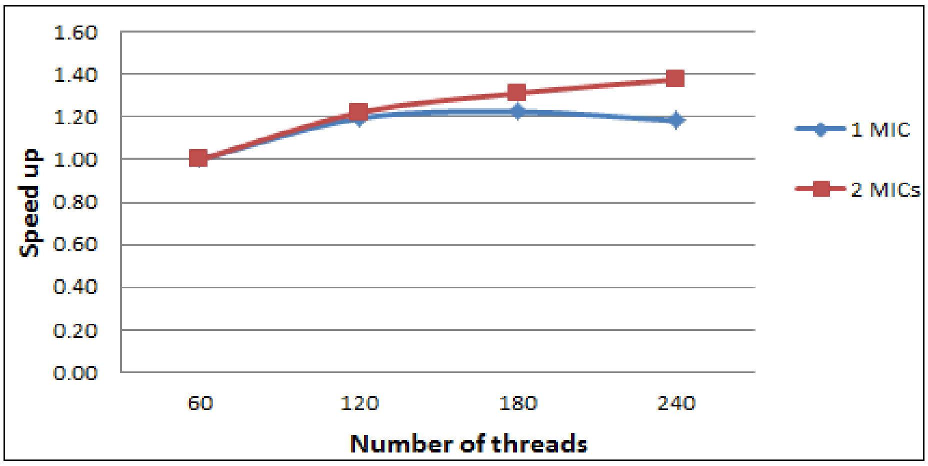

Figure 5: Speedup of GADGET code on multiple Intel Xeon Phi with 2 MPI tasks per co-processor

The hybrid MPI/OpenMP implementations on multiple MICs achieve reduction in execution time. The speedup results show that the parallel system scales on the multiple co-processors. The higher scales are for the cases up to 120 treads. The experimental part on Intel Xeon Phi is part of the project 2010PA1487, Type C, part of PRACE 1IP - WP7 Extension.

8. Conclusion and Future Work

The hybrid MPI/OpenMP parallelization of the GADGET code for cosmological N-body/SPH simulations has been developed and investigated. It is best suited to work in homogenous native Xeon Phi MPI mode in a combination of MPI tasks and threads. Porting the developed hybrid parallel code of GADGET software is very straightforward as Intel support a wide range of programming models, including the widely use MPI and OpenMP as well as Intel-specific models like TBB, Cilk+ and others. The future work is to provide additional tests in offload mode.

Performance metrics, such execution time and speedup, have been measured. The performance measurements for the galaxy data sets of the GADGET code show that the hybrid parallel implementation utilizing MPI and OpenMP of the software scales as the number of the cores increases. The future work is to demonstrate the experimental results of GADGET code for different data sets on Intel Xeon Phi in order to provide a complete scalability analysis.

The hybrid GADGET code can be apply for other similar research projects and experiments in the field of cosmological N-body/SPH simulations and will allow researchers to conduct their experiments on even more powerful supercomputers. They will be able to perform cosmological simulations with very large amounts of data.

The code can be found at http://81.161.243.12/bgmoodle/course/index.php?categoryid=46. Registration is required for download.

References

[1] Volker Springel, The cosmological simulation code GADGET-2, Mon.Not.Roy.Astron.Soc. 364 (2005) 1105-1134
[2] Hernquist, Lars; Katz, Neal, TREESPH - A unification of SPH with the hierarchical tree method, Astrophysical Journal Supplement Series (ISSN 0067-0049), vol. 70, June 1989, p. 419-446.
[3] Klypin A. A., Shandarin S. F., 1983, MNRAS, 204, 891
[4] White S. D. M., Frenk C. S., Davis M., 1983, ApJ, 274, L1
[5] GADGET-2 Best Practices, HPC Advisory Council http://www.hpcadvisorycouncil.com/pdf/GADGET-2_Best_Practices.pdf
[6] http://www.mpa-garching.mpg.de/gadget/
[7] James Jeffers, James Reinders, Intel Xeon Phi Coprocessor High Performance Programming, ISBN-13: 978-0-12-410414-3, Morgan Kaufmann Publisher, Feb 2013
[8] http://software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-codename-knights-corner
[9] http://www.cineca.it/en/content/EURORA
[10] http://www.cs.uoregon.edu/research/tau/home.php

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-261557. The project was realized using the EURORA System at CINECA, Italy.