IBERGRID

6th IBERIAN Grid Infrastructure Conference Proceedings

Lisbon, Portugal
November 7th-9th
2012

LIP

II IBERGRID

Editors

Gonçalo Borges
Laboratório de Instumentação e Física Experimental de Partículas

Jorge Gomes
Laboratório de Instumentação e Física Experimental de Partículas

Isabel Campos Plasencia
Instituto de Física de Cantabria – CSIC, Santander

IBERGRID

Laboratório de Instrumentação e Física Experimental de Partículas (LIP)
Avenida Elias Garcia, 14, 1°
1000-149 Lisboa
Portugal

Design
Carlos Manuel, Laboratório de Instrumentação e Física Experimental de Partículas (LIP)

ISBN
978-989-98265-0-2

Legal Deposit
354254/13

IV IBERGRID

Session II: Cloud Activities

New Computational Developments in Cosmology

Miguel Cárdenas-Montes**[1], Juan Jose Rodríguez-Vázquez[1], Rafael Ponce[1],
Ignacio Sevilla[1], Eusebio Sánchez[1], Nicanor Colino[1], and
Miguel A. Vega-Rodríguez[2]

[1] CIEMAT, Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas,
Departamento de Investigación Básica
Avda. Complutense 40, 28040, Madrid, Spain.
[2] ARCO Research Group, Dept. Technologies of Computers and Communications,
University of Extremadura, Escuela Politécnica, Campus Universitario s/n, 10071,
Cáceres, Spain.

Abstract. Presently cosmology is suffering a deluge of data provided by the new telescopes and other facilities. These data need to be analysed, compared and simulated in a faster and more precise mode than in the recent past. For this reason, new computational platforms are being incorporated to the common practice of this discipline. Among other things, larger and more complex codes need more efficient platforms to reduce as much as possible the execution time. In this paper, examples of adaptation to many core, cluster and cloud computing platforms are presented. Concerning the cosmological problems, two cases are in-depth described. The first case corresponds to the creation of simulated universes in the VENUS-C cloud infrastructure; whereas the second case presents the porting of the Two-Point Angular Correlation Function to a heterogeneous platform with NVIDIA GPU's. By creating simulated universes, the statistical properties of galaxy distributions can be measured, and then compared with the real one. As a consequence, the prediction of different cosmological models can be tested. On the other hand, the Two-Point Angular Correlation Function is a statistical estimator to test large-scale structures —spatial distribution of galaxies— and therefore the predictions of the ΛCDM model. This function is computationally intensive, and taking into account the expected increment in the data volume, any improvement in the performance of the analysis will be productive. In all cases, the constraints, requirements and benefits obtained after the adaptation process are stated.

1 Introduction

New telescopes and facilities are producing high volumes of high-quality data which have to be sifted, analysed, stored and delivered to the community. So, computational methods in cosmology are becoming more demanding on resources and for this reason they are focussing on any technology able to fulfil the computing requirements of the open problems of the discipline. Case examples of Cloud Computing and GPU Computing are presented in this article.

** e-mail of corresponding author: miguel.cardenas@ciemat.es

The first example corresponds to the simulation of matter fluctuations in the early universe over a cloud computing platform (VENUS-C [1]); whereas the second case corresponds to the adaptation of the Two-Point Angular Correlation Function to GPU. In both cases, the drawbacks and the benefits obtained are reported.

From the cosmological point of view, both are state-of-the-art problems, with numerous initiatives trying to cover them. From the computational point of view, both problems are challenging and they require new strategies in order to foster new developments.

2 Cloud computing to simulate matter fluctuations

The inflationary ΛCDM[3] model[2] is now the most accepted description of the early universe. This description implies that 96% of the matter-energy budget is in the form of dark matter and dark energy; whose true nature remains unknown[4]. The ΛCDM model makes accurate predictions about the fluctuations in the early universe. Those fluctuations should take the form of a Gaussian random field, with non-linear structures for small scales (less than 10 Mpc). Cluster and supercluster of galaxies are consequences of this non-linearity. Oppositely, on large scales became nearly linear.

Galaxy clustering is the most straightforward probe of Large-Scale Structure (LSS). By measuring the statistical properties of the galaxy distribution, predictions of different cosmological models and properties of dark matter and dark energy can be tested.

LSS analysis requires a large amount of simulated universes to fairly compare with real galaxy distribution. For each cosmological model, diverse realizations are needed to study and reduce the error of fitted parameters.

In order to accurately describe the distribution of matter in the universe, a Legendre polynomial series up to 2,000 degrees has to be developed for each sampled point (Fig. 1). For lower degrees, Baryonic Acoustic Oscillation (BAO) become unobservable. Due to this need of high degree development, the code becomes computationally very intensive.

Moreover, for each set of cosmological parameter studied, it is necessary to simulate the structure of the universe from early times (around one half of the current universe age) to the present. This is usually done in several time slices, and each slice needs an independent calculation.

A rough estimation of the execution time indicates that the code takes from 2 to 2.5 minutes for each spatial point for the typical processor currently in the market. By projecting the execution time for a 1/8 of the sky with a grid of points of 512 x 512 will take approximately 1.2 years; and almost 10 years for the complete sky. The previous reasoning is just for one slice of time of one model. This gives clues about the volume of the task confronted.

---

[3] Lambda Cold Dark Matter.

[4] The quantification of the budget between ordinary and dark components in the universe is a major issue as proven by the recognition of the Science magazine in 1998 and 2003 as "Scientific Breakthrough of the Year".

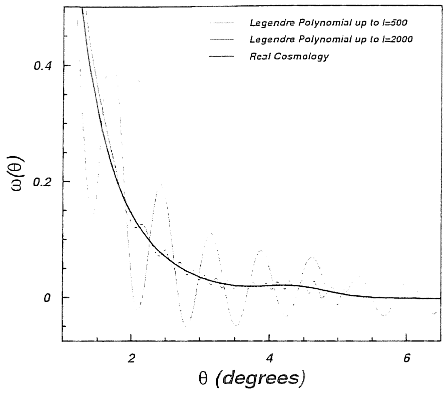

Fig. 1. Comparison between approximations. The black line shows the full calculation of the correlation function, the pink line shows the expansion up to l=500 and the red line shows the expansion up to l=2000. The BAO peak is not visible when the expansion is truncated at l=500, and at least l=2000 must be used if one wants to recover it.

2.1 The VENUS-C platform and the COMPSs Programming Framework

The VENUS-C is a FP7 project funded by the European Union whose main objective is the development and deployment of an industrial-quality service-oriented cloud computing platform for research and industry communities. This one follows the paradigm of PaaS (Platform as a Service), hiding the complexity of a grid while reducing costs at the same time. The user interacts with the endpoint, a web service interface based on Open Grid Forum's standards like BES (Basic Execution Service) and JSDL (Job Submission Description Language), which acts as a multiplexer in charge of enacting the job on the appropriate cloud platform. Currently, VENUS-C supports two models: COMPSs (COMP Superscalar, Fig. 2) [3] and a customized Azure's Generic Worker implementation; both of them can manage dynamically the resources of the underlying virtualization platform as needed. Regarding storage, a shared repository is available both for applications and data and can be accessed by the enactment service as well as by the running applications on the cloud; data management in VENUS-C is carried out using the CDMI [4] specification although COMPSs also supports FTP.

Concerning COMPSs, this programming framework facilitates the development of scientific applications for several cloud environments such as: Amazon EC2 [5], Windows Azure [6], EMOTIVE [7], OpenNebula [8] and other OCCI-compliant ones; in addition, supercomputers are also supported. The programming model itself and the runtime system comprise this middleware, which conceals the inherent details each distributed platform has from the programmer.

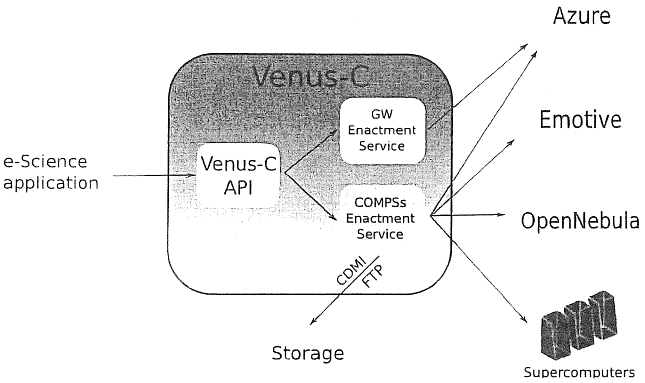

Fig. 2. Architecture of the VENUS-C middleware.

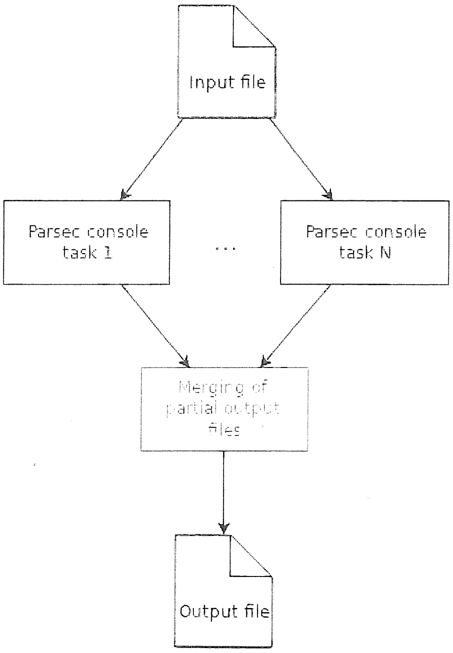

Fig. 3. Workflow of the PARSEC application.

2.2 Methodology

In order to test the VENUS-C platform, and given the good expectations regarding scalability, a computationally intensive cosmological application was chosen. Since this code generates maps of matter distribution in simulated universes, the general idea was to split the portion of the sky assigned into several sub-areas and run each one on one processor's core. For that, COMPSs requires that the programmer defines the three parts which comprises the application: the main code, the remote method/s and the annotated interface. The first approach consisted of developing the entire application in Java (this requirement about the language was imposed by the framework). This idea was disposed shortly after due to the difficulties of finding the appropriate version of the GSL (GNU Scientific Library) in Java. Taking into account that GSL is available in C/C++, the second approach resulted in a hybrid application; the application's core, which is in charge of the heavy calculations, was programmed in C++ and the rest of the application in Java. The main application's code (the sequential part) is responsible for parsing the command line arguments, controlling the execution time, calling the remote method and merging the partial results into one final output file (Fig.3). Concerning the remote method (defined in another class), the code is very simple: it only calls the C++ binary with the corresponding parameters: input file, sky coordinates, etc. Finally, the annotated interface tells COMPSs about the method which is going to be run remotely, the class it belongs to, as well as other metadata: number, type and direction of the method's parameters. With this information, COMPSs is able to intercept each call to the remote method from the main code, and launch it as a parallel task in the platform.

In addition to this application, a COMPSs client was developed as well so as to make possible for the user to interact with the enactment service. Regarding data management, the COMPSs client used CDMI at first as the protocol for staging in and out the input and output files; but later, it was changed to FTP since it proved to be faster, more reliable and, since ftp clients are ubiquitous, there is no need for an special client to transfer the files.

2.3 Results

During the exploitation phase, two major constraints were found: the maximum number of cores (20 cores) and the wall time (5 hours). Although a maximum of 20 cores were available, due to the elastic nature of the resources policy, only up to 16 cores were simultaneously allocated.

The high density required by the spatial grid carried out that the highest map size allocatable was: 32×32 points with variables ϕ ranging (0,0.1) and $cos(\theta)$ ranging (0,0.1)[5]. With those constrains, the mean execution time takes 10,155s (roughly 2h49m). A example of this kind of maps is presented at Fig. 4.

During the exploitation phase, the infrastructure demonstrated to be sufficiently robust and stable to support the scientific production. Unfortunately, the

---

[5] ϕ and θ are the angular components of the spherical coordinates.

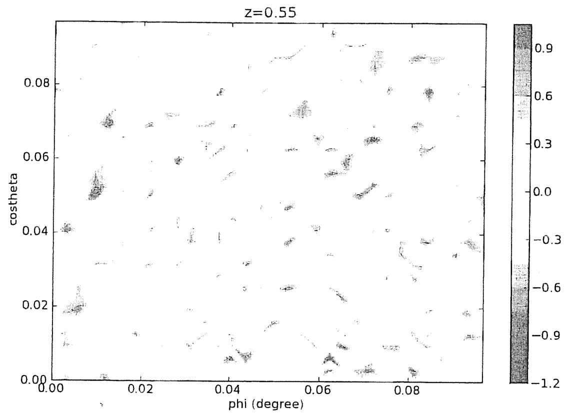

Fig. 4. Example of matter fluctuation map generated in VENUS-C infrastucture.

Table 1. Execution time for MPI implementation and Cloud implementation.

Grid	ϕ	$cos(\theta)$	Sky Portion	Cores (Processors)	Time
512X512	(0,π/2)	(0,1)	1/8	256 (64)	1,468m32.290s
256X256	(0,π/4)	(0,1)	1/16	128 (16)	734m19.466s
Cloud	ϕ	$cos(\theta)$	Sky Portion	Cores (Processors)	Time
32X32	(0,0.1)	(0,0.1)	1/1,600	16 (4)	10,155s

limitations in the number of nodes and the wall-time make able to produce maps covering a tiny area (roughly 1 part in 1,600 of the total sky).

For comparison purposes, a MPI version of this code was created and tested in Euler facility at CIEMAT. This cluster is composed by 144 nodes with two Quad-Core Xeon processors at 3.0GHz with 8 GB. In Table 1, the results for two alternative configurations executed in Euler are presented. In both cases, a similar density of sample points that in the cloud execution has been implemented. The results show that a more relevant sky area can be simulated, but as a counterpart a higher volume of computational resources must be mobilized.

2.4 Conclusions

The numerical experiments performed at the cloud infrastructure of the VENUS-C project have demonstrated its ability for supporting small-scale simulations of the problem proposed. Robustness and reliability shown by the cloud infrastructure of VENUS-C have been worthwhile features. In the near future, large-scale tests are being scheduled in coordination with the project.

3 Two-Point Angular Correlation Function

As was mentioned previously, the distribution of galaxies in the universe is one of the most important probes for cosmological models. Particularly, the most relevant observable to study this distribution is the Two-Point Angular Correlation Function (TPACF), which is a measure of the probability excess, relative to a random distribution, of finding two galaxies separated by a given distance or angle.

Roughly, the TPACF quantifies the likelihood, over random, to find a source within a given angle of other source. When using twice the same population of galaxies, the outcome is an autocorrelation function, whereas if two different sources are used, then is a correlation function. In both cases, the usual presentation is through a histogram of the correlation function versus the angle or the distance between the galaxies.

The calculation of the TPACF is computationally demanding, specially for large surveys that are currently taking data or about to start, which measure tens or hundreds of millions of galaxies. Taking into account the computational intensity of the analysis, diverse implementations (CUDA, MPI and hybrid MPI-CUDA) are tested.

3.1 TPACF

The TPACF, $\omega(\theta)$, is a measure of the excess or lack of probability of finding a pair of galaxies under a certain angle with respect to a random distribution. In general, estimators TPACF are built combining the following quantities:

- DD(θ) is the number of pairs of galaxies for a given angle θ chosen from the data catalogue D.
- RR(θ) is the number of pairs of galaxies for a given angle θ chosen from the random catalogue R.
- DR(θ) is the number of pairs of galaxies for a given angle θ taking one galaxy from the data catalogue D and another from the random catalogue R.

Although diverse estimators for TPACF do exist, the estimator proposed by Landy and Szalay [9], (Eq. 1), is the most widely used by cosmologists due to its minimum variance.

$$\omega(\theta) = 1 + \left(\frac{N_{random}}{N_{real}}\right)^2 \cdot \frac{DD(\theta)}{RR(\theta)} - 2 \cdot \left(\frac{N_{random}}{N_{real}}\right) \cdot \frac{DR(\theta)}{RR(\theta)} \qquad (1)$$

In Eq. 1, N_{real} and N_{random} are the number of galaxies in the data and random catalogues respectively.

A positive value of $\omega(\theta)$ —estimator of TPACF— indicates that galaxies are more frequently found at angular separation of θ than expected for a randomly distributed set of galaxies. On the contrary, when $\omega(\theta)$ is negative, a lack of galaxies in this particular θ is found. Consequently $\omega(\theta) = 0$ means that the distribution of galaxies is purely random.

The calculation of TPACF implies computing the angle among all pairs in a sample of N galaxies. As a consequence, the complexity of the calculation is $O(N^2)$.

3.2 GPU Implementation

Concerning the GPU implementation, several strategies have been considered. First of all, an intense use of the *shared memory* for the intermediate calculations has been performed. For example, the dot product and the arc-cosine calculations are fully implemented in *shared memory*, avoiding the read-and-write in *global memory* —much slower than *shared memory*—. However, the true bottleneck is the construction of the histogram. Until this point of the kernel, the multithreaded code is calculating in parallel angles and, at this point, it must assign them to the correct bin in the histogram. Due to the multithreaded nature of the kernel, it must be avoided that two or more threads modify the same bin simultaneously. For this reason, atomic operations are compulsory to create the histogram; and specifically the *atomicAdd* function.

The atomic functions for integers in CUDA are supported in devices with compute capability 1.1 or higher in *global memory*; and with compute capability 1.2 or higher in *shared memory* [10]. For floating point numbers, compute capability 2.0 or higher is necessary.

In our implementation, the use of atomic functions in *global memory* causes a major performance degradation. Therefore, the solution is to perform more atomic operations in parallel. For this reason, an alternative strategy has been followed. Each block of threads implements a histogram in *shared memory*; and in these histograms, angles are binned in parallel. Next, all shared-memory-built-histograms are reduced to a single one in *global memory*. This strategy produces a parallel treatment of the most critical operation, being the foundation of the success of the GPU implementation.

On GTX295 hardware, the maximum amount of *shared memory per block* is limited to 16 KB. So, for an efficient execution, the thread block size in our implementation is limited to 256 threads. This limitation of 256 threads per block leads to a total number of bins in the histogram of 256. This number is enough for the foreseen cosmological analysis.

Besides, the code was created with coalesced global memory access. Coalesced access maximizes global memory bandwidth usage by reducing the global memory transactions.

3.3 Data Origin

The data used in this work have been extracted from the MICE simulations [11] [12]. The MICE (Marenostrum Institut de Ciencies de l'Espai) simulations has been produced at the Marenostrum facility of Barcelona Supercomputing Center. The file employed in this work is composed of $4.3 \cdot 10^5$ galaxies.

3.4 Comparative Results

One of the main objectives of this work is to produce an implementation for the TPACF calculation, which can be executed on GPU hardware. The GPU election is based on the capability to accelerate the execution as well as the low price of the hardware. This makes high performance computing accessible for small research groups. Additionally the results of GPU implementation, more comparative results with other implementations are presented.

In order to test the capacity of the GPU implementation to accelerate the TPACF, comparative tests are executed among CPU, GPU and multi-GPU —using 3 GPUs—; and OMP, MPI and hybrid MPI-CUDA implementation. The OpenMP implementation is executed on the machine holding the GPU cards. This machine has two Intel Xeon E5520 Quad-Core processors, and the numerical experiments were executed with hyperthreading activated.

The results of the comparisons among the implementations are presented at Table 2. The analysis of these results shows the excellent speed-up obtained by the GPU implementation —115.15±4.84— in opposition to the speed-up obtained by the OpenMP implementation —10.58±0.48—. It should be underlined that the GPU implementation is fast enough to be competitive, and for this reason, to be used by small research groups lacking in more expensive equipment such as clusters or supercomputers.

Table 2. Mean execution time and speedup for CPU, GPU, Multi-GPU and OpenMP implementations.

Implementation	Mean execution time (s)	Speedup
CPU	35,186.327 ± 1,452.419	
OpenMP	3,326.363 ± 28.498	10.580±0.478
GPU	305.570 ± 1.143	115.154±4.835
Multi-GPU	184.108 ± 2.127	191.174±8.897

Besides, the previous implementations, an MPI one is also created and tested. In Table 3, the execution time and the speedups of the MPI implementation are shown. This numerical experiment was performed in a cluster composed by 144 nodes with two Quad-Core Xeon processors at 3.0GHz with 8 GB.

As can be appreciated, the behaviour of the speedup is close to theoretical maximum speedup for small and mid-size configurations (up to 32 cores). As far as the number of cores increases, the speedup deviates from the theoretical maximum obtaining a poor performance for 512 cores. This degradation of the performance occurs because communication time becomes relevant in comparison with computing time. Larger input files mitigate this degradation by the increment of computing time in comparison with communication time. A similar effect will be observed later with the hybrid MPI-CUDA implementation.

In any case, as the future expected input size is much higher that the sample file employed in this work as benchmark, the severity of this effect will be drifted towards larger number of nodes in both implementations.

Finally, taking into consideration the large input files expected (up to two orders of magnitude larger than the sample used in this work), a hybrid MPI-CUDA implementation is also created to cope with those executions. It has to be

Table 3. Mean execution time and speedup for diverse numbers of cores in the MPI implementation.

Cores	Mean execution time (s)	Speedup
1	23,712.281 ± 145.690	
2	11,652.683 ± 59.959	2.035 ± 0.014
4	6,327.005 ± 68.821	3.748 ± 0.047
8	3,162.222 ± 34.165	7.499 ± 0.084
16	1,577.899 ± 52.818	15.046 ± 0.562
32	799.457 ± 11.793	29.667 ± 0.455
64	403.937 ± 6.044	58.716 ± 0.965
128	205.008 ± 4.262	115.713 ± 2.423
256	104.040 ± 6.493	228.796 ± 14.210
512	71.292 ± 2.841	333.090 ± 12.301

underlined that even the GPU implementation forecasts execution time of days of files of tens of millions of galaxies.

In Table 4, the execution times for two input files MICE 0.35 (430,931 galaxies) and MICE 0.55 (654,094 galaxies) are shown. This production has been executed on a cluster with 8 nodes, each one with a C2050 card. The different sizes of files will allow compare the execution times when supplying different computational charge to each node.

The analysis of these results shows an excellent speedup in most of the cases, except for the MICE 0.35 file when executing on 8 nodes. In this case the communication time becomes relevant in comparison with the calculation time. Fortunately, this tiny degradation of the performance will become irrelevant when handling files with tens of millions of galaxies, as can be appreciated when analysing the file MICE 0.55. This second file is, mildly speaking, a 50% larger than the MICE 0.35. As an example, the comparative results for 8 nodes show a higher speedup the analysing the file MICE 0.55 than MICE 0.35. Essentially, for 8 nodes the execution time for each chunk in the later case is relevant in comparison to the communication time, where as for the largest files is less relevant.

Table 4. Mean execution time (s) for the single precision MPI-CUDA implementation for 1, 2, 4, and 8 nodes for MICE 0.35 (430,931 galaxies) and MICE 0.55 (654,094 galaxies).

	MICE 0.35		MICE 0.55	
Nodes	Execution Time	Speedup	Execution Time	Speedup
1	301.473±0.232		711.702±0.474	
2	151.622±0.498	1.988	349.279±0.506	2,037
4	78.907±0.461	3.821	181.331±0.073	3,925
8	43.273±0.037	6.967	94.274±0.035	7,549

3.5 Conclusions

In this paper, two proposals to deal with the new computational challenges faced by the cosmologists are presented. On the one hand, a proof-of-concept in order to validate the capacity of the Venus-C's cloud infrastructure in relation to the cosmologists' requirements. On the other hand, the performance of diverse implementations of the two-point angular correlation function are compared.

Considering the first problem implemented, the Venus-C's cloud infrastructure has proven to be very robust and reliable at running small-scale cosmological simulations. For this reason, larger-scale tests on this platform are being considered for the short-term future in order to produce more significant simulations.

On the two-point angular correlation problem, it has to be underlined the excellent performance provided by the GPU implementation. When comparing with other implementations, in most of the cases it performs better. Only when using a large number of cores (MPI implementation with more than 128 cores), the GPU implementation is outperformed. However, still the GPU hardware is still less expensive than the equivalent cluster. Considering the hybrid MPI-CUDA implementation, it is expected to be the main tool for the analysis of very large galaxy surveys, and for this reason it has been considered in this work.

Finally, it must be mentioned that a public version of the GPU implementation is freely available at http://wwwae.ciemat.es/cosmo/gp2pcf/. In spite of the short time from its release, researchers already have downloaded and tested the code. Suggestions have been received to implement new functionalities in the code.

Acknowledgements

This work has been partially funded by the European Community's Seventh Framework Programme (FP7/2007-2013) via the project VENUS-C under the grant agreement number RI-261565.

We acknowledge the use of data from the MICE simulations, publicly available at http://www.ice.cat/mice

References

1. VENUS-C: Virtual multidisciplinary ENvironments USing Cloud infrastructures. http://www.venus-c.eu
2. Frieman, J., Turner, M., Huterer, D.: Dark Energy and the Accelerating Universe. (2008)
3. Lezzi, D., Rafanell, R., Lordan, F., Tejedor, E., Badia, R.M.: Compss in the venus-c platform: enabling e-science applications on the cloud. 5th Iberian Grid Infrastructure Conference (IBERGRID 2011) (Jun 2011) 73–84
4. CDMI: Cloud Data Management Interface. http://www.snia.org/cdmi
5. Amazon Elastic Compute Cloud. http://aws.amazon.com/ec2/
6. Windows Azure. http://www.windowsazure.com/en-us/
7. EMOTIVE: Elastic Management Of Tasks In Virtualized Environments. http://www.emotivecloud.net

8. OpenNebula. http://www.opennebula.org/
9. Landy, S.D., Szalay, A.S.: Bias and variance of angular correlation functions. American Journal of Physics **412** (July 1993) 64–71
10. Sanders, J., Kandrot, E.: CUDA by Example: An Introduction to General-Purpose GPU Programming. 1 edn. Addison-Wesley Professional (July 2010)
11. Crocce, M., Fosalba, P., Castander, F., Gaztañaga, E.: Title: Simulating the Universe with MICE: The abundance of massive clusters. Mon.Not.Roy.Astron.Soc. **403** (2010) 1353–1396
12. Fosalba, P., Gaztañaga, E., Castander, F., Manera, M.: The onion universe: all sky light-cone simulations in shells. Mon.Not.Roy.Astron.Soc. **391** (2008) 435–446