Performance enhancement of a GIS-based facility location problem using desktop grid infrastructure

Andrés García-García[(a)], Carolina Perpiñá[(b)], Carlos de Alfonso[(a)], Vicente Hernández[(a)]

([a]) Instituto de Instrumentación para Imagen Molecular (I3M) - Universitat Politècnica de València, Camino de Vera, s/n, Edificio 8E, Escalera N, 1ª Planta, 46022 Valencia, Spain

([b]) Instituto de Ingeniería Energética, Universitat Politècnica de València, Camino de Vera, s/n, Edificio 8B, Escalera F, 2ª Planta, 46022 Valencia, Spain

Abstract

This paper presents the integration of desktop grid infrastructure with GIS technologies, by proposing a parallel resolution method in a generic distributed environment. A case study focused on a discrete facility location problem, in the biomass area, exemplifies the high amount of computing resources (CPU, memory, HDD) required to solve the spatial problem. A comprehensive analysis is undertaken in order to analyse the behaviour of the grid-enabled GIS system. This analysis, consisting of a set of the experiments on the case study, concludes that the desktop grid infrastructure is able to use a commercial GIS system to solve the spatial problem achieving high speedup and computational resource utilization. Particularly, the results of the experiments showed an increase in speedup of fourteen times using sixteen computers and a computational efficiency greater than 87% compared with the sequential procedure.

Keywords: biomass, Desktop Grid, Geographical Information System

1. Introduction

Spatial data processing covers several fields of research, such as geography, urban planning and natural resources management, among others. These spatial problems generally process a large volume of spatial data (raster or vector format) and many resources or computing time [1]. Geographical Information Systems (GIS) are commonly used in order to capture, storage, retrieval, analyse and display of spatial data, and also includes a set of tools, functions or algorithms that can be used over spatial datasets [2].

In recent years, grid computing technologies have been used to solve very large scale problems, where regular computers do not offer a competitive performance [3]. The grid computing approach to this type of problems is based on dividing the input data into independent subsets, solving smaller instances of the same problem in parallel, and combining the output data to obtain the final result of the original problem. Hu [4] studied whether it is feasible to use grid computing for resolving spatial problems, with positive conclusions. Therefore, due to the nature of spatial problems, grid computing emerges as an option to improve the performance of GIS.

The parallelization of spatial problems is not a new topic. Openshaw [5] proposes a parallel algorithm for the classification of spatial datasets for the Cray T3D supercomputer, motivated by the large quantity of spatial information generated by the new technologies at that time. Dowers [6] mentioned the difficulty of providing parallel computing in commercial GIS software, since a GIS software manages different data formats, models, algorithms, etc. The authors subsequently exposed that the adoption of standard interfaces [7] by the different GIS solutions unifies data format and available operations and enables the creation of a general model for a parallel GIS implementation.

The grid computing high throughput paradigm matches well with the nature of geospatial analysis problems. Shen [8] utilized grid computing technologies to provide a parallel version of an algorithm for image processing. Also, the usage of grid computing for GIS is discussed in [9] and [10]. Xiao [9] proposed an architecture that enables the usage of a mix of GIS operations and spatial analysis algorithms in a computer cluster for high performance processing. The system includes ad-hoc algorithms and certain GIS functions, and relies on the compliance of GIS software to the Open GIS specification. Huang [10] proposed a grid approach to spatial data management based purely in databases technologies and distributed processing of queries. More recently, Huang [11] presented a native parallel implementation of GRASS GIS software [12]. This development enables users to attain high performance capabilities for their operations, but is restricted to that specific GIS suite. Li [13] proposed the usage of complete GIS software suites in the by means of wrapper grid services.

Whereas most of available HPC (High Performance Computing) solutions for geospatial analysis are focused on specific algorithms, specific operations of GIS or rely on standardized interfaces to provide parallel processing only a few deal with the parallelization of an entire GIS suite. The purpose of this paper is to integrate a desktop grid infrastructure in a commercial GIS (ArcGis 9.3) in order to solve a discrete facility location problem by applying parallel processing techniques. This paper overcomes, on the one hand, the restrictions imposed by the GIS suite and, on the other hand, the shortage of computational resources caused by the resolution of large instances of vector spatial processes in a GIS.

The remainder of this article is organized as follows. The methodology framework used to implement the desktop grid infrastructure is described in Section 2. Results and discussion of the performance experiments are presented and analysed in Section 3, as a case study. Conclusions and future works are described in section 4. Appendix A completes the paper.

2. Methodology framework

2.1. An overview of the desktop grid infrastructure

Specifically desktop grid infrastructures are instances of grid infrastructures where the nodes composing the infrastructure consist of commodity computers which are federated using the master/slave model. According to this model, a problem composed by a set of independent data can be solved in a distributed environment by dividing the input in smaller sets, solving the problem for each subset, and composing the final solution. These three stages are known as preprocessing, processing and postprocessing, and are fully described next. A general schema of the conceptual model is depicted in Figure 1. More details of the main steps for the resolution of a problem in a desktop grid infrastructure, from both server and client side, are given in Appendix A (table 4 and 5).

One of the main differences of desktop grid infrastructures from traditional grid infrastructures is the assumption that the working nodes operate on disjoint networks, and hence communication between processes is not possible. This imposes a limitation on the type of problems that can be solved using desktop grid, specifically being limited to problems composed by independent calculations.

Nevertheless this family of problems include a wide variety of problems of interest to the scientific community [14]. In the case study we include an example of a geospatial problem that exhibits such feature.

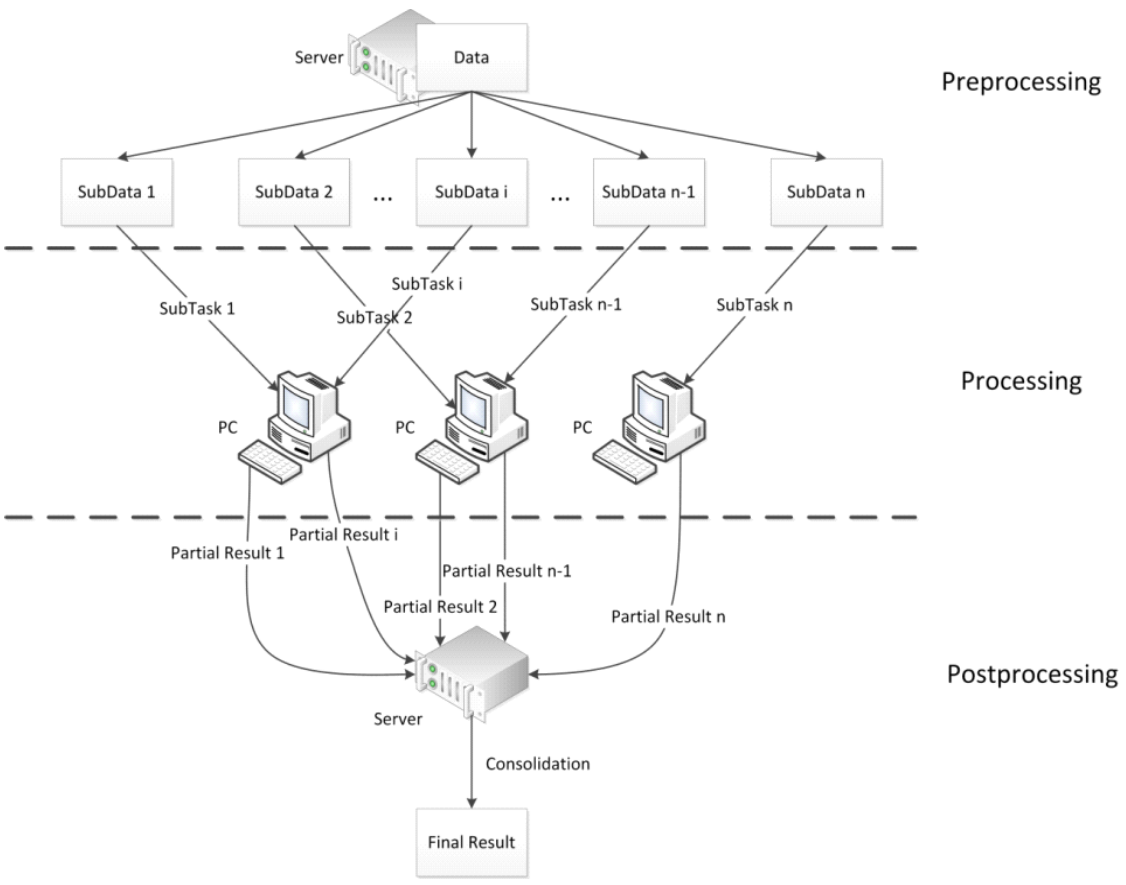


Figure 1: Conceptual flow with the three main stages for the execution of a task in a desktop grid

The mathematical model of a desktop grid execution utilized to study the expected behaviour of the system is the Grid Speedup as introduced by [15]. The execution time of a workload in a grid infrastructure with C computing elements and p processors is defined by the equation 1.

$$T_{C,p}(W) = \frac{W_p}{pC\Delta} + Q_2\left(\frac{W_p}{C}, p\right) + Q_1(W_p, C) \tag{1}$$

where W_p is the execution time of the workload, Q_2 express the overhead of intra-computing elements communications with p processors, and Q_1 express the overhead introduced by the inter-computing elements communications.

2.2 Material and infrastructure

BOINC (Berkeley Open Infrastructure for Network Computing) [16] is the desktop grid middleware chosen to solve the facility location problem proposed in this paper, as it provides every needed tool to coordinate the execution, transference of information, result gathering, etc.

The desktop grid infrastructure is composed by a set of desktop computers available in the working office. In particular, the experiments have been done using a maximum of sixteen computers with specs 2 (biprocessor) AMD Opteron™ processor with 2 GB of RAM. These desktop computers are Windows-based and have installed ArcGIS, version 9.3. The spatial data are mainly vector data stored in shapefile format. In addition, a webpage has been created in order to make available and include the scripts involved in the three steps of the desktop grid execution [17].

2.3. Implementing the desktop grid infrastructure in GIS environment

BOINC provides a framework for the implementation of desktop grid that automatically manages the three phases of the master/slave model. As a framework, BOINC completely implements the common steps of a desktop grid such as task distribution, result retrieval, node management, etc. The integration of a specific system with the generic framework is performed by providing a set of problem-specific scripts. These are called: 1) the partition script, 2) the solve script and 3) the merge script, and each one corresponds with the preprocessing, processing and postprocessing states of a desktop grid execution. BOINC executes the scripts to distribute, execute and retrieve the problem instances, as described next. In our particular case, our aim is to interface the desktop grid with the ArcGIS suite.

2.3.1 Preprocessing: preparing the spatial data

Prior to the execution of a task in the desktop grid infrastructure, a preprocessing step is necessary. This step generates the input data for each of the subtask to be processed by the clients. The partition script has been used to perform the division of the input dataset [17]. The script examines the dataset, and divides the data into disjoint subsets. Each subset is disjoint from one another in order to not overlap the work performed by the computational subtasks. This is possible since the calculations performed by the model are independent, and hence each task needs no information contained in other datasets. Once the data has been divided, the server schedules for execution of a subtask for each newly created dataset.

2.3.2 Processing: solving the spatial problem

In desktop grid infrastructure, the client nodes ask the server for work, receive the task to calculate and send back the results to the server when done. In this case, we need to interface BOINC client with the ArcGIS suite in order to solve the subtasks generated by the preprocessing step.

ArcGIS provide tools for the automatic translation of geospatial models (using model builder tool) to executable Python scripts. These scripts interact with the ArcGIS programming API to execute models, and enable importing and exporting data. These are the solve script [17], which make possible to import the subtask data to ArcGIS, resolve the subtask and export the output data back to a format that can be retrieved by BOINC.

The solve script introduces an abstraction layer between BOINC and the underlying GIS system. From the desktop grid point of view, the system dispatches subtasks to the client and retrieve back the output data. On this step, it is possible to interchange the solve script produce by ArcGIS with solve script designed for any other GIS system transparently to the rest of the infrastructure.

Particularly, clients download for each subtask the solve script to execute and the subset of data needed for the execution. The script usually is shared between many substasks, and hence only need to be downloaded once. Clients then solve each subtask, and generate a partial result for the problem that is sent back to the server.

2.3.3 Postprocessing: obtaining the problem results

In order to compose the final result, a postprocessing step is needed. Once all subtasks generated for an instance have been completed, these partial results need to be composed together to generate an output that is equivalent to execute the original problem in a single machine. Since, just like the data partition, this step is dependent of the problem, BOINC relies in a merge script [17] to perform this job. In the particular case of ArcGIS, we rely on models specific for each geospatial problem to perform this labour. The equivalence of the output of the execution of the same problem using both the desktop grid and a single machine is a proof that the desktop grid solves the same problem and provides the same outcome.

3. Case study: a spatial location problem

Desktop grid infrastructure has been implemented using a facility location problem as a test subject. The main aim is focused on overcoming computational resources shortage resulting from the large number of spatial processes through parallelization. These spatial processes involve both the biomass evaluation and the biomass logistic optimization in GIS environment. In this section the scenario of the spatial location problem is explained, followed by the results of the performance experiments.

3.1 Spatial location problem scenario

The location of a biomass plant have been carried out by applying a GIS-based environment to a set of provinces of Spain in order to identify, calculate and map the most suitable locations of biomass plants per district. Figure 2 shows the most representative data that are part of the employed logistics strategy.

The size of the districts belonging to each province varies greatly, from 430 km^2 for the smaller to 4,579 km^2 for the largest. The resolution of the biomass plant location problem in a GIS environment need the consideration of a large quantity of spatial data as is shown in table 1. The purpose is calculating and evaluating the possible combinations between all biomass collecting points (origins) and the potential sites for sitting a biomass plant (destinations). In addition, several factors, both natural and artificial, must be considered when planning a site for such an energy installation. These factors could be identified as restrictive areas in which it is not allowed to place this facility [18, 19].

Table 1. Cartographic data used in the study

Layer name	Feature	Cartographic data from
Agricultural areas	Polygon	Spanish cartography of crops and land use [20]
Forest areas	Polygon	3rd National Forestry Inventory (2008) [21]
Boundaries	Polygon	BCN25: Numerical Cartographic database [22]
Urban areas	Polygon	BCN25: Numerical Cartographic database [22]
Protected areas	Polygon	Nature2000 network [22]
Orography	Raster	National aerial ortophotogrametric Plan [22]
Hydrology	Line	BCN25: Numerical Cartographic database [22]
Biomass collecting points	point	Origins 1: resulting from the centroid of each 1km square of the grid covering the agricultural and forestry biomass amount over the study area.
Biomass plant candidate points	point	Destinations 1: resulting from apply several factors, both natural and artificial (constrains). These points are also Origins 2.
Biomass distribution (consumers)	point	Destinations 2: resulting from the centroids of each municipality where biomass is transported as pellets.
Transport network	Line (network)	Geograma S.L. (VAR de Tele Atlas) [23]

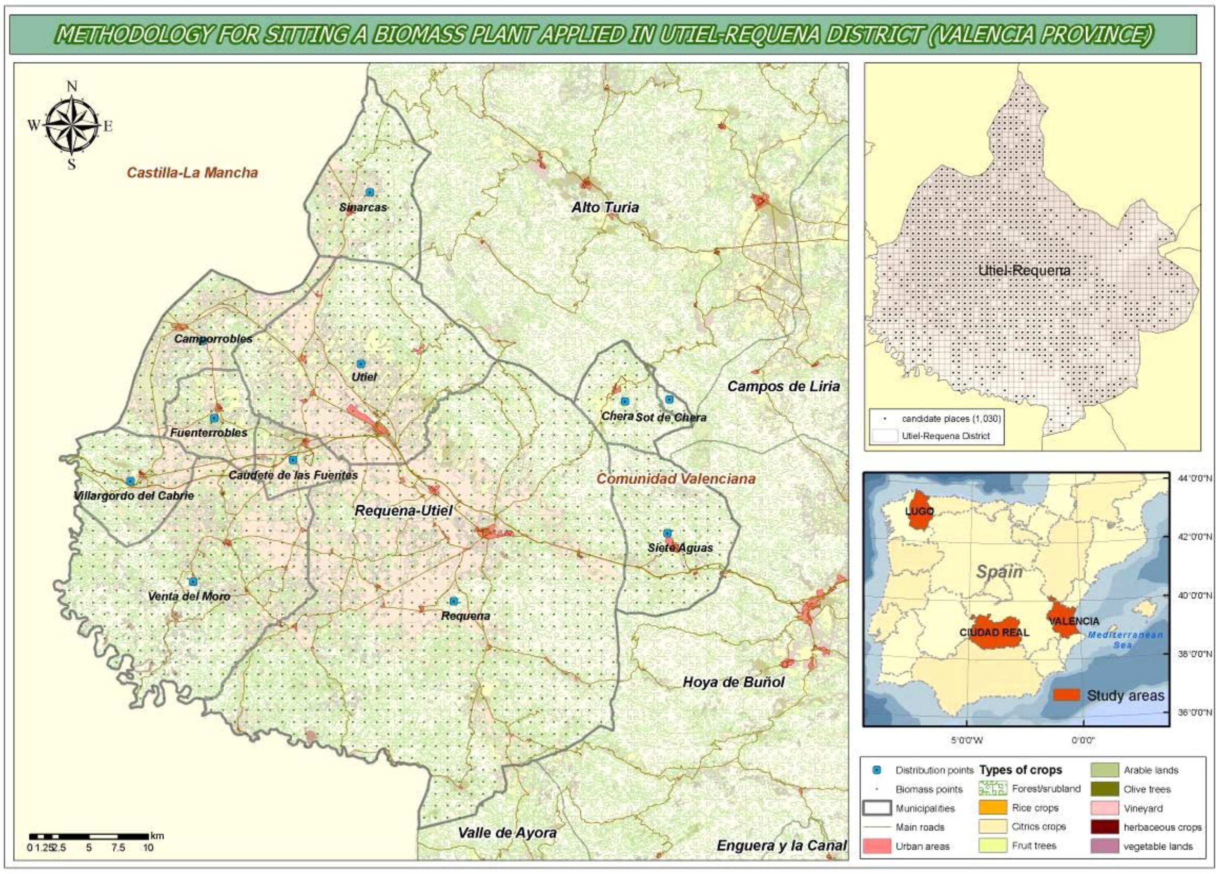


Figure 2. Main cartographic data used in the methodology applied in Utiel-Requena district

The methodology was structured in two main stages: first, the identification and quantification of available biomass from agricultural and forestry resources, and second, the analysis of biomass logistic and optimization to locate biomass plants. Specifically, this paper is focused on the second stage where a desktop grid infrastructure has been deployed in order to be applied to the logistic and transport strategies by means of an Origen-Destination matrix. O-D matrix is not an optimization model in itself but a strategy for determining the cost of all candidate sites. This allows knowing the range of costs whiting the study area, and to be able to identify the areas with the lower, intermediate and the higher costs.

The main idea is to locate the biomass plant by evaluating, on the one hand, from the collecting biomass phase, all the possible combinations between the candidate sites for sitting a biomass plant and all the biomass collecting points. On the other hand, from the distribution phase, all the possible combinations between the candidates place for sitting a biomass plant and the municipality centroids as potential consumers (Figure 2). For both phases the total biomass cost is calculated and the minimum cost is obtained. In order to transport the biomass from the origins to the destinations the mathematical formulation to minimize the weighted distance is shown in equation 2.

$$Min\ Z = \sum_{j=1}^{n} d_{ij} w_i \qquad (2)$$

where i = 1,…, m is the location of demand, j = 1, ..., n are the candidates sites, dij is the shortest distance between the location of demand i and candidate sites j, and w_i is the weight of demand point i (number of trips).

Further information related to the methodology herein presented and more details about the applied network analysis is fully described in Perpiñá and Perpiñá et al. [18, 19].

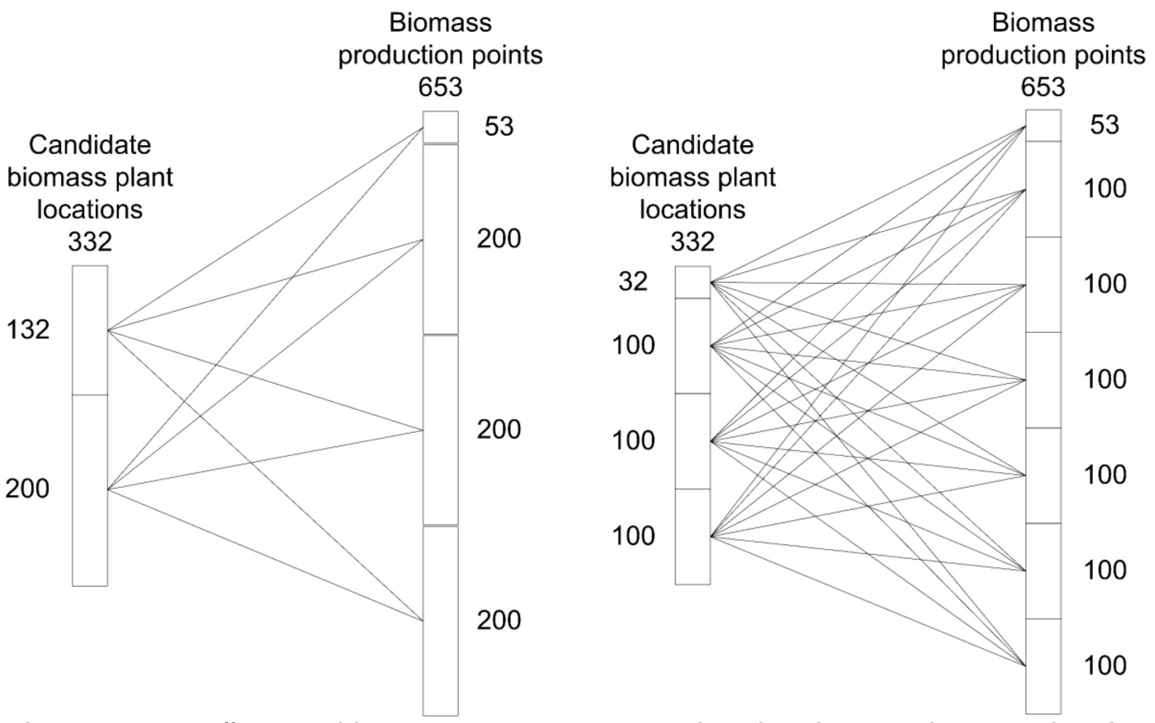
Taking into account the second stage, the execution of the equation 2 using spatial data to solve the network analysis becomes a computing intensive task, and takes not only a long computational time, but also a great amount of memory provoking the failure of the instances execution. In table 2 is represented an example of several district showing the number of candidates, biomass collecting points, number of combinations, subtask by the partition, and the needed memory to solve each problem. The table shows the rate of grow of the memory when the size of the input increases, and even for middle-size instances the required memory exceeds the one available in current commodity computers. Hence, the memory imposes the primary limitation on the problem execution.

Table 2. Information for sample input data

District	Candidates places	Biomass points	Area (km^2)	Combinations	Subtasks	Memory (GB)
La Safor	117	383	430	44,811	16	3.015
L`Horta	191	518	594.36	98,938	33	6.742
Fonsagrada	114	1,771	1,585.91	201,894	72	12.464
A Mariña	273	1,476	1,465.17	397,488	150	46.793
Montes Sur	840	1,197	1,308.9	1,005,480	384	15.488
Requena-Utiel	1,030	1,760	1,126	1,812,800	720	159.111
Central	1,536	2,544	2,572.3	3,907,584	1,530	364.716
Sierra Morena	2,275	3,688	2,322	10,802,152	3,330	347.601
La Mancha	4,196	4,756	4,579	19,956,176	7,968	1,142.807

3.2. Performance experiments: Results and discussion

In order to measure the performance of the parallel execution of the location problem using a desktop grid infrastructure, different experiments has been designed. These experiments consist of solving a set of problems instances using different number of nodes. The parallel execution time for each experiment configuration has been calculated according to equation 1, in order to obtain metrics about the speedup and throughput of the parallel approach compared to the sequential execution. The sequential execution time has been calculated as the aggregation of the execution time of each individual subtask in a single computer.


Candidate biomass plant locations 332
132
200
Biomass production points 653
53
200
200
200
Candidate biomass plant locations 332
32
100
100
100
Biomass production points 653
53
100
100
100
100
100
100
Figure 3: Smaller partitions generate more subtasks, but each one size is also smaller

The input datasets are composed by the biomass collecting points (origins) and the candidate plant location points (destination). A subset of the input data is then a set of *n* origins and *m* destinations, producing an n-to-m partition of the data, as illustrated in Figure 3. Therefore, in order to provide the best efficiency possible, it is necessary to provide values to n and m such that maximize the size of the subtasks without introducing memory overflow issues. According to empirical tests the optimal values for n and m have been estimated at 50-50.

The first analysis represents time execution obtained for different number of cores. In the parallel approach, the parallel execution time has been calculated as the time elapsed between the creation of the first subtasks and the retrieval of the last result. Such way of calculating the parallel time intentionally includes any overhead introduced by the BOINC infrastructure. However, the postprocessing time has been left out from this calculation, since this step is common to both the sequential and parallel versions. In Figure 4 it can be seen that the BOINC approach using just one core yields slightly higher execution times than the sequential procedure. This behaviour is typical of parallel systems and it is due to the overhead introduced by the desktop grid infrastructure, which implies: generating subtask, distributing the input data among clients and retrieving the results. When more cores are added to the infrastructure, the execution time drops proportionally as expected.

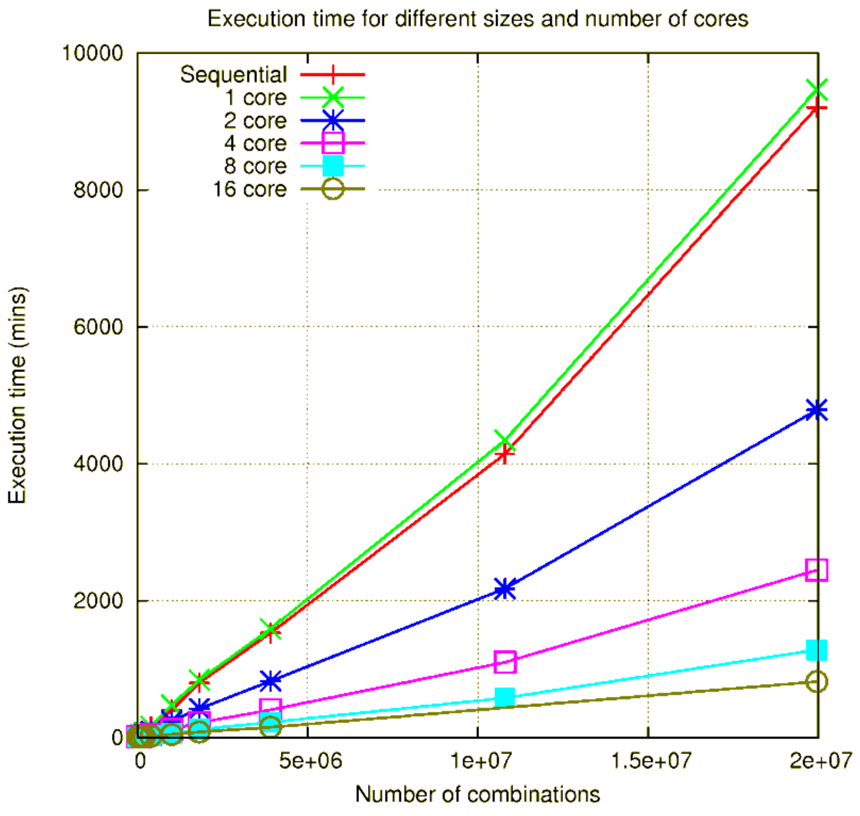


Figure 4: Execution time for different instance size and number of cores

Such behaviour can be better analysed in Figure 5, which shows speedup values for each test (the horizontal axis represents the size of the problem in a logarithmic scale, to enhance the readability of the results). The speedup values grow along with the size of the instance, achieving a stable value for the mid-size problems. Speedup growth is almost linear with the number of cores, until the higher values, where performance gets standstill due to the overheads introduced by the parallel approach. Such trend is obvious because as the size of the problem grows, more tasks are introduced and therefore the parallel overhead is increased.

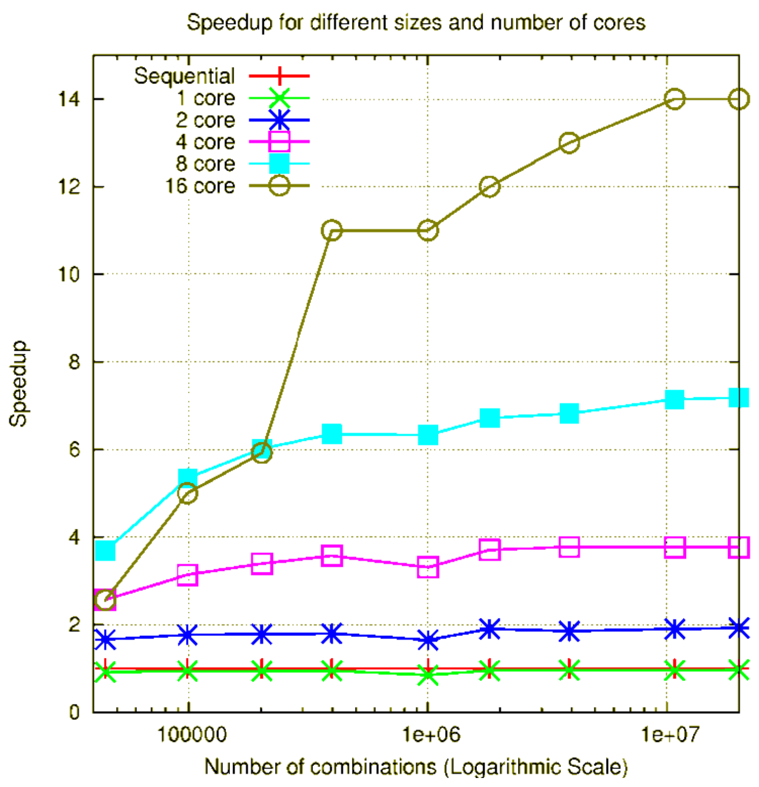


Figure 5: Normalized Speedup for different instance size and number of cores

Figure 6 illustrates the throughput of the experiments. The throughput is represented in subtasks completed per unit of time (hour). This measure enables us to quantify “how much work is done” for each desktop grid configuration. Table 3 shows other parameters that define the quality of the infrastructure, such as the failure rate and the overhead. The failure rate accounts for the effectiveness of the solution, and low values indicate that, despite the usage of commodity computers, the infrastructure is reliable. The overhead accounts for the efficiency of the solution, and low values indicate that the infrastructure is highly efficient, and therefore the resources utilization is maximized. According to this parameter, we could keep including new nodes in the system and expect a proportional increase in the performance of the solution.

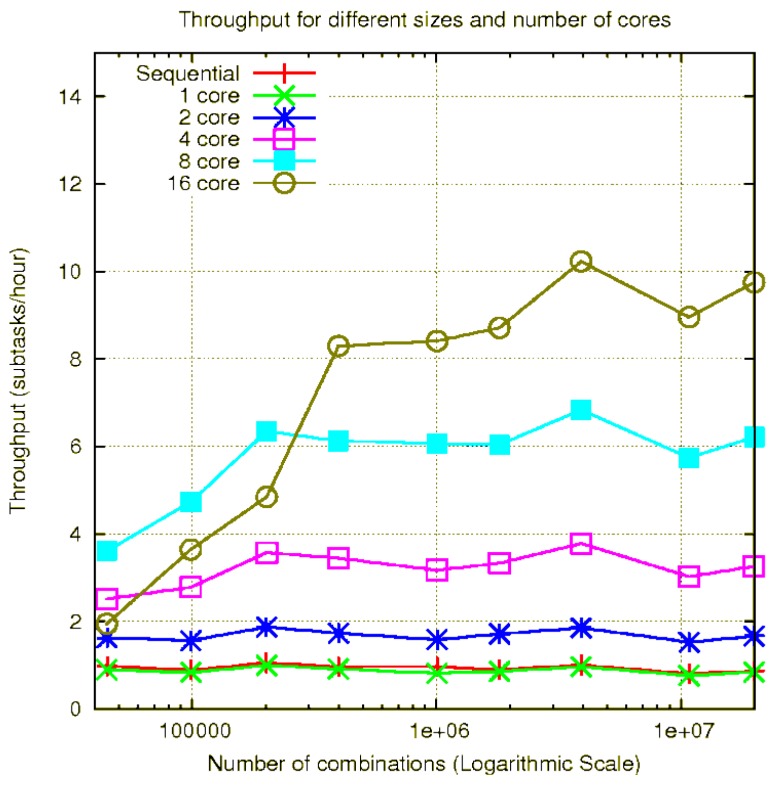


Figure 6: Normalized throughput for different instance size and number of cores.

Table 3: Failure rate and overhead for different infrastructure configurations

	1 core	2 cores	4 cores	8 cores	16 cores
Failure rate	0.56%	0.14%	0.22%	0.11%	0.27%
Overhead	2.8%	13.3%	10.9%	11.4%	10.4%

The obtained results are in concordance with the mathematical model introduced by the equation 1. Equation 1 predicts high speedup when the workload presents little communications, and high efficiency when the workload presents little overhead.

Firstly, the spatial location problem adapts well to the desktop grid configuration and provides positive values for speedup, throughput and efficiency. This fact is due to a large ratio between the work unit size and the computing element overhead, and the little communications involved in the process. Secondly, the computing element overhead is independent of the number of computing units involved. This fact is due to the absence of communication between subtasks. Thirdly, it has been shown that the location problem scales well along with the number of nodes. This fact is due to the independence of the calculations. Good scalability means that the speedup improves the performance in an almost linear progression respect to the number of nodes, which is the same than achieving high values of efficiency for all executions. Good scalability also implies that growing the size of the infrastructure would further increase the performance in the resolution of problems. Comparing experimental results, [24] provides the execution time of a problem instance in GRASS GIS software using grid for different number of nodes. The main conclusion was that the speedup increases almost linearly for different number of nodes. This behaviour is a sign of good scalability as happens in our case study as well.

4. Conclusions and future works

The computing intensive nature of a GIS-based facility location problem introduces a limitation on the size of the problem instances that can be solved. Parallel and distributed techniques for the spatial data processing are used in order to overcome these limitations. Particularly, in this study a desktop grid infrastructure has been utilized to perform distributed execution of spatial analysis, mainly vector data in shapefile format, using commodity computers.

This approximation provides a generic framework for the execution of legacy spatial-specific models, by using a partition script, solve script, merge script that interface the platform with the underlying GIS suite. The main advantage is the ability to solve memory bound problems, reducing the execution time of very large instances or providing reliability in the execution of very long computations.

A case study made over real instances manifests how desktop grid enables the resolution of large problems, producing a significant speedup in the problem execution, reducing the time to a fourteenth part of the sequential execution time using 16 computers. Also, the high values obtained for the efficiency, higher than 87% for all the executions, state that the application scales well, and therefore increasing the size of the infrastructure would further improve the performance.

The adaptation of GIS tools to desktop grid infrastructure enables the resolution of problems where the complexity is defined by the size and independency of the input. Using the approach proposed in this paper, the infrastructure built to support this problem can be reutilized to solve other spatial problems that exhibit independent calculations. This ability opens working lines in adapting new GIS problems to desktop grid and enables further developments in this field, for instance, raster data.

Future lines of work include the adaptation of the methodology to more recent computing paradigms. The advent of Cloud Computing as a successor of Grid Computing enables researchers to solve traditional Grid-enabled problems in new, more innovative ways. The biomass plant location problem could be adapted from the traditional master-slave computing model to a cloud-enabled map-reduce paradigm, taking advantage of the cloud elastic provisioning and massive parallelism to provide even greater speedups.

Appendix A. Main steps for the resolution of a problem in a desktop grid infrastructure.

Table 4: Steps for the resolution of a problem in a desktop Grid infrastructure (server side)

Step	Description
Step 1: Data partition	When a new job is introduced in the server, the first step is to generate a partition of the input data of the problem.
Step 2: Subtask generation	Using the sets of data generated in Step 1, a number of subtasks is created. Each subtask consists on a smaller instance of the original problem.
Step 3: Subtask scheduling	Once the subtasks are created and stored in the system, they are scheduled for execution. A scheduled subtask is sent to any client which asks for work.
Step 4: Wait for results	The server waits in an idle state until the client nodes report completed subtasks. A completed subtask can be either successful or failed. Failed subtasks are rescheduled for execution (Step 3), whereas successful executed subtasks are stored for further processing (Step 5).
Step 5: Partial result consolidation	Once the partial results from all the subtasks have been retrieved, they are processed to produce the outcome of the original problem.

Table 5: Steps for the resolution of a problem in a desktop grid infrastructure (client side)

Step	Description
Step 1: Waiting period	When a client node is idle, it waits a fixed period of time before request the server for work.
Step 2: Request for work	Idle clients ask the server for work periodically. If the server has any subtask scheduled for execution, the client moves to Step 3, otherwise, it goes back to Step 1.
Step 3: Download subtask	If the request for work is replied positively, the client proceeds to download the scheduled subtask from the server. This download consists on the retrieval of the subtask input files.
Step 4: Execute subtask	The client proceeds to the subtask execution.
Step 5: Subtask outcome report	When an execution is finished, the client reports back the outcome to the server. The outcome can either be a successful execution or a failed execution. If the execution is successful, the output files are uploaded to the server. After any execution, the client proceeds to Step 2.

Acknowledgements

This work has been developed under the support of the program Formación de Personal Investigador, grants number BFPI/2009/103 and BES-2007-17019, from the Conselleria d’Educació of the Generalitat Valenciana and the Spanish Ministry of Science and Technology.

References

[1] R.L. Church (2002) Geographical information systems and location science. Computers and Operational Research 29: 541- 562.

[2] K.C. Clarke (1986) Advances in Geographic Information Systems, Computers. Environment and Urban Systems 10:175 - 184.

[3] EELA Consortium (2012) E-science grid facility for Europe and Latin America. Lightweight Middleware for Grid Computing

[4] Y. Hu, et al. (2004) Feasibility Study of Geo-spatial Analysis Using Grid Computing. International Conference on Computational Science, pp. 956–963.

[5] S. Openshaw, I. Turton (1996) A parallel Kohonen algorithm for the classification of large spatial datasets. Computers & Geosciences 22:1019 - 1026.

[6] S. Dowers, B.M. Gittings, M.J. Mineter (2000) Towards a framework for high-performance geocomputation: handling vector-topology within a distributed service environment. Computers Environment and Urban Systems 24:471- 486.

[7] Open Geospatial Consortium, Inc. (2012) Open GIS Specification Model

[8] Z. Shen, et al. (2007) Distributed computing model for processing remotely sensed images based on grid computing. Information Sciences 177: 504 - 518

[9] N. Xiao, W. Fu (2003) SDPG: Spatial data processing grid. Journal of Computer Science and Technology 18: 523 - 530

[10] Z. Huang, et al. (2009) Geobarn: a practical grid geospatial database system. Advances in Electrical and Computer Engineering 9:7 - 11

[11] Huang, F. et al. (2011) Explorations of the implementation of a parallel IDW interpolation algorithm in a Linux cluster-based parallel GIS. Computers & Geosciences 37: 426-434.

[12] GRASS Development Team (2012). GRASS GIS.

[13] W.J. Li, et al. (2005) The Design and Implementation of GIS Grid Services, H. Zhuge and G. Fox, eds, Grid and Cooperative Computing. Vol. 3795 of Lecture Notes in Computer Science 10, Springer Berlin / Heidelberg, pp. 220 -225

[14] University of California. List of BOINC projects. http://boinc.berkeley.edu/projects.php

[15] A.G. Hoekstra and P.M.A. Sloot (2005) Introducing Grid Speedup: A Scalability Metric for Parallel Applications on the Grid, EGC 2005, LNCS 3470, pp. 245- 254

[16] University of California (2012). Grid computing with BOINC

[17] Available scripts webpage: http://personales.upv.es/angarg12/

[18] C. Perpiñá, D. Alfonso, A. Pérez-Navarro (2007) BIODER project: biomass distributed energy resources assessment and logistic strategies for sitting biomass

plants in the Valencia province (Spain), 17th European Biomass Conference and Exhibition Proceedings, Hamburg, Germany, pp. 387 - 393

[19] C. Perpiñá, et al. (2008) Methodology based on Geographic Information Systems for biomass logistics and transport optimization. Renewable Energy 34: 555 - 565

[20] Spanish Ministry of Agriculture, fisheries and food (2002) Agricultural and land use cartography.

[21] Spanish Ministry of Environment (2008) 3rd National Forestry Inventory

[22] National Geographic Institute (2003) BCN25: Numerical Cartographic Database (Spain)

[23] Geograma S.L. (2012) Teleatlas

[24] J. Marco, I. Campos, I. Cotterillo, A. Monteoliva, C. Oldani (2008) Modelling of a Watershed: A Distributed Parallel Application in a Grid Framework. Computing and Informatics 285-296