Research Article **Open Access**

Mariela Curiel*, David F. Calle, Alfredo S. Santamaría, David F. Suarez, and Leonardo Flórez

Parallel Processing of Images in Mobile Devices using BOINC

https://doi.org/10.1515/eng-2018-0012
Received October 1, 2017; accepted November 28, 2017

Abstract: Medical image processing helps health professionals make decisions for the diagnosis and treatment of patients. Since some algorithms for processing images require substantial amounts of resources, one could take advantage of distributed or parallel computing. A mobile grid can be an adequate computing infrastructure for this problem. A mobile grid is a grid that includes mobile devices as resource providers. In a previous step of this research, we selected BOINC as the infrastructure to build our mobile grid. However, parallel processing of images in mobile devices poses at least two important challenges: the execution of standard libraries for processing images and obtaining adequate performance when compared to desktop computers grids. By the time we started our research, the use of BOINC in mobile devices also involved two issues: a) the execution of programs in mobile devices required to modify the code to insert calls to the BOINC API, and b) the division of the image among the mobile devices as well as its merging required additional code in some BOINC components. This article presents answers to these four challenges.

Keywords: grids, mobile grids, mobile devices, image processing, Android, ITK, parallel processing

1 Introduction

The use of mobile devices has increased significantly around the world in the recent years. According to Digital Trends (https://www.digitaltrends.com/mobile/smartphone-users-number-6-1-billion-by-2020/), the number of smartphone users in the world is expected to reach 6.1 billion by 2020. Also, the current capabilities of these devices have increased considerably. With the recent advances in low powered processors, mobile devices can perform computationally intensive operations enabling these devices to be considered as computing platforms [1–3].

Taking advantage of the previously mentioned smartphones capabilities, since the last decade, many research projects have addressed the problem of incorporating mobile devices to the grid. Grid Computing involves the aggregation of geographically-disperse and heterogeneous resources from different organizations to solve computationally complex problems [4]. The inclusion of mobile devices to the grid infrastructure brought new categories of grids [5]. One of these new categories is the **Mobile Grid**, which was defined by Furthmüller and Waldhorst [6] as a grid that includes at least one mobile device. Users can connect their mobile devices to the grid (*smartphones*, *tablets*, etc.) with basically two purposes: 1) to obtain access to grid resources and/or 2) to place their mobile devices at the disposal of grid users (i.e. mobile devices are resource providers). Our research focuses on the use of mobile devices as providers of computing resources [6–10]. The idea is to take advantage of idle computing cycles of mobile and fixed devices to process in parallel any kind of images, in particular medical images. We are currently testing the feasibility of this technology, which would allow, in the long term, health professionals to send diagnostic images and receive results making few clicks on his/her mobile devices, while examining his/her patients. With mobile devices, it is also possible to reduce costs and space in computing resources for medical institutions as well as to reach places and communities with low penetration by other types of computer systems.

The chosen platform used to build our proposal was BOINC [11]. BOINC is an open grid framework that has been used mainly for voluntary grid computing projects. BOINC volunteers dedicate unused computing cycles on their personal computers, laptops, mobile devices, etc., to parallel scientific research computations.

***Corresponding Author: Mariela Curiel:** Depto. de Ingeniería de Sistemas, Pontificia Universidad Javeriana, Bogotá, Colombia. Código postal: 11001000. E-mail: mcuriel@javeriana.edu.co
David F. Calle, Alfredo S. Santamaría, David F. Suarez, Leonardo Flórez: Depto. de Ingeniería de Sistemas, Pontificia Universidad Javeriana, Bogotá, Colombia. Código postal: 11001000. E-mail: {dcalle,asantamaria,dsuarez,lflorez}@javeriana.edu.co

The parallel processing of images in mobile grids involves at least four technical challenges: 1) Transparency for the programmer: the execution of programs in the mobile grid should not require changes in the source code; 2) Cross-compilation of the libraries for image processing (i.e. ITK [12]); 3) Distribution and collection of data; 4) Achieve acceptable performance.

In order to look for answers to these challenges, we pose the following problem: Apply the *Smoothing Recursive Gaussian Image Filter* [13] to an image. This filter computes the smoothing of an image by convolution with the Gaussian kernels. The idea is to divide the image in regions among several mobile devices to apply the filter in parallel. Then the individual results are merged in a final integrated result.

We start the article by providing some definitions in Section 2. In Section 3 the problem to be solved is presented. We briefly explain the main characteristics of BOINC in Section 4. In Section 5 we describe the modifications to BOINC components and the steps to generate the executable file to process the image. Once the infrastructure was ready, we conducted some preliminary tests to evaluate the performance of our solution; thus, experimental design is explained in Section 6. We present some related work in Section 7 and conclude in Section 8.

2 Background

A **Grid** is a collection of heterogeneous distributed computing resources for solving large-scale computational and data intensive problems ([4, 14]).

The rise of wireless technology and mobile devices has increased the number and types of resources to be integrated to the grid. As a result, the concept of grid has been enriched and new categories have emerged [5]. One of these new categories is the **Accessible Grid**, whose resources are available regardless of their physical capabilities and geographical locations. Accessible grids consist of a group of mobile or fixed devices with wired or wireless connectivity and predefined or ad hoc infrastructure [15]. Wireless, mobile, and ad hoc grids belong to this category. In **Wireless Grids**, wireless devices can be either resource providers, which contribute to data processing and/or storage, or only be mere consumers of grid services [6]. The concept of **Mobile Grids** is very similar to wireless grids concept; in fact, many authors used both terms indistinguishably. Furthmüller and Waldhorst [6] define a mobile grid as a grid that includes at least a mobile device. Finally, some researchers strictly define **Ad Hoc Grids** as grid environments without fixed infrastructures, i.e., all their components are mobile [16].

Volunteer computing is a type of distributed computing in which people, so-called volunteers, provide computing resources to projects, which use the resources to do distributed computing and/or storage [17]. Volunteers are usually members of the general public in the possession of their own personal computing resources (desktops, laptops, smartphones, etc.) with an Internet connection. Organizations can also act as volunteers and provide their computing resources. The Berkeley Open Infrastructure for Network Computing (BOINC) is an open grid framework that has been used mainly for voluntary grid computing projects [11]. Examples of volunteer computing with BOINC are SETI@home, ClimatePrediction.net, Rosetta@home and Einstein@home.

Image processing is a method to perform operations over images using mathematical operations, in order to change the image itself or to extract useful information from it. The output can be either an image or a set of characteristics or parameters related to the image [18].

Filters are algorithms that receive images as inputs and perform operations over these. The output is another image that highlights or de-emphasize certain characteristics of the initial image. For example, filters can reduce noise and highlight or soften curves in an image. In the spatial domain, most filters operate over each pixel independently, modifying its value given an operator matrix or kernel and the neighboring pixel's values [19]. On the other hand, the frequency domain operates with the rate at which the the pixels change over the spatial domain.

Gaussian Filters are linear filters that operate in the special domain, modifying the input with a Gaussian convolution, namely, the weights in the operator matrix are given by the normal variance γ^2. These filters have the property of being separable, i.e. each component can be calculated independently [19, 20].

3 The Problem

Our research question is: Is it possible to do parallel processing of images using mobile devices?

The first step of our research was to choose the execution platform. We decided to use an existing grid instead of making our own implementation with MPI on Android devices (as in the works described in [21–23]) because grid technology offers extra values, mainly in regard to resource discovery, scheduling and quality of service.

In a previous paper [24], we compared several systems that implement mobile grids according to the literature. We analyzed BOINC [11], Ibis [25], MoGrid [26] and OurGrid [27]. Finally, we selected BOINC for the following reasons:

- BOINC enables the inclusion of mobile devices among its resources. The code for execution in mobile devices is available.
- It is possible to run tasks (workunits in BOINC) written in various programming languages. This is an advantage because the processing of medical images imposes certain constraints, such as the use of the ITK library [12], whose algorithms are programmed in C++. However, at the beginning of our research the execution of C++ code on Android required the introduction of additional code in the application.
- The system has good documentation and it is an active project with many resources to solve possible difficulties in its deployment and use.

As the goal is the parallel processing of medical images using mobile devices, we chose a filter algorithm. Some filtering algorithms are adequate for parallel processing because of their simplicity and because they can process different regions of one image independently or by interchanging few data with their neighbors, i.e. they exhibit coarse or medium grained parallelism. For the previously mentioned reason, filters that require information of the whole image are not suitable for our purposes.

The parallel model according to Flynn's taxonomy is the single-instruction-multiple-data model (SIMD), where multiple nodes do the same operation repeatedly over a large data set. In this case, each node processes a part of the image. The selected filter should have another important characteristic: the amount of computation should be high enough to justify the division of the work among multiple workunits. Initially we thought about using binarization or umbralization filters as they are very simple algorithms: for example, binarization algorithm generates a binary image based on a given threshold, replacing the values of each pixel, i.e. black or white. This characteristic makes this filter suitable for parallel processing since each pixel is independently processed. However this simplicity implies short execution times, hence parallel processing is not needed.

Finally, we decided to use a filter that only needs the information of some neighboring pixels (see Figure 1): The *Smoothing Recursive Gaussian Image Filter* of ITK [12]. This filter generates a smooth image by performing a convolution on it, which uses a kernel with values given by the Gaussian distribution. The result of this algorithm can be observed in Figure 2 on a magnetic resonance of a brain.

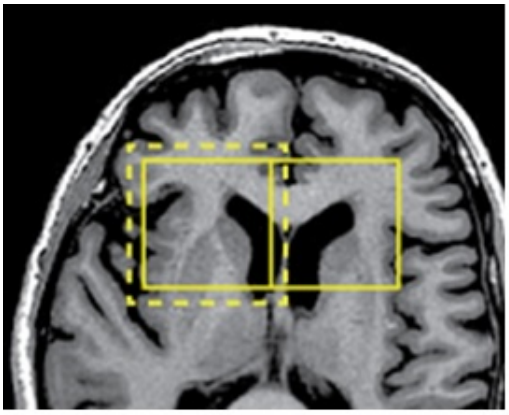

Figure 1: Overlap of regions in the processed image

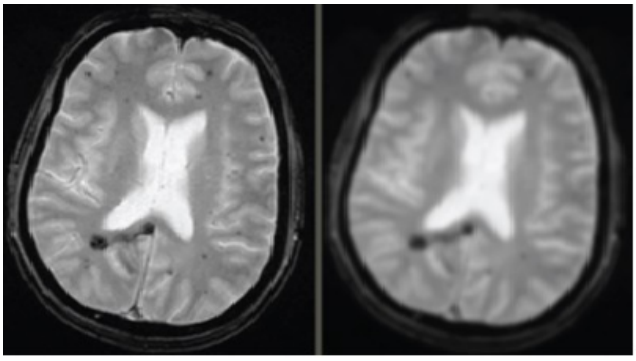

Figure 2: Application of the Smoothing Recursive Gaussian Image Filter. Left: Original image. Right: Blurred image after applying the filter.

4 BOINC

BOINC (**B**erkeley **O**pen **I**nfrastructure for **N**etwork **C**omputing) is an open grid framework that has been used mainly for voluntary grid computing projects. BOINC volunteers dedicate unused computing cycles on their personal computers, laptops, mobile devices, etc., to parallel scientific research computations [11]. The combination of both concepts: grids and mobile devices allows to define BOINC as a mobile grid. The main BOINC concepts are on the web page (https://boinc.berkeley.edu/trac/wiki/BasicConcepts).

The **BOINC project** corresponds to an organization or research group that must do volunteer or distributed computing. The project is identified by a master URL, which is the home page of its web site. Participants register with projects. A project can involve one or more applications. An **Application** includes several executable codes for different platforms (Windows, Linux/x86, Mac OS/X, etc.), a set of **workunits** and results. A **Platform** is a compila-

tion target that typically includes a CPU architecture and an operating system. The **Workunits** represent the inputs or parameters to a computation such as CPU time, memory and storage requirements and a soft deadline for completion. Each computation has results that consists of a reference to a workunit and a list of references to output files. Files (associated with application versions, workunits, or results) have unique names by projects and are immutable.

The Server of a BOINC project manages a relational database that stores descriptions of applications, platforms, versions, workunits, results, accounts, teams, etc. Participants who want to be part of a BOINC project must visit the project's website to download the **BOINC Client**.

The life-cycle of a BOINC workunit is:

1. A **Work Generator** (supplied by the user) creates the workunit and its input files.
2. BOINC creates one or more instances of the workunits and dispatches them to different hosts.
3. A client program in the host downloads the input files and executes the workunit.
4. Once completed the workunit, the host reports its culmination to the server.
5. A **Validator** (supplied by the user) checks the output files, perhaps comparing replicas. BOINC provides support for **redundant computing** by means of replicas. Two reasons justify the use of replicas:
   - **Lack of Battery**: Considering that a mobile device can run out of battery or can be suddenly disconnected, BOINC can preventively send the execution of the same workunit to different volunteers (replicas).
   - **Malfunctioning computers or malicious participants**: Public-resource computing projects must deal with erroneous computational results due to these two problems. In this case redundant computing is used for identifying and rejecting erroneous results. A project can specify that N results should be created for each workunit. Once M $\leq$ N of these have been distributed and completed, an application-specific function is called to compare the results and possibly select a **canonical result**. When no consensus is found, or if results fail, BOINC creates new results for the workunit, and continues this process until either a maximum result count or a timeout limit is reached. BOINC also offers the option of not creating replicas. In that case the validator (*SampleTrivialValidator*) only marks the result of the workunit as valid.
6. When a valid instance is found, an **Assimilator** program, supplied by the user, handles the results (e.g. by inserting them in a separate database).
7. When all instances have been completed, the **File Deleter** removes the input and output files.

5 Solution of the problem in BOINC

The solution of the given problem in BOINC involved four challenges. Three of them were related to technical issues and they are described in this section. The performance-related challenge is explored in the next section. The three technical issues were: 1) Transparency for the programmer: the execution of programs in the mobile grid should not require changes in the source code. 2) Cross-compilation of the libraries for image processing (i.e. ITK [12]). 3) Distribution and collection of data for parallel processing. The following subsections address each of these problems.

5.1 Generating the Wrapper for Android Mobile Devices

BOINC provides a C++ API that contains the methods that must be executed by a workunit to start and finish its execution, communicate progress, access input and output files and report errors, among others. For a workunit to be executed on the BOINC platform, you must modify the programs to include calls to this API or use available wrappers for the executing platform. The use of wrappers is an approach to "gridify" applications (i.e. achieve easy pluggability of ordinary applications into the Grid) [28] without modifying a single line of code. According to the taxonomy established by Mateo et al. in [28], the use of wrappers is a coarse-grain gridification approach where applications are taken in their binary form, along with some configuration files provided for the users (e.g., input and output parameters and resource requirements). The solution consist of wrapping the executable code with a software entity that isolates the complex details of the underlying Grid. With the use of wrappers the user does not need to modify the code to execute it in the Grid; usually modification requieres of some knoledge on the Grid system. Besides, applications can be plugged into the Grid even when the source code is not available. The article deepen into the description of this and other strategies to gridification.

Boinc wrappers run the applications as subprocesses, and handles all communications between

the BOINC client and the server. At the beginning of this work, a compiled wrapper to execute workunits on mobile devices with Android operating system was not available in BOINC distribution. The existent pre-compiled versions were: Windows_intelx86, Windows_x86_64, i686-pc-linux-gnu, x86_64-pc-linux-gnu, i686-apple-dxarwin and x86_64-apple-darwin (http://boinc.berkeley.edu/trac/wiki/WrapperApp).
Therefore, this was the first step of our work: to compile the wrapper for Android. The following paragraphs explain the different steps to cross-compile the wrapper for the ARM7 architecture.

5.1.1 Initial Setup

It was important to first become familiar with the functioning of a BOINC wrapper by running the compiled version for Linux. We ran examples of the source code repository (https://github.com/BOINC/BOINC). By observing the correct wrapper execution in Linux, it became easier to differentiate configuration errors from execution errors in Android.

For the wrapper to work correctly, the file *job.xml* must be created and configured. This is the first file the wrapper reads to know what application to execute, how to execute it (additional commands, sequence of workunits, input parameters, etc.) and how to locate the resources that the application requires. Additionally it is necessary to create and configure several additional files specified in BOINC web pages (http://boinc.berkeley.edu/trac/wiki/WrapperApp).

5.1.2 Cross-compilation and execution

Current Android OS phones run on *ARM* CPUs but are not shipped with a pre-installed native compiler. Hence, tools and application codes for *ARM* architectures need to be generated on a platform where C++ compiler is available. Using a compiler to generate executable code for a platform different than the one where compilation takes place, is a process named **cross-compilation**. We used a Docker container (https://github.com/dockcross/dockcross) in the cross-compilation process since it has pre-built and configured toolchains for cross-compiling to different platforms including Android-ARM.

Although there is sufficient information about the process of cross-compiling applications in the BOINC documentation, there is little documentation about how to compile the wrapper for specific platforms such as Android. To build the wrapper it is necessary to execute the script *build_wrapper_arm.sh*. The required steps and problems solved to generate and execute the wrapper are:

1. Pre-configure the Android toolchain using the NDK provided by Google. NDK contains everything needed to do Android application development using C/C++. It was necessary to download the version 10e of the NDK.
2. Cross-compile Curl and OpenSSL libraries for Android. Since it was not possible to find pre-compiled versions of these libraries, it was necessary to generate the corresponding executables from the source code downloaded from *github.com*. Additionally, the script *build_wrapper_arm.sh*. required a specific version of *Curl* to construct the library *libBOINCapi*. We deleted this validation of the *configure.ac* file.
3. Add the flags (–fPIE and -pie) to the script *build_wrapper_arm.sh*. When trying to run the wrapper on mobile devices with an Android versions greater than or equal to 5.0 (Lollipop), a run-time error was reported because the program was not a PIE (Position Independent Executable). Android security page reports that Android versions greater than 5.0 require that all programs that want to run on their platform should contain the PIE security feature (https://source.android.com/security/enhancements).
4. Replace the call to *PWD* by the function *getcwd* in the source code of the wrapper. On Android, the *PWD* variable does not exist or is not visible to the programmer. Therefore, we replace it by the function *getcwd* that is part of the C library *unistd.h*.

The compiled wrapper was placed in the official repository of BOINC source code (https://github.com/BOINC/boinc/pull/1671).

5.2 Cross-compilation of ITK Modules

ITK [12] is a library for performing registration and segmentation of images. This library together with other libraries, such as VTK [29] and IGSTK [30] are the mostly used open-source libraries in the field of processing and analyzing medical images. They have been extensively tested and their robustness, flexibility and extensibility are guaranteed. All these characteristics establish them as standard setters [31].

The ITK library is implemented in C++ and it has cross-platform support using the CMake build environment to manage the configuration process. The library provides a

wide range of segmentation, registration and image filtering techniques.

A requisite to use this library together with BOINC's infrastructure is to generate the executables for the target platform, in this case Android ARM7, with the help of the CMake tool. ITK consists of multiple subsystems and since version 4.0.0, launched in December 2011, was officially modularized. Currently, ITK consists of more than 100 internal and other remote modules, which are grouped in *Core, ThirdParty, Filtering, IO, Bridge, Registration, Segmentation, Video, Compatibility, Remote, External and Numerics*.

As previously mentioned, the cross-compilation process is performed in a platform other than the target. Once the executable is generated, it is placed on the final platform. The cross-compilation of some ITK modules is more complex because their construction requires the generation of intermediate executable programs that must be invoked during the process to complete the construction of the whole executable code successfully. Since these executables are generated for the target ARM platform, they cannot be executed inside the platform in which the cross-compilation is being performed (e.g. Linux, Windows, etc.). For this reason, we could not cross-compile the entire toolkit in this first step of our work. The modules built in this project were: *Core*, *Filtering* and *Smoothing Recursive Gaussian Filter*, which do not present the problem described above.

Also, it was necessary to deactivate the test flag *BUILD TESTING*. When this flag is ON, some tests are made at the end of library construction to check the product. These tests assume that the system is running in the target platform, which is not true in this case.

The last problem was associated to the resulting executable code that uses the ITK library [12]. This code loads libraries at runtime into its object factory and links against shared libraries, which are features used by default. Since it is impossible to ensure that the target mobile devices have the necessary software installed to run the algorithms, we generated a self-contained static executable code without any shared dependencies. This was accomplished by turning off the flags *ITK_DYNAMIC_LOADING* and *BUILD_SHARED_LIBS*. This solution increases the executable's size significantly.

5.3 Parallel Processing of the Image

The cross-compilation of the ITK modules allowed us to generate the executable code to process the image.

For the parallel processing we developed code to: a) automate the generation of workunits giving the corresponding parameters (i.e. parts of the image) and b) collect the outputs and generate the final result. Two BOINC components were extended to address these issues: the **Work Generator** and the **Assimilator**.

Additionally, we implemented a program that divide the image and interact with the work generator to create the workunits.

5.3.1 The Work Generator

This new component uses the adapter pattern, allowing the integration of a programmer-provided data-division algorithm with BOINC. Figure 3 shows the class diagram. The class *WorkDivisor* implements the algorithm that divides the image in N sections which are then placed in N different files. The algorithm was programmed in C++. The *WorkDivisor* communicates with the *WorkGeneratorAdapter* and creates a workunit for each file previously created. The *WorkGeneratorAdapter* communicates with the BOINC server's API and returns to the programmer the workunit id. The *WorkGeneratorAdapter* was programmed using Python.

5.3.2 The Assimilator

The objective of this component is to collect individual results generated by each finished workunit and to construct the resulting processed image. By default, once a workunit is completed and validated, BOINC deletes all the files uploaded by the client. Hence, the need to immediately process the results or save them. Currently BOINC offers an Assimilator module written in Python that does not perform any function but can be extended.

To extend this module it is necessary to know some details of BOINC, namely, to know where do the outputs get stored and understand how the outputs are related to each workunit.

In the development of the Assimilator, we set two objectives:

- Simplify the work of the programmer, as he/she will only receive the output file's location of the canonical result.
- Create an extensible Assimilator for linking images that can be adopted for multiple algorithms.

In order to fulfill these requirements, we implemented the *observer pattern*. In this pattern, one observer contains

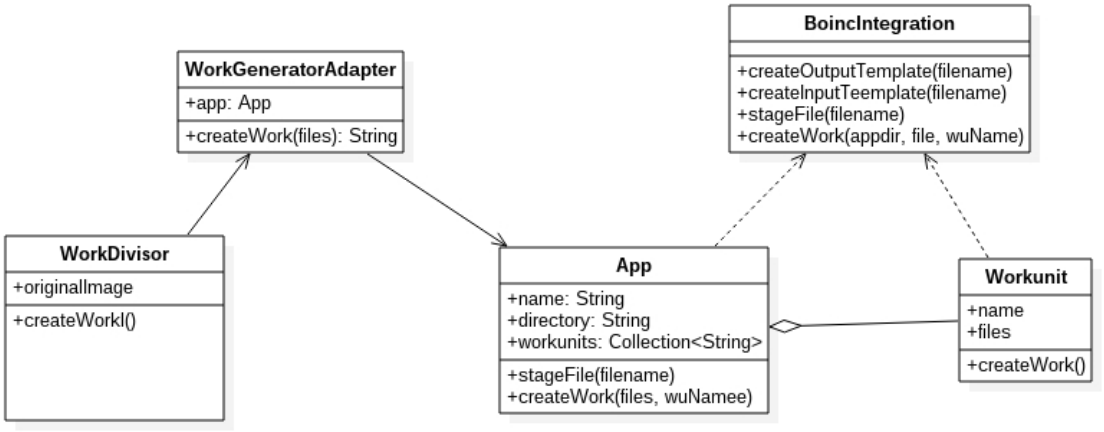

Figure 3: Class Diagram of the Work Generator

a programmer-implemented algorithm to merge the data. The *ConcreteObserver* class is responsible for both, to periodically detect when a workunit is ready to be assimilated, and notify the changes to the *MergeWork class*, that joins the various parts of the image. This class was implemented in the C++ language. The *ConcreteObserver* class, which was programmed in Python, uses the BOINC API to look for completed workunit. Figure 4 shows the class diagram of the *Assimilator*.

6 Experiments

In this section we describe a performance evaluation of the proposed problem. Initially we performed a base experiment to have the execution time of *Smoothing Recursive Gaussian Filter* on a single desktop computer without the use of BOINC. Then we conducted a 2^k experimental design to evaluate how several factors affect the performance of the parallel processing in BOINC.

6.1 Base Test

This base test was executed in an 8 core Intel Core i7-920XM 2.0 GHz processor, 4GB of RAM and running Linux *x*86_64 Ubuntu 14.04 operating system. We used a tiff image of 301 MB, 30576 pixels wide and 9860 pixels high. We took the image from the LROC project (*The Lunar Reconnaissance Orbiter Camera*, http://lroc.sese.asu.edu/images). The image can be observed in Figure 5. This image was selected for its large size and format, which is one of the formats used in medical images. The running time of the base test was 49 minutes.

6.2 2^k Factorial Design

This type of experimental design pretends to determine the effect or impact of k factors on a response variable, each of which has 2 alternatives or levels [32]. This is an exploratory experimental design used to determine what factors have more influence on the response variable. Once the most important factors are determined, it is possible to perform more in-depth experiments with them using a higher number of levels. In our 2^k design, we selected four relevant factors to the project and the *total execution time* as response variable. The image and the filtering algorithm were the same in all the experiments. We explained the selected factors below and shown them in table 2.

- **Processing Devices (mobile or desktop)**: This factor referred to technical characteristics of BOINC clients that would process the image (see Table 1).
- **Degree of parallelism**: This factor indicated the number of regions in which the original image was divided and, as a result, the number of workunits required to complete the total processing of the image. The two alternatives were 450 and 1000 workunits, each having image of regions of sizes 2MB and 900KB respectively.
- **Redundancy**: This factor specified how many results were required for a workunit to be successfully completed. For this factor, we used the levels: 1 for no redundancy and 2 as the number of results required to accept a response. These two levels would represent two feasible scenarios: in the first one, all grid devices belong to the organization that generates the works

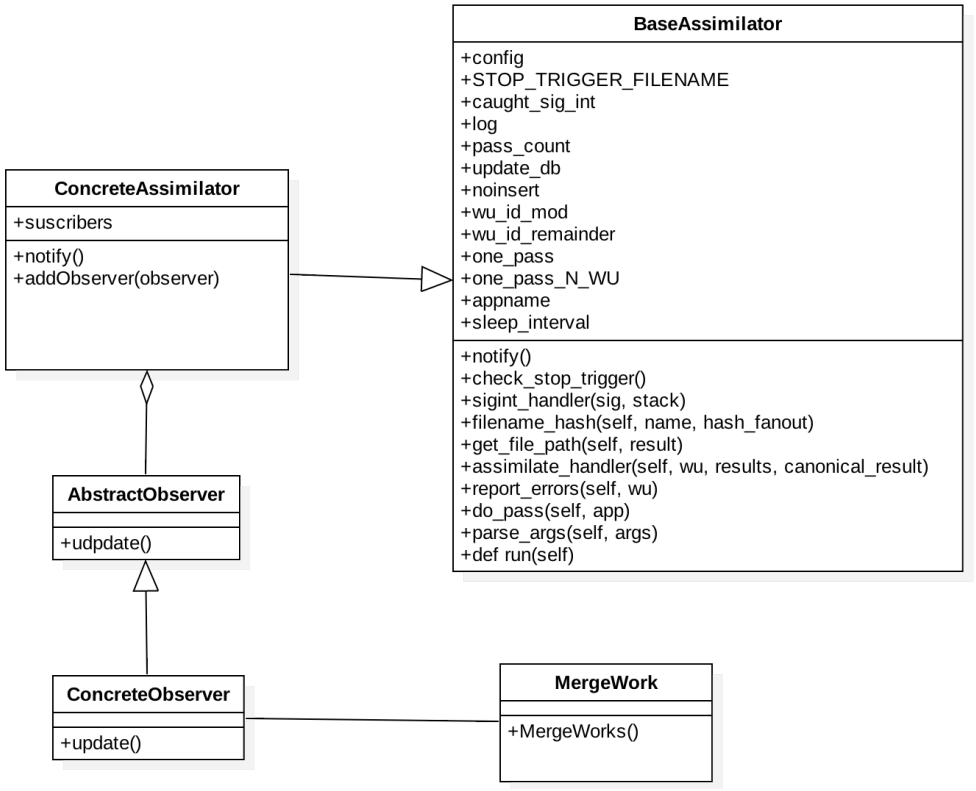

Figure 4: Class Diagram of the Assimilator

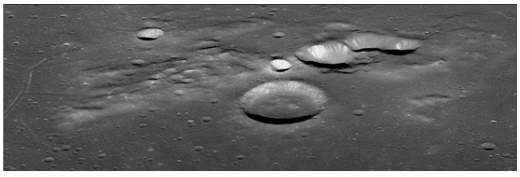

Figure 5: Image from the moon (http://lroc.sese.asu.edu/images)

and therefore, the results are considered trusted. In the second scenario, grid clients could be unknown volunteers, so at least 2 results are required to declare them as valid.

– **Server Location**: This factor was used to determine the impact of the latency due to the location of the BOINC server. We defined two levels: a local server and a remote server. The first one was a computer with Ubuntu 14.04, 4GB of RAM, 8 CPUs and a hard disk of 50GB located in the same network as the BOINC clients; the second one was a VPS provided by AWS (Amazon Web Services) with Ubuntu 14.04, 1GB of RAM, 1 VCPU and 10GB of SSD, located in a remote network.

6.3 Discussion

Table 3 presents the execution times obtained with each combination of factors. Figures 6, 7 and 8 show the execution times of two factors at once. Table 4 shows the percentage of variation on the response variable explained by each factor and their interactions. From the table, it is clear that the factors that affect the most the response variable are the server location and redundancy. Other factors and their interactions do not impact significantly the response variable. The third-order interactions are negligible; therefore, we do not present it. In the next paragraphs, some of the aspects will be explored in detail.

6.3.1 Server Location

Figure 6 shows execution times of the base test and parallel executions, which correspond to the factors devices and servers. As it is shown in Figure 6, the execution times of the parallel solutions with a local server were reduced more than 50%, improving even the sequential solution. Setting the redundancy in 2 also causes the times to in-

crease by more than 50% (Figure 8). When the performance is important, the recommendation is that both the BOINC server and the clients are on the same local network.

6.3.2 Redundancy

Redundancy has a significant impact in the execution time because it increases the number of workunits that must be sent and processed in the devices. As it was mentioned in Section 4, redundant computing can be a scheduling strategy to deal with problems that arise when using mobile devices as resource providers. With mobile devices new significant challenges appear, which are mainly due to mobility patterns and limitations in processing, memory, battery power and wireless communication capabilities. The limited battery and mobility patterns can cause intermittence of mobile devices, which could lead to a break of running tasks and to their subsequent rescheduling.

In [33] the scheduling strategies in mobile grids are divided into preventive and reactive. **Preventive scheduling strategies** try to ensure the task culmination either by selecting the most reliable resource to execute the grid tasks or using redundant resources in the execution (replication). Reliability in resource selection can be obtained by collecting the current status of the devices, using predictive models, etc. **Reactive scheduling strategies** act after the tasks have been assigned to resources. In this case at least two scenarios can be possible: a) a new assignment could be necessary when resources become unavailable because of unexpected events or faults; b) re-assignments could be necessary to balance the load and to maximize the number of tasks that can be completed.

BOINC supports redundant computing, so it is possible to implement a preventive strategy based on replication. The works presented in [34–36] explore these strategies. However, the performance obtained with the use of replication for this particular problem (see figure 8) lead us to think of other types of preventive or reactive strategies, especially if fast processing of the image is needed.

In BOINC, it is also possible to implement a predictive model based on the current status of the resources. BOINC clients send their characteristics before requesting a workunit; the scheduler use this information for the assigment of workunits. Part of these characteristics can be the current status of the battery to parameterize a predictive model. The works described in [37–39] exemplify this type of strategy. Also it is possible to implement rescheduling using the parameter **delay_bound**, which is specified when the workunit is created. The parameter **delay_bound** is an upper bound on the time (in seconds) between sending a result to a client and receiving a reply. If the client does not respond within this interval, the server gives up on the result and generates a new result, which will be assigned to another client.

BOINC group has some recommendations to avoid battery consumption due to the use of the mobile device for volunteer computing (https://boinc.berkeley.edu/trac/wiki/AndroidBoinc). They recommend to do computing only when the device is recharging. During these periods the device typically is on Wi-Fi. To avoid using up cell-phone data transfer quotas, they recommend to do network communication only over Wi-Fi.

6.3.3 Devices

In table 3 and figures 6, 7 and 8, we did not observe a significant difference in execution times when comparing desktop computers with mobile devices: the highest difference when BOINC is running at a local server is 26%. The percentage impact of this factor on the response variable is negligible. This is a positive point with respect to using one or the other as executing platform. The best option may be to use both types of devices when possible.

6.3.4 Degree of parallelism

The parallel solution with mobile devices is better than the sequential one when the server is local and the image is divided in 450 workunits. The solution with desktops for the same case has a slightly better performance. Table 5 shows the speedup values computed for execution times obtained in a local network with no redundancy. These were the best results. Although the parallel solution is better than the sequential one, the speedup is not proportional to the large number of processors and it decreases significantly when the number of processors is more than double. The speedup is better in desktop computers compared to mobile devices. This makes the efficiency very poor: we obtained 0,007 and 0,005 for desktop computers and mobile devices with Android respectively, with 450 processors. We compute the efficiency only for the best speedup values. It is important to remark that we chose high values for the number of processors in this first phase of the experiments, thinking about a problem of voluntary computing where a substantial number of users participate. Our preliminary results have demonstrated that the parallelism could be adequate but in lower degrees. A finer

performance evaluation of the parallel solution is left for the next phase of the project.

6.3.5 Resource Usage

Regarding the use of BOINC, it is important to mention that it uses a non-intrusive model in clients, characterized by low resource utilization. Thus, users can continue using their devices without perceiving a drop in performance of their running applications.

In the base test, when the image was processed in a desktop computer without BOINC, the algorithm monopolized all processing units, and the CPU usage reached 100%. In contrast, the CPU usage during the parallel processing of the image using BOINC was around 5%. This result is interesting because even using a much lower percentage of resources it was possible to significantly reduce the execution time by parallelizing the image processing. This also implies that, with an advanced configuration of the BOINC project, which allows, for example, to request more RAM or storage capacity in each client, it may be possible to obtain better execution times.

Table 1: Technical characteristics of the devices

Num. of Devices	Architecture	Model	CPU	Cores	RAM
1	Linux x86_64	Lenovo ThinkPad W510	Intel Core i7-920XM 2.0 GHz	8	8GB
5	Linux x86_64	Lenovo ThinkCentre Serie M	Intel Core i7 6700T 2.8 GHz	8	8GB
4	Android ARM7	Samsung Galaxy S4	Quad-core 1.9 GHz Krait 300	4	2GB
1	Android ARM7	Nexus 4	Quad-core 1.5 GHz Krait	4	2GB

7 Related Work

Mobile Grids are a type of Accessible Grid [5] where devices are used as resources to execute tasks. According to the revised literature, some implemented mobile or wireless grid systems are: IBIS [25], MoGrid [26], Akogrimo [40] MORE [41], MiPeG [42], Mobile OGSI.NET [35], BOINC [43], Misco [44] and ELASTIC [45]. M. Black and W. Edgard [46] proposed the use of mobile devices as grid computing nodes. They implemented a BOINC client on an APPLE iPhone to demonstrate the feasibility of this concept. Other researchers have proposed the use of mobile devices as execution platforms without the support of a grid system (e.g. [21, 47, 48]).

Several articles are devoted to basic definitions and stand out the main challenges of mobile grids ([7–9, 49, 50]). The adoption of mobile devices as CPU platforms implies a number of challenges. While most of these challenges come from technical features of the mobile devices (i.e. heterogeneity, CPU capacity, screen size, short life of the battery, mobility, and intermittent disconnections), others are related to the possibility that mobile owners will allow their devices to be part of the Grid.

Regarding technical issues, there is a group of papers devoted to solve specific problems such as: middleware architectures ([25, 42, 51–53]), resource management ([15, 33, 54–61]) or scheduling ([37, 62–67]). Authors of [68] apply good programming practices and code refactoring to reduce battery consumption of scientific mobile applications running on Android.

The search for incentives for users to place CPU cycles of their mobile devices at the disposal of voluntary or collaborative computer project is currently a research problem. Duan et al. [3] designed an incentive mechanism for smartphone collaboration in distributed computing. Van de Wijngaert et al. studied in [69] the circumstances under which people could share resources on their edge devices (PDA's, laptops and mobile phones) using a technology with which they are not yet familiar (wireless grids). Tapparello et al. [1] presented the current state of the art of the mobile volunteer computing. Their analysis includes motivations and challenges of adopting mobile devices as computing resources.

Mobile devices have also been widely used in various health contexts. Most of the applications developed help to manage and monitor diseases with the help of sensors

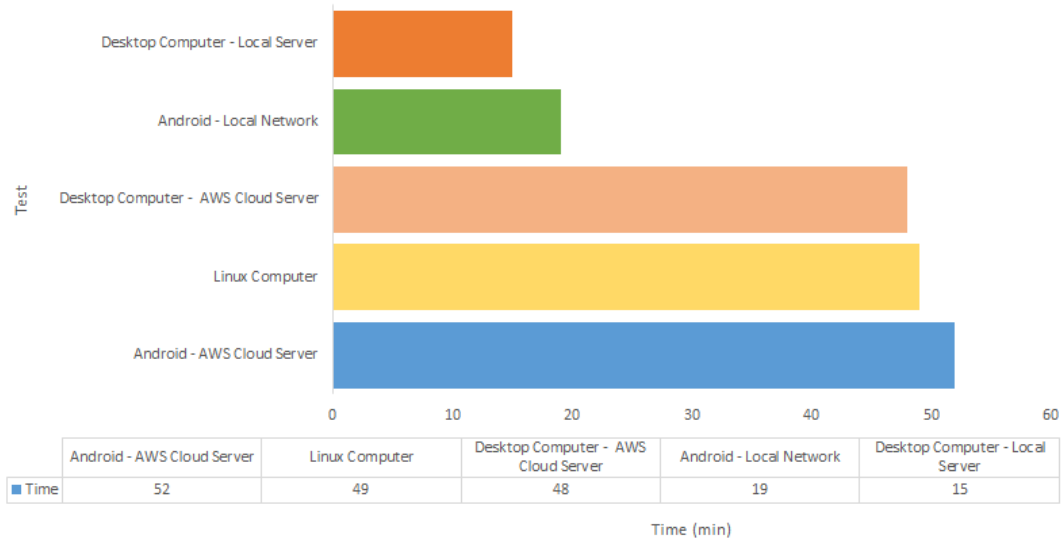

Figure 6: Impact of the server location

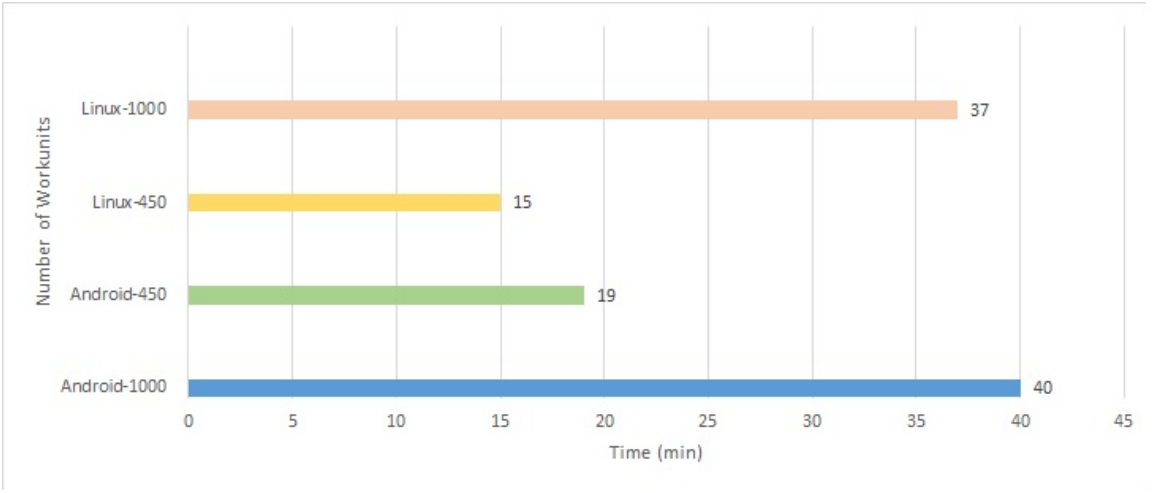

Figure 7: Impact of the degree of parallelism

Table 2: Factors and levels

Factor	Low Level (-1)	High Level (+1)
Devices, A	Linux *x*86_64 computers	Mobile devices Android ARM7
Degree of parallelism, B	450	1000
Redundancy, C	1	2
Server, D	Local	Remote

located on mobile devices. Articles [70, 71] present a review of these works. In the search for related works, there were no similar experiences of the use of BOINC and mobile devices for parallel processing of images. In [72], authors present the development of software for electrocardiogram analysis using mobile devices. In that work, however, the use of one device was sufficient for the amount of processing required.

8 Conclusions and Future Work

The proposal of our research is to explore the use of a mobile grid for parallel processing of medical images. The solution of this problem using BOINC presented at least four challenges: 1) The execution of programs in the mobile grid without changing the source code; 2) Cross-

Table 3: Results of the 2^k experiment

		desktop computers		mobile devices	
		450 workunits	1000 workunits	450 workunits	1000 workunits
Local	Redundancy 1	15min	37min	19min	42min
	Redundancy 2	38min	1h22min	40min	1h26
Remote	Redundancy 1	48min	59min	54min	1h03min
	Redundancy 2	1h37	1h54min	1h52min	2h01min

Table 4: Percentage per factor

A (Devices)	B (Parallelism)	C (Redundancy)	D (Server)	AB	AC	AD	BC	BD	CD
0%	6%	47%	30%	2%	1%	1%	1%	0%	6%

Table 5: Speedup. Local Server with no Redundancy

	450 processors	1000 processors
Desktop computer (Linux)	3.27	1.32
Mobile Devices (Android)	2.58	1.17

compilation of the ITK library for image processing; 3) Distribution and collection of data for parallel processing; 4) The achievement of acceptable performance. In this article, we presented solutions to these challenges.

Related to the **first challenge**, it was possible to compile the wrapper of BOINC for Android mobile devices. We made this contribution to the official repository of BOINC source code. The wrapper allowed the execution of our programs in Android mobile devices without modifying the code. With regard to the **second challenge**, part of ITK library was compiled for Android. This was achieved by deactivating the flags that perform tests on the target platform and disabling the load of libraries at runtime. We generated a static executable including the *Core* and *Filtering* ITK modules.

Using BOINC and Android mobile devices, it was possible **to apply a filter in parallel** to a 301 MB image. BOINC supports the distribution of workunits in the computing platforms. However, when the SIMD model is used, some code must be provided to distribute and collect data between clients and the server. The development of this code can be laborious since it requires a deep understanding of BOINC functioning. Therefore, we implemented a Work Generator that allows the programmer to provide her/his own image division algorithm, which can be integrated to BOINC infrastructure without exhaustive knowledge of its API. Our implemented assimilator simplifies the process of unifying results and abstracting details of how BOINC works. This gives an answer to the **third challenge**.

With regard to **performance**, we obtained very promising results. Execution times of image processing were very similar in mobile devices and desktop computers. For most scenarios involving a local server, the execution times of the parallel solution decreased significantly with respect to sequential. However, it is important to conduct more experiments in parallel scenarios to explore the behavior of metrics such as speedup and efficiency.

Additionally, the platform did not fully invade the resources of the mobile devices. All these reasons make the use of BOINC as mobile grid interesting and attractive for the problem raised.

Once the feasibility of our goal has been proven, future work includes:

- Continuing experiments with several types of BOINC clients (fixed and mobiles) and different techniques of image processing. Study the performance of several hardware-software configurations with new metrics, such as resource usage.
- Finding use cases for this technology. This involves answering questions such as: Which situations or scenarios will this technology be adequate in? In what type of medical institutions? What types of medical problems will this technology be adequate in? Who would be the volunteers? What could be the incentives for them?

References

[1] Tapparello C., Karaoglu C.F.B., Ba H., Hijazi S., Shi J., Aquino A., et al., Volunteer computing on mobile devices: State of the

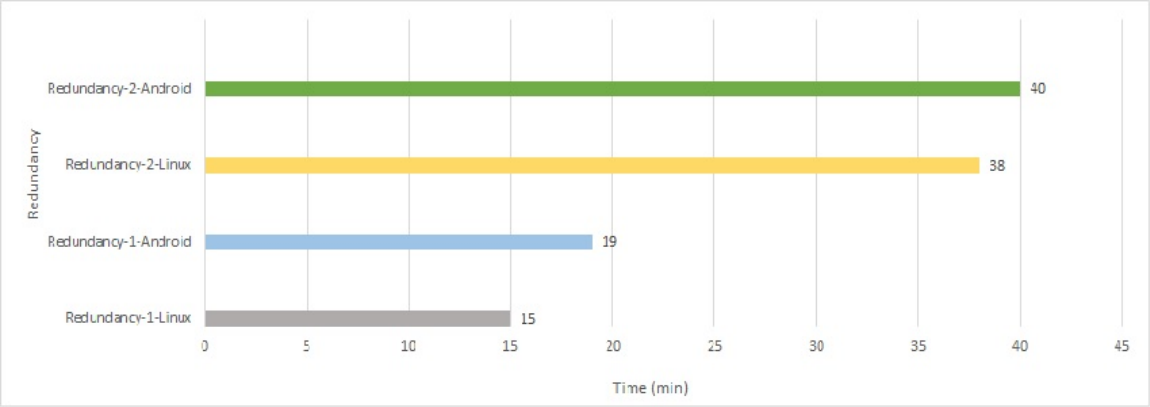

Figure 8: Impact of the redundancy

art and future, *Enabling Real-Time Mobile Cloud Computing through Emerging Technologies*, 2015, 153–181.

[2] Rodríguez J.M, Mateos C., Zunino A., Are smartphones really useful for scientific computing?, *Advances in New Technologies, Interactive Interfaces and Communicability*, Springer, 2012, 38–47, DOI: 10.1007/978-3-642-34010-9_4

[3] Duan L., Kubo T., Sugiyama K., Huang J., Hasegawa T., Walrand J., Motivating smartphone collaboration in data acquisition and distributed computing, *IEEE Transactions on Mobile Computing*, 2014, 13, 10, 2320–2333, DOI:10.1109/TMC.2014.2307327.

[4] Foster I., Kesselman, C., *The Grid 2: Blueprint for a new computing infrastructure*, Elsevier, 2003.

[5] Kurdi H., Li M., Al-Raweshidy H., A classification of emerging and traditional grid systems, *Distributed Systems Online, IEEE*, 2008, 9,3,DOI: 10.1109/MDSO.2008.8

[6] Furthmüller J., Waldhorst O.P., A survey on grid computing on mobile consumer devices, *Handbook of Research on P2P and Grid Systems for Service-Oriented Computing*, 2010, 313–363,DOI: 10.4018/978-1-4666-0879-5.ch510.

[7] Ahuja S.P, Myers J.R., A survey on wireless grid computing, *The Journal of Supercomputing*, 2006, 37,1,3–21, DOI: https://doi.org/10.1007/s11227-006-3845-z.

[8] Manvi S.S., A review of wireless grid computing, *International Journal of Computer and Electrical Engineering*,2010, 2,3,1793–8163,

[9] Parmar K., Jani N., Shrivastav P., Patel M., Mobile grid computing: Facts or fantasy?, *International Journal of Multidisciplinary Sciences and Engineering*, 2013, 4,1.

[10] Rodríguez J.M., Zunino A., Campo M., Introducing mobile devices into grid systems: a survey, *International Journal of Web and Grid Services*, 2011, 7, 1, 1–40, DOI:https://doi.org/10.1504/IJWGS.2011.038386.

[11] Anderson D.P., A system for public-resource computing and storage, In: *ACM International Workshop on Grid Computing*, (November, 2004, Pittsburgh), PA (November 8, 2004), 2004.

[12] Ibañez L., Schroeder W., Ng L., Cates J., The itk software guide, 2005.

[13] Young I.T., Van Vliet L.J., Recursive implementation of the gaussian filter, *Signal processing*, 1995, 44,2,139–151.

[14] Foster I., Kesselman C.,Nick J.M., Tuecke S., The physiology of the grid, an open grid services architecture for distributed systems integration, Technical report, Global Grid Forum, 2002.

[15] Hijab M., Avula D., Resource discovery in wireless, mobile and ad hoc grids: Issues and challenges, In:*Proceedings of the 13th International Conference on Advanced Communication Technology (ICACT), (13-16 February, Seoul, South Korea)*, 2011.

[16] Marinescu D.C., Marinescu G.M., Ji Y., Boloni L.,. Siegel H.J., Ad hoc grids: Communication and computing in a power constrained environment, In: *Proceedings of the 2003 IEEE International Conference on Performance, Computing, and Communications (9-11 April 2003, Phoenix, USA)*, IEEE, 2003, 113–122. DOI: 10.1109/PCCC.2003.1203690.

[17] Sarmenta L.F.G., *Volunteer computing*. PhD thesis, Massachusetts Institute of Technology, USA, 2001.

[18] Anbarjafari G., Digital image processing, URL https://sisu.ut.ee/imageprocessing/book/1.

[19] Glasbey C.A., Graham H., *Image analysis for the biological sciences*, Wiley Chichester, 1995,1.

[20] Fisher R., Perkins S., Walker A., Wolfart E., Image processing learning resources, 2000, URL: http://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr_top.htm.

[21] Datta P., Dey S., Paul H.S., Mukherjee A., Angels: A framework for mobile grids, In: *Applications and Innovations in Mobile Computing (AIMoC) (27 February-1 March 2014, Kolkata, India)*, IEEE, 2014, 15–20, DOI: 10.1109/AIMOC.2014.6785513.

[22] Arslan M.Y., Singh I., Singh S., Madhyastha H.V., Sundaresan K., Krishnamurthy S.V., Computing while charging: building a distributed computing infrastructure using smartphones, In: *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, ACM, 2012, 193–204, DOI:10.1145/2413176.2413199

[23] Büsching F., Schildt S., Wolf L., Droidcluster: Towards smartphone cluster computing–the streets are paved with potential computer clusters, In: *32nd International Conference on Distributed Computing Systems Workshops (ICDCSW), (18-21 June 2012, Macau, China*, IEEE, 2012, 114–117, DOI: 10.1109/ICDCSW.2012.59.

[24] García R., Flórez-Valencia L., Curiel M., On existing mobile grids for android devices, In: *8th Euro American Conference on Telematics and Information Systems (EATIS)* (27-29 April 2016, Cartagena, Colombia), IEEE, 2016, 1–7, DOI: 10.1109/EATIS.2016.7520117.

[25] Palmer N., Kemp R., Kielmann T., Bal H., Ibis for mobility: solving challenges of mobile computing using grid techniques, In: *Proc.*

of the 10th workshop on Mobile Computing Systems and Applications (23-24 February 2009, Santa Cruz, California), 2009, DOI: 10.1145/1514411.1514426.

[26] dos S Lima L., Gomes A., Tadeu A., Ziviani A., Endler M.,Soares L.F.G., et al., Peer-to-peer resource discovery in mobile grids, In: *Proceedings of the 3rd international workshop on Middleware for grid computing* (28 November-02 December 2005, Grenoble, France), ACM, 2005, 1–6, DOI: 10.1145/1101499.1101510.

[27] Andrade N., Cirne W., Brasileiro F., Roisenberg P., Ourgrid: An approach to easily assemble grids with equitable resource sharing, In: Springer, Berlin, Heidelberg, *Job scheduling strategies for parallel processing*, 2003, 61–86, DOI: https://doi.org/10.1007/10968987_4

[28] Mateos C., Zunino A., Campo M., A survey on approaches to gridification, Software: Practice and Experience, 2008, 38, 5, 523–556. DOI:110.1002/spe.847.

[29] Schroeder W.J., Avila L.S., Hoffman W., Visualizing with vtk: a tutorial, *IEEE Computer graphics and applications*, 2000, 20,5,20–27.

[30] Gary K., Ibanez L., Aylward S., Gobbi D., Blake M.B., Cleary K., Igstk: an open source software toolkit for image-guided surgery, *Computer*, 2006, 39,4,46–53, DOI: 10.1109/MC.2006.130.

[31] Caban J.J., Joshi A., Nagy P., Rapid development of medical imaging tools with open-source libraries, *Journal of Digital Imaging*, 2007,20,1,83–93, DOI:https://doi.org/10.1007/s10278-007-9062-3.

[32] Jain R., *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*, John Wiley & Sons, 1990.

[33] Curiel M., Wireless grids: Recent advances in resource and job management, In: *Handbook of Research on Next Generation Mobile Communication Systems*, IGI Global, 2016, 293–320, DOI: 10.4018/978-1-4666-8732-5.ch012.

[34] Chin S., Suh T., Yu H., Genetic algorithm based scheduling method for efficiency and reliability in mobile grid, In: *Proceedings of the 4th International Conference on Ubiquitous Information Technologies & Applications, ICUT'09, (20-22, December, Fukuoka, Japan)*, IEEE, 2009, 1–6, DOI: 10.1109/ICUT.2009.5405741.

[35] Litke A., Halkos D., Tserpes K., Kyriazis D., Varvarigou T., Fault tolerant and prioritized scheduling in ogsa-based mobile grids, *Concurrency and Computation: Practice and Experience*, 2009, 21,4, 533–556, DOI: 10.1002/cpe.1351.

[36] Du L., Yu Z., Scheduling algorithm with respect to resource intermittence in mobile grid, In: *6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM) (23-25 September 2010, Chengdu, China)*, IEEE, 2010, 1–5, DOI: 10.1109/WICOM.2010.5600181.

[37] Vaithiya S.S, Bhanu S.M.S., Scheduling tasks in mobile grid environment using mobility based resource prediction, In: *2010 1st International Conference on Parallel Distributed and Grid Computing (PDGC),(28-30 October 1010, Solan, India)*, IEEE, 2010, 89–94, DOI: 10.1109/PDGC.2010.5679600.

[38] Lee J., Song S., Gil J., Chung K., Suh T., Yu H., Balanced scheduling algorithm considering availability in mobile grids, In: *International Conference on Grid and Pervasive Computing* (4-8 May 2009, Geneva, Switzerland), Springer, 2009, 211–222.

[39] Liu L.,Li C., Mobile grid task scheduling considering resource reliability, In: *International Symposium on Computer Network and Multimedia Technology, CNMT (18-20 January 2009, Wuhan, China)*, IEEE, 2009, pages 1–4, DOI: 10.1109/CNMT.2009.5374742.

[40] Jáhnert J.M., Wesner S., Villagrá V.A., The akogrimo mobile grid reference architecture-overview, whitepaper, 2008.

[41] Wolff A., Michaelis S., Schmutzler J., Wietfeld C., Network-centric middleware for service oriented architectures across heterogeneous embedded systems, In *EDOC Conference Workshop, 2007. EDOC'07 (15-16 October 2007, Annapolis, MD, USA)*, IEEE, 2007, 105–108, DOI: 10.1109/EDOCW.2007.20.

[42] Coronato A., De Pietro G., Mipeg: A middleware infrastructure for pervasive grids, *Future Generation of Computer Systems*, 2008, 24,1, 17–29, DOI:https://doi.org/10.1016/j.future.2007.04.007

[43] Anderson D.P., Boinc: A system for public-resource computing and storage, In: *Proceedings. Fifth IEEE/ACM International Workshop on Grid Computing (8 Nov 2004, Pittsburgh, PA, USA)*, IEEE, 2004, 4–10.

[44] Dou A., Kalogeraki V., Gunopulos D., Mielikainen T., Tuulos V.H., Misco: a mapreduce framework for mobile systems, In: *Proc. of the 3rd international conference on pervasive technologies related to assistive environments (23-25 June 2010, Samos, Greece)*, ACM, 2010, 32, DOI: 10.1145/1839294.1839332.

[45] Austinm D., Lai R., Elastic parallel execution framework on android, https://www.contrib.andrew.cmu.edu/ awdavis/15418/project/

[46] Black M., Edgar W., Exploring mobile devices as grid resources: Using an x86 virtual machine to run boinc on an iphone, In: *Proceedings of the 2009 10th IEEE/ACM International Conference on Grid Computing (13-15 October 2009, Banff, AB, Canada)*, IEEE, 2009, 9–16, DOI: 10.1109/GRID.2009.5353077.

[47] Yannes Z., Portable mpich2 clusters with android devices. M.Sc.thesis, The Florida State University, 2015.

[48] Arslan M.Y., Singh I., Shailendra S., Madhyastha H.V., Sundaresan K., Krishnamurthy S.V., Cwc: A distributed computing infrastructure using smartphones, *IEEE Transactions on Mobile Computing*, 2015, 14,8,1587–1600, DOI: 10.1109/TMC.2014.2362753.

[49] Gang L., Hongmei S., Gao H., Yu H., Cai Y., A survey on wireless grids and clouds, In: *Eighth International Conference on Grid and Cooperative Computing (27-29 August 2009, Lanzhou, Gansu, China)*, IEEE, 2009, 261–267, DOI: 10.1109/GCC.2009.44.

[50] Shah S., Recent advances in mobile grid and cloud computing, *Intelligent Automation & Soft Computing*, 2017, 1–13, DOI: http://dx.doi.org/10.1080/10798587.2017.1280995.

[51] Masinde M., Bagula A., Ndegwa V.. Mobigrid: a middleware for integrating mobile phone and grid computing, In: *2010 International Conference on Network and Service Management (CNSM), (25-29 October 2010, Ontario, Canada)*, IEEE, 2010, 523–526.

[52] Grace P., Hughes D., Coulson G., Blair G.S., Porter B., Taiani F., Overlay-based middleware for the pervasive grid, *Handbook of Research on P2P and Grid Systems for Service-Oriented Computing, IGI-Global*, 2010, 981–1002,DOI: 10.4018/978-1-4666-0879-5.ch302.

[53] McKnight L.W., Bose T., Marsden J., Nanno E., Treglia J., Volos H., et al., Open specifications for wireless grids technical requirements, *Version 0.1 approved: WiGiT Group. Syracuse, NY: School of Information Studies, Syracuse University*, 2012.

[54] Birje M.N., Manvi S.S., Multiagent model for device state control in the wireless grid, In: *2011 3rd International Con-*

ference on Electronics Computer Technology (ICECT), (8-10 April 2011,Kanyakumari, India, IEEE, 2011,3, 456–460, DOI: 10.1109/ICECTECH.2011.5941834.

[55] Birje M.N, Manvi S.S., Wigrimma: A wireless grid monitoring model using agents, *Journal of Grid Computing*,2011, 9,4, 549–572, ISSN 1570-7873, DOI: https://doi.org/10.1007/s10723-011-9181-4.

[56] Sedaghat M., Othman M., Agent based resource brokering and allocation in wireless grid: Revisited, *Journal of Theoretical and Applied Information Technology*, 2009.

[57] Birje M.N., Manvi S.S, Das S.K., Reliable resources brokering scheme in wireless grids based on non-cooperative bargaining game, *Journal of Network and Computer Applications*, 2014, 39, 266–279, DOI: https://doi.org/10.1016/j.jnca.2013.07.007

[58] Zhang Y., Pei Y., A resource discovery algorithm in mobile grid computing based on ip-paging scheme. In: Chen Ding, Zhiyuan Shao, and Ran Zheng, editors, *Network and Parallel Computing (13-15 september, Zhengzhou, China)*, *Lecture Notes in Computer Science*,Springer Berlin Heidelberg, 2010, 6289, 402–411, ISBN 978-3-642-15671-7.

[59] Farooq U., Khalil W., A generic mobility model for resource prediction in mobile grids, In: *International Symposium on Collaborative Technologies and Systems, CTS 2006 (14-17 Mayo 2006, Las Vegas, NV, USA)*, IEEE, 2006, 189–193, DOI: 10.1109/CTS.2006.7.

[60] Farooq U., Mahfooz S., Khalil W., An efficient resource prediction model for mobile grid environments, *Punjab University College of Information Technology Lahore, Pakistan*, 2006.

[61] Litke A., Skoutas D., Varvarigou T., Mobile grid computing: Changes and challenges of resource management in a mobile grid environment, In: *5th International Conference on Practical Aspects of Knowledge Management (PAKM 2004)*, 2004.

[62] Mahmood A., Elmallah E.S., Incremental routing and scheduling in wireless grids, In: *Global Telecommunications Conference, GLOBECOM 2009 (30 November-4 December 2009,Honolulu, HI, USA)*, IEEE, 2009, 1–6, DOI: 10.1109/GLOCOM.2009.5425699.

[63] Ghosh P., Das S.K., Mobility-aware cost-efficient job scheduling for single-class grid jobs in a generic mobile grid architecture, *Future Generation Computer Systems*,2010, 26,8, 1356–1367, DOI: https://doi.org/10.1016/j.future.2009.05.003.

[64] Birje M.N, Manvi S.S., Bulla C., Economical job scheduling in wireless grid, In: *Proc. of the 3rd International Conference on Electronics Computer Technology (ICECT), (8-10 April 2011, Kanyakumari, India)*, 2011, DOI: 10.1109/ICECTECH.2011.5941835.

[65] Katsaros K., Polyzos G.C., Evaluation of scheduling policies in a mobile grid architecture, In: *SPECTS 2008. International Symposium on Performance Evaluation of Computer and Telecommunication Systems, (16-18 June 2008, Edinburgh, UK)* , IEEE, 2008, 390–397.

[66] Adeyelu A., Olajubu E., Aderounmu A., Ge T., A model for coordinating jobs on mobile wireless computational grids, *International Journal of Computer Applications*,Foundation of Computer Science, New York, USA, 2013, 84,13,17-24.

[67] Litke A., Skoutas D., Tserpes K., Varvarigou T. Efficient task replication and management for adaptive fault tolerance in mobile grid environments, *Future Generation Computer Systems*, 2007, 23,2,163–178, DOI: https://doi.org/10.1016/j.future.2006.04.014.

[68] Rodríguez A., Mateos C., Zunino A., Improving scientific application execution on android mobile devices via code refactorings, *Software: Practice and Experience*, 2017, 47,5,763–796, DOI: 10.1002/spe.2419.

[69] van de Wijngaert L., Bouwman H., Would you share? predicting the potential use of a new technology, *Telematics and Informatics*, 2009, 26,1,85–102, DOI: https://doi.org/10.1016/j.tele.2008.01.002.

[70] West D., How mobile devices are transforming healthcare. *Issues in technology innovation*, 2012, 18,1, 1–11.

[71] Boulos M., Wheeler S., Tavares C., Jones R., How smartphones are changing the face of mobile and participatory healthcare: an overview, with example from ecaalyx, *Biomedical engineering online*, 2011, 10, 24. DOI: https://doi.org/10.1186/1475-925X-10-24.

[72] Folador. J.P., Desenvolvimento de um software para análise de eletrocardiogramas utilizando dispositivos móveis. MSc. thesis, Universidade Federal do Triângulo Mineiro, Uberaba, 2005.