Towards Pervasive Applications in a Grid Computing Environment

Gustavo Frainer, Luciano Cavalheiro da Silva, Cláudio Fernando R. Geyer
PPGC-II-Federal University of Rio Grande do Sul
Porto Alegre, RS, Brazil
frainer, lucc, geyer@inf.ufrgs.br

and

Iara Augustin
PPGI-INF-Federal University of Santa Maria
Santa Maria, RS, Brazil
august@inf.ufsm.br

and

João Ladislau B. Lopes, Adenauer Corrêa Yamin, Renata H. Sander
PPGINF-ESIN-Catholic University of Pelotas
Pelotas, RS, Brazil
joaolopes, adenauer, reiser@ucpel.tche.br

Abstract
In this paper we present a proposal for managing large-scale pervasive applications and discuss the features and services of its implementation, which was based on the integration of three computing fields: mobile, context-aware and grid computing. We also discuss how the adopted context-aware adaptation model, a main feature in both the applications and middleware design, helps to allow wide physical and logical mobility.

Keywords: Pervasive Computing, Context-Aware Adaptation, Middleware.

1 Introduction

The ISAM project (http://www.inf.ufrgs.br/isam)\(^1\) aims to study alternatives to simplify the design and execution of pervasive applications with physical and logical large-scale mobility [2, 3]. With this perspective, we are building the ISAM software architecture, which is composed by a middleware that provides support for language abstractions which enable the programming of large-scale pervasive applications, and also manages the application execution in a global scope.

In order to develop pervasive applications that are useful to the final user, it is necessary to make available development tools that absorb the complexity created by large-scale mobility. This requires an associated development environment with tools, language abstractions and a support layer. In this paper, we don’t discuss the language viewpoint, focusing instead in the support layer for the language provided by the middleware.

The wide pervasive scenario defines a personal virtual environment that should follow the user in his movements. These movements can involve both logical mobility (data, code and computation) and physical mobility (resources and devices in use). We refer to this feature of the environment as *follow-me semantics*.

We find in the support infrastructure of the grid computing field the basis to provide the global mobility of user, terminal, data, code and session. This way, the ISAM architecture approaches the integration aspects of three computing fields: grid, mobile and context-aware computing to manage the large-scale pervasive environment and to make available the follow-me semantics that allows to build large-scale pervasive applications.

\(^1\)ISAM project is supported in part by CNPq, RNP and FINEP Brazilian Foundations.
The integration of the grid and pervasive computing fields is complex due to the fact that the dynamism of the pervasive environment contrasts with the static resource management of traditional grid systems. We believe that both the pervasive and the grid computing fields gain advantage with this convergence. We let grid applications adjust themselves to current resources instead of looking for resources that satisfy their needs. We highlight that such an adaptation goes beyond an adjustment to a change in the number of available resources of a given kind, as we usually find in traditional grid solutions, because it allows applications to adapt to the kind of the available resources itself. In the pervasive computing side, we let the infrastructure manage a bigger environment and make decisions based on many context dimensions: temporal, spacial, personal or social.

In this paper, we present the concepts, the features and services to implement the follow-me semantics. We also argue that context-aware adaptation should be embedded in both applications and middleware design to allow for large-scale physical and logical mobility. Moreover, the adaptation is a process managed by the middleware and it occurs in a collaborative way with the application code. To allow the user’s virtual environment (composed by data, code, profile, permissions and others) to accompany the user’s displacement an infrastructure with wide visibility of the available resources is needed. So, our infrastructure converges to the grid computing field.

The rest of the paper is structured as follows. In section 2 the related work is presented. Our approach is summarized in section 3. Section 4 talks about the context-aware adaptation model used to design the large-scale pervasive applications and the middleware. Section 5 summarizes the services of the middleware responsible for supporting to follow-me semantics. Finally, section 6 closes this position paper.

2 Related Works

The research and projects of grid computing are currently focused in wired computers such as desktop PCs, supercomputers and clusters [7, 18]. So, mobile grid research is still weak. In general, mobile grids consider the use of PDAs as an access interface to the grid application results or as start points for application that in fact run in the static network [7, 6]. The works that are closest to our approach are listed in subsection 2.2.

Large-scale mobility can be provided in a number of ways [12] and the target of mobility can be several computational elements: user interface, devices, code, computation, data. Many works deal only with an independent element, as user interface [4], devices [11], code [9], computation [13], data [8]. This point is summarized in subsection 2.1. Nowadays, the difficulty is deal with the moving of many elements at same time. To accomplish that, an unified approach that involves an adaptive behavior to context is necessary.

The ISAM view of large-scale pervasive applications, whose management is based on solutions coming from grid computing field, differs from other pervasive computing projects, such as Aura [10] and Gaia [16], which originally proposed solutions focused on a personal view and a local view, respectively. Large-scale pervasive system is the focus focus of some recent works, as is described in subsection 2.2.

2.1 Grid Systems and Mobility

Our thesis is that grid solutions to resource (data, code, devices, others) management can enable large-scale pervasive systems. Some recent initiatives aim to join mobility and grid computing in a way similar to us: Akogrimo [19] and Ubigrid [6]. Akogrimo (www.mobile grids.org) adds the mobility dimension to the European Grid. In this vision, grid services, pervasively available, are eventually meeting the “every where at any time in any context” paradigm. The services, comprising of personalized knowledge and semantics, will allow ad-hoc, dynamic and possibly federated formation of problem solving scenarios, in business and science, to fixed, nomadic and mobile citizens. The application scenarios foreseen are e-health and e-education. Ubigrid (http://ubigrid.lancs.ac.uk/) addresses the use of grid technologies to support experiments in, and development and deployment of, ubiquitous computing systems. Since Akogrimo and Ubigrid are recent projects, a comparison among our solution and these projects is postponed for when the results become available.

Achieving automatic user interface mobility among several device platforms is a difficult task. The thesis argued by the MDAT project [4] is a hybrid design-time and runtime adaptation. Design-time adaptation converts a generic application into multiple device-specific versions before deploying the application on a server. Runtime adaptation converts a generic application into a device-specific version of the application on a Web server in response to a client request. MDAT addresses the Internet domain and, currently, supports the translation among XHTML, XHTML Mobile profile and WML.

Fluid computing denotes the replication and real-time synchronization of application states on several devices. Thus then application state flows as a ’fluid’ between devices [11]. In this solution, each device has a replica of the
application state, which allows it to operate autonomously. The replica consistency is weak, depending on the quality of the network connectivity. The Fluid Computing middleware is a lightweight Java library that runs on PDAs such as Sharp Zaurus or PocketPC devices.

Mobile computation was the theme of several researches developed a decade ago, specially in the operating systems field [13, 17, 14], but these didn’t produce a large use commercial product. The effort to accomplish the mobility of computation is costly in relationship to the advantages obtained with this process. Nowadays, the common approach is code mobility, referred as weak mobility [9]. The success of the Java language in the Internet domain can be attributed to code mobility. Code mobility is exploited on an Internet-scale, programming is location-aware and it is under programmer’s control. Mobile code technologies and a classification of mobility mechanisms are discussed in [9].

Ubiquitous data management has been studied in the database field [5, 8]. The functionalities required of new databases include support for mobility, context awareness and support for collaboration. One requirement is present across all of them: adaptivity. These aspects are discussed in [8]. On another hand, an interesting format for storage and communication is the Distributed Tuple Object, which allows an uncoupled temporal and spatial communication [15].

2.2 Large-Scale Pervasive Systems

Super Spaces [1] extends the concept of Active Spaces defined in the Gaia project [16] to enable large-scale management, operation and maintenance of pervasive computing environments. It emphasizes the creation of Super Spaces through the grouping of Active Spaces into bigger spaces. The framework allows two models of services and applications: hierarchical recursive model and peer-to-peer model. However, recursiveness and peer-to-peer approaches have a classical problem: performance. Super Spaces has a set of core services - naming, discovery, space repository, event management, file system and context services - that allow to perform operations in multiple subspaces and synchronize data and activities across different subspaces. The project is ongoing.

3 The ISAM Proposal

Our middleware uses grid strategies to manage the user’s displacement, accomplishing the migration of his virtual environment.

3.1 ISAMpe: Pervasive Grid Environment

The large-scale computational environment, referred as ISAMpe (ISAM pervasive environment), is formed by cells, formed by the union of the several mobile and stationary physical resources on the infrastructure network [21]. The ISAMpe is illustrated in figure 1. We can say that this organization is similar to Super Spaces [1], but its creation and management is different. We use a grid infrastructure to provide the follow-me semantic at a global level and organize the environment resources in a hierarchical form: (i) cell indoor organization is based on base nodes (EXEHDAbase) e components nodes (EXEHDAnode and EXEHDAmob-node); (ii) cell outdoor organization is based on peer-to-peer relationship among cells. This way, we adopt a Super-Peers organization. In this scenario, the mobile devices store neither code nor data persistently, but operate as windows (portals) that receive the code to be executed and can transfer the execution to other devices using proximity or resource availability as a selection criteria. The application code is installed on-demand on the used devices and this installation is adaptive, taking into account the context of each device. The computing environment elements (codes, data, services, devices) are spread and managed by the middleware, which provides pervasive access to them.

The key to allow the follow-me semantics is the migration of application components whose code will be adapted to the current state of the target environment (dynamic context-based adaptation) and the pervasive access to the user virtual environment. This dynamic behavior of applications must be managed by the system support at runtime, which obtains and provides all the necessary information to support it.

This way, the ISAM applications are distributed, mobile, adaptive and reactive to changes in the execution context. Moreover, ISAM applications are pro-actively managed by the middleware and they also express a follow-me semantics, i.e., the application follows the user as he moves in the large-scale space.

In the architecture, context is a first order concept and its meaning is “all the relevant information to the application that can be obtained from the support system”. In our model, the application explicitly identifies and defines the entities that characterize a situation and integrate its context. State alterations in these entities trigger an adaptation process in the application components. So, the context definition can then be refined to “every entity’s attribute for which a state alteration triggers an adaptation process in the application”. So, the context allows only focusing on
aspects relevant to a particular situation, while ignoring others. The Context Recognition Subsystem implements this concept (see figure 2).

Another notion present in the architecture is planned disconnections. Disconnections are not failures but a natural aspect of the pervasive execution. In this view, they are of an elective nature because they can be treated as foreseeable and prepared failures. They are planned in the way that the user, application code or the middleware can alter the connection state. A mobile device, for example, might want to operate disconnected for a while in order to reduce battery consumption and, at specific moments, reconnnect to update the state of the global execution. Such disconnection/reconnection procedures should be, whenever possible, transparent to the final user. Dealing with disconnection through fault tolerance techniques is an issue in our agenda of research.

3.2 ISAM Software Architecture

The ISAM software architecture has been designed to make the above scenario available and it provides support to both development and execution of applications. The ISAM software architecture, see figure 2, provides an integrated solution, from programming language support to execution environment, to build and execute large-scale pervasive applications. The upper layer corresponds to the ISAM abstractions for application designers, which includes a prototype programming language, named ISAMadapt [2]. The execution of ISAM applications is managed by the middleware, named EXEHDA.

The EXEHDA middleware is based on services which assist to three perspectives: (i) ISAMpe management [21], by providing services to control the physical medium where the processing will take place; (ii) support to application programming providing an API (Application Programming Interface) [2]; and (iii) support for execution of appli-
cations, by providing the services and abstractions necessary for the implementation of the follow-me semantics of pervasive applications. To accomplish that, EXEHDA provides support to adaptation at application level, and is itself adaptive. It is desirable that, when the system state changes, the middleware dynamically reallocate, reschedule and restructure the resources available for the application. The need for reallocating and rescheduling the system resources is determined by the execution context state [20].

4 Managing Adaptation

The adaptation model splits the responsibility of this process between the application and the middleware in two moments: (i) at development-time, when the adaptive behavior is defined and codified in multiversions (adapters) which exhibit different utilization profiles of computational elements, or when are defined adaptation policies which guide the middleware in its decision making; (ii) at runtime, when the adaptation execution over control of the middleware takes place. The adaptation policies are described in the policies.xml file that indicates options selected by programmer to tune the adaptation decisions of the middleware to the particular application peculiarities. In general, the policies are relative to adaptation commands of the programming language [2].

Figure 3: ISAM Collaborative Adaptation Model

Figure 3 illustrates the main components involved in the collaborative adaptation model. From the application’s viewpoint, the middleware is mostly responsible for (i) resources allocation; (ii) resources and services discovery; (iii) adaptation control; (iv) execution of adaptation and contextualized commands; (v) supplying context information.

We consider that the interference of the programmer is needed to make more optimized use of the available resources. Automatic adaptation is necessary, but it is not sufficient for assisting the user expectation. Because of that, the notions of context and adaptive behavior are at the heart of language constructions, which are based on four main abstractions [2]: context, adapters, adaptation commands and policies of adaptive behavior management. The language abstractions are implemented by call to middleware services or by defining service’s profiles. The middleware runs and manages the behavior of the application. The choice of the suitable code for the current situation is achieved by Adaptation and Context Recognition subsystem of the middleware, in runtime, considering the information stored in the file adapters.xml. This information is provided by the developer and they link the adapter code with the context element’s state.

5 Supporting the Follow-me Semantics

The middleware services are conceptually organized in subsystems: pervasive data and code access, uncoupled spatial and temporal communication, large-scale distribution, context recognition and adaptation [21]. The integration of the subsystems is shown in figure 4.

It is desirable that, when the systems’ state changes, the middleware dynamically reallocate, reschedule and restructure the (logical and physical) resources available for the application. In ISAM, this feature is known as non-
functional adaptation. The adaptation codified at language level or in the configuration of application components, as interesting context definition, are referred to as functional adaptation.

As a user physically moves (by carrying its current device - user mobility - or changing the device being used - terminal mobility), his currently running applications, in addition to the user’s virtual environment, need to be continuously available to him, following the user’s movement in the pervasive space. To provide this functionality, the middleware maintains information about user, applications, resources and services.

The Pervasive Access Subsystem is responsible for implementing the resource’s availability anywhere, at anytime. The ISAMdesktop tool, allows the user to manipulate its virtual environment, configuring user preferences or even launching applications installed there.

Applications are not installed in the traditional way, meaning that the executable code of applications is neither stored in nor managed by the AVU service. In fact, application installation consists only of copying the application’s launching descriptor to the users virtual environment. The executable code for the application is still provided on-demand by the BDA service by the time the application starts to execute on a given device. Application profiles (resource descriptors and shared data) are stored in the Application Virtual Environment (AVE), which disappears when the application finishes its execution.

The execution life-cycle of the application begins when the Gatekeeper Service receives the application launching request, which consists of the extended application launching descriptor, an XML document generated at development-time that groups several meta-data about the application, as the location independent reference to the application code, allowing the application to be launched from any EXEHDAnode in the ISAMpe.

After the launcher request is validated, the Gatekeeper delegates it to another service, the Executor Service, which effectively launches the application. At this time, the Executor instantiates and configures a class loader object for the application and begins the processing using the class defined in the <main> element of the launching descriptor. The installed class loader accesses the BDA Service in order to carry out the on-demand installation of application components (classes).

Once it has started, the application adds an identifier (ApplicationId) to it, which is unique in the scope of the ISAMpe. Moreover, an extended launching descriptor of the application is stored in the CIB Service (Cell Information Base). The CIB Service maintains the attributes related to the management of the ISAMpe, describing the resources that constitute the cell and its neighborhood, as well as attributes that describe the running applications and the resources allocated to them. Additionally, the CIB Service keeps track of information about the users registered to the cell. Hence, whenever a new node is incorporated to the application’s distributed execution, the local instances of the EXEHDA’s services running at that node could recover from the CIB Service all the application’s execution attributes necessary for their operation.

When the application finishes its processing, the middleware uses the information stored in the CIB Service to execute a clean-up step, reclaiming resources previously allocated to that application.

Figure 4: Middleware’s Subsystems
The assembling of the context state information, which guides many of the middleware operations and also the application’s adaptive behavior, is accomplished by the Context Recognition Subsystem, through the cooperative operation of the Monitor, Collector and Context Manager services [21]. The produced context state information feeds both, functional (that modifies the code being executed) and non-functional (related to scheduling and resource allocation) adaptation processes, which are managed by the AdaptEngine and Scheduler services respectively.

6 Final Considerations

The ongoing ISAM project proposes a software architecture that aims to explore the alternatives to build and run large-scale pervasive applications. From this perspective, the thesis argued by us is "the support infrastructure for pervasive computing at global scale can be constructed through integration of three computing fields: mobile computing, context-aware computing and grid computing". The proposed ISAM software architecture is large, in this paper we only talk about aspects directly involved with follow-me semantics. All adopted strategies to design the middleware’ services, such as minimum core, remote and adaptive instantiation, and uncoupling between access device and equipment that stores data and code, reflect that we intend to offer device independence to the mobile user.

We have also argued that there needs to be a way for the the application code and configuration profile to guide the decisions making process of the middleware that manages the whole environment. This strategy allows to reduce the level of inadequate decisions taken automatically by the middleware about the adaptation process.

Our current prototype has been developed considering two flavors of the Java Platform: J2SE, which is used in the desktop computers and the CDC profile of the J2ME Specification for Sharp Zaurus PDA’s.

References


