GRADEp*: Towards Pervasive Grid Executions

Cláudio F. R. Geyer[1] , Luciano C. da Silva[1] , Adenauer C. Yamin[34] , Iara Augustin[2] , Alberto E. S. Filho[1] , Maurício C. Moraes[1] , Rodrigo A. Real[3] , Gustavo C. Frainer[1] , Rafael P. Pires [2]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre, RS

[2]Centro de Tecnologia – Universidade Federal de Santa Maria (UFSM)
Santa Maria, RS

[3]Escola de Informática – Universidade Católica de Pelotas (UCPel)
Pelotas, RS

[4]Centro de Informática – Universidade Federal do Pelotas (UFPel)
Pelotas, RS

geyer, lucc, egon, mcmoraes, frainer @inf.ufrgs.br
adenauer, rreal @ ucpel.tche.br
august, rafaelpp @inf.ufsm.br

Abstract. *This paper provides an overview of our on-going work in the GRADEp middleware. GRADEp research is being developed by the GRADEp Working Group, sponsored by RNP, and aims at extending the traditional grid computing proposal with the notion of "pervasive grid executions" by incorporating aspects of mobility of devices, users and application components.*

Resumo. *Este artigo provê uma visão geral do trabalho em andamento no middleware GRADEp. As pesquisas no GRADEp são desenvolvidas pelo Grupo de Trabalho GRADEp, mantido pela RNP, tendo por objetivo a extensão da perpespectiva tradicional de computação em grade pela inclusão de aspectos de mobilidade de dispositivos, de usuário e de componentes das aplicações.*

1. Introduction

Pervasive computing promises ubiquitously support to users in accomplishing their tasks [Saha, 2003, Satyanarayanan,]. Hardware and networking infrastructure to make this come true are increasingly becoming a reality. Cell phones now include a processor and storage, PDA-like devices are widely available and becoming increasingly powerful. Moreover, wireless networking standards, such as IEEE 802.11 and Bluetooth, provide local connectivity for mobile nodes, while the Internet provides world-wide connectivity.

On the other hand, researches in GRID Computing are motivated by the need of coordinated access to shared resources in dynamic, geographically disperse, often multi-institutional, distributed environments. Foster [Foster, 2001] defines such set of individual and/or institutions involved in a common problem solving task, together with the rules that controls the relationships between the parts, a *Virtual Organization*. Applications developed with this kind of environment in mind also demand a high degree of both coordination and adaptation, since the access to shared resources in the system is inherently concurrent and the quality of such access may vary over time. Typical problems related to

*Sponsored by RNP under the GRADEp Work Group.

distributed systems like authentication, accounting, lookup and allocation of resources, as well as the potential heterogeneity of the system, became very expressive in such GRID environments.

GRADEp is an on-going work of the GRADEp Working Group, sponsored by Brazilian's RNP, which aims at extending the traditional grid computing proposal with the notion of "pervasive grid executions" by incorporating aspects of mobility to the grid scenario. Such view of pervasive executions rather enhance the challenges that need to be addressed by grid computing infrastructures. Currently, few applications, if any, are designed to effectively explore the whole potential of that distributed infrastructure.

In this paper we give an overview of the on-going work in GRADEp. The paper is structured as follows. Section 2 outlines the main elements of the GRADEp proposal. Next, in section 3, we introduce a perspective of further extending GRADEp features by borrowing new elements being maturated in a more experimental branch of our research – the ISAM Project. Finally, in section 4 we present our concluding remarks, shortly describing our next steps in GRADEp development.

2. The GRADEp Proposal

GRADEp's main goal is to develop a middleware over which typical applications from the Grid Computing scenario would be built under the perspective of a *pervasive execution*. Towards this goal, GRADEp proposes a middleware and defines building policies for assembling the "pervasive environment" from currently existing computing resources.

At the physical level, the GRADEp proposal assumes an existing wired network backbone, which is extended by leaf nodes, those may be both wired and wireless devices. A core concern at this level is that leaf nodes would exhibit a mobile behavior, meaning the access point to the wired backbone may change over time. Moreover, leaf nodes need not to be connected full-time to the wired backbone. Although such behavior is more often observed with wireless devices, which disable their wireless link in order to reduce battery consumption, we assume users with wired devices also would decide to operate disconnected since it would lead to a more general and flexible solution.

At the logical level, the distributed infrastructure that builds up the GRADEp pervasive environment is structured in a peer-to-peer (P2P) network of computing cells. Each cell in the system is identified by a system-wide unique cell identifier (the cell's name). The notion of computing cell resembles that of Virtual Organizations[Foster, 2001] as each cell has autonomy to establish access policies to its own resources. Moreover, cells would be seen as institution boundaries, though an institution would decide to split its resources in several cells too and that should not compromise the system.

P2P cell networking is an effective way of conferring scalability to the system. Since it is an inherently dynamic structuring, it would be easy to extend the system by aggregating new cells. Further, inserting a new cell should not be an expensive operation, because, by construction, each cell in system need not to know about every other cell in the system, but just a subset of those cells. The set of know cells of a given cell in a given time defines its neighborhood. A cell's neighborhood may be tuned during the life-time of the system by considering affinity aspects (e.g. geographical proximity, economical cost, quality of resources or, more generally, quality of the services provided by cells) to other cells in the system.

Internally, each cell is build up of a cell base and a set hosts (wired and wireless). The cell base is a virtual, eventually distributed, entity inside the cell responsible for running the cellular instances of the middleware services. It acts as a reference point

for both the remaining nodes in the cell and the other cells in the system. Thus, it allows nodes inside a cell to exhibit a more dynamic behavior and it is a key element in supporting disconnected operation and mobility of leaf nodes.

Every host is assigned a system-wide unique host identifier (HostId) which is built relative to the identifier of the *home-cell* of the device (i.e., the cell in which the device was originally registered). Additionally, each host has a certificate, signed by the home-cell's manager, which is used to authenticate that host's issued operations. Being able to identify resources and, more specifically, mobile hosts in the system is a key concern in order to securely support guest devices in a cell.

Similarly to hosts, users have certificates signed by their home-cell managers. The user's certificates is used to authenticate user's operations and for checking user's access permissions to resources. An important concern regarding to GRADEp users is that they are also mobile, meaning the system needs to provide them access to their data and applications in different devices and, eventually, in different cells.

2.1. GRADEp Core Concepts and Components

The computing model adopted in GRADEp is based on the OX abstraction defined by Yamin [Yamin, 2004]. An OX consists of an object (autonomous entity which encapsulates code and data) managed middleware, which allows one to bind meta-attributes at run-time to that object. Further, an OX typically has its own execution thread and may, optionally, exhibit a mobile behaviour (i.e., it may migrate between devices).

GRADEp applications may be seen, therefore, as collections of OX spread over the distributed infrastructure. Typically, the application starts by the creation of one OX and grows over the resources in the environment through two mechanisms: (i) remote instantiation of other OX and (ii) OX migration between nodes.

Though the OX concept resembles that of mobile agents, in the GRADEp proposal, differently from pure agent systems, there is clearly the notion a distributed identity of a GRADEp application. Such identity is represented in the form of an system-wide unique `ApplicationId`. Hence, the middleware is aware of the whole application life-cycle, being able to relate such application to the user who launched it (i.e., the application's owner and its permissions in the system), as long as to the resources (hosts) currently allocated for the application. Such information enables the implementation of single-sign-on authentication schemes.It would also be used as base for an accounting mechanism for the pervasive grid.

In order to handle the high degree of heterogeneity in the system, the middleware is structured in a minimum core. Pluggable services are loaded on-demand to extend this base core. The services loading mechanism is guided by a per-host middleware profile, which binds an device-optimized implementation to each abstract service interface as long as defines the loading strategy that must be used for a given service (at bootstrap time or on-demand). Such profiles are described as XML documents. Three components form the minimum GRADEp core:

- the **ProfileManager**, which manages the middleware profile data;
- the **ServicesLoader**, which is responsible by the on-demand download and installation of services executable code from the pervasive code repository; and
- the **ServicesManager**, which controls services activation and life-cycle bases on the information defined in the middleware profile.

2.2. GRADEp Services

Among the services being developed in GRADEp, we highlight:

Cell Information Base – implements the cells catalog of resources, users and currently running applications;

Executor – provides operations for controlling application life-cycle, executing remote action in behalf of an application, remote OX instantiation and migration;

Scheduler – guides the target host selection for remote OX instantiation and migration, considering in the decision taking dynamic aspects of the system;

Resource Broker – enforces resource access control and allocation policies;

OXManager – supports the OX meta-attributes in the pervasive environment;

WORB – object-request broker with support for disconnected operation semantics.

CodeRepository – the pervasive code repository used by on-demand code installation;

User Virtual Environment – pervasive access to user's private data and preferences;

Collector – implements a monitoring scheme based on parameterizable sensors.

3. GRADEp-ISAM Integration: joining research efforts

The ISAM Project (`http://www.inf.ufrgs.br/~isam`), another branch in our Pervasive Computing Systems research, aims at providing an integrated solution from programing language to execution environment for building general purpose pervasive applications. Under the perspective of ISAM, the context conditions are monitored in a proactive way and the execution support must allows that both, the ISAM middleware itself and the application, use those informations in the management of its functional and non-functional aspects.

We envisage two possible ways GRADEp would take benefit of ISAM research: (i) by borrowing new services targeted to the pervasive computing scenario and (ii)through the definition of new frameworks that would make easier for one to build distributed adaptive applications. Regarding to frameworks, we highlight a master-worker with adaptive job-size and a dependency-graph-oriented frameworks under development in ISAM. With respect to services, two promising works are the Perdis and Dimi services. Those services are briefly described in the following sections.

3.1. Perdis: a pervasive discovery service

The *PerDiS* (*Pervasive Discovery Service*)[Schaeffer Filho et al., 2004] allows resources to register themselves in directory services, and users to query these directories. The pervasive computing imposes to the design of a discovery service new requirements and challenges. Among them, we can highlight *adaptive behavior in response to context changes*, *expressiveness for resource description*, *large-scale resource discovery support*, *timely aspects related to resource access* and *self-healing strategies for consistency maintenance*. Figure 1 outlines the components of the *PerDiS* architecture, which aims to address this set of requirements. Next, the main aspects related to each component are discussed.

Resource component (RC) – is resposible for *(a)* managing the descriptive properties related to a resource specified by a XML document and *(b)* periodically indicating the availability of a resource.

User component (UC) – allows the user to perform the discovery process. It provides a query building mechanism. One of the main shortcomings of current discovery strategies is the lack of expressiveness for representing resources and query specifications [Chakraborty and Chen, 2000]. *PerDiS* proposes a language which is expressive for resource description/query specification, ease for extension and possible to interoperate with other strategies. In order to address these issues we are using a XML based approach for building resource descriptors and search queries.

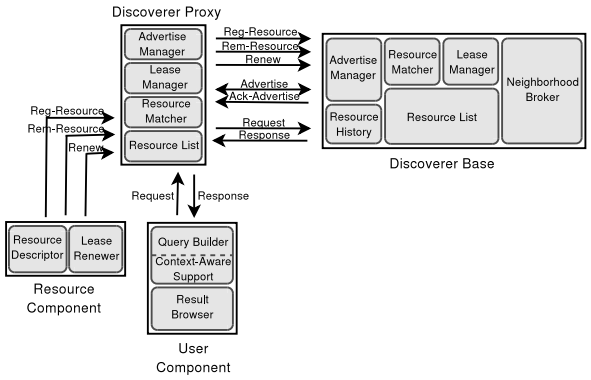

Figure 1: Components of the discovery architecture

Discoverer base (DB) and Discoverer proxy (DP) – act as a catalog of resources in an execution cell and in a specific node respectively. A DP instance manages the local resources made available in a specific ISAMpe node (the host itself, a database server, a file server, etc) and communicates with the DB instance to catalogue these resources at execution cell level. Resources and users communicate with a `proxy` instance of the discovery service. A `discoverer base` has some additional modules not found in a `discoverer proxy` which provide *(a)* a history along with each DB, that tries to identify criteria and results previously processed for a given user and *(b)* support for peer-to-peer communication among the cells that make up the ISAM pervasive environment to enable large-scale resource discovery.

3.2. Dimi: a service for application-level multi-cast in pervasive environments

Application-level multicast (ALM) dissemination as opposed to IP Multicast, has becoming an alternative for group comunication, offering accelerated initialization, simplified configuration and better access control, to the cost of additional network load [Pendarakis et al., 2001]. ALM do not demand heavy control over the infrastructure of the network (routers, protocols and software). The communication topology used in ALM is an overlay network made by endhosts, that exchange data using unicast transmissions.

DIMI (Information Multicast Disseminator) [Moraes, 2004] is an ALM service designed to outfit necessary features ISAM's architecture. DIMI sends data from one origin to an arbitrary number of destinations. DIMI's overlay network adapt itself to the underlying phisical network, accordingly to user parametrization and to the state of service members at running time.

DIMI proposes an algorithm that seeks to improve scalability to the service, by potencially reliefing the overlay network's root node from handling multiple concurrent requests. DIMI supports message priorization and uses dissemination filters. Also, DIMI support planned disconection and user mobility in the ISAMpe. To comply with the ISAM architecture, DIMI dissemination model has the following objectives: (i)escalability: DIMI has to avoid bottleneck caused by the concurrent entry of new consumers on the system; (ii) dynamic adaptation: DIMI has to adapt on the fly to the dinamic caracteristics of the execution environment, redirecting data and creating new connections according the state of the service's members; (iii) suport to planned disconection: DIMI has to permit that connectivity restricted computational devices be part of the dissemination; (iv) suport to user mobility on the ISAMpe (v) filter utilization: filters are processing done on the data disseminated to turn its size small and, consequently, save network resources and optimize dissemination performance.

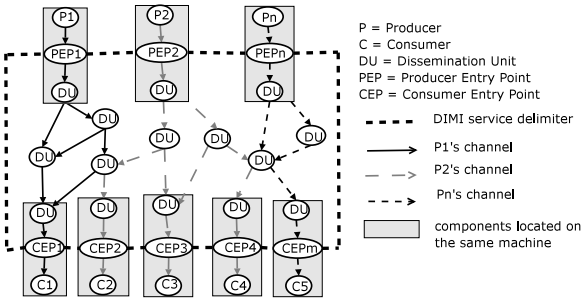

Figure 2: DIMI architecture and its main components.

DIMI architecture is composed by chained DUs (Dissemination Units) responsible for data transportation from the producer to all the consumers (figure 2). Users use PEPs (Producer Entry Points) and CEPs (Consumer Entry Points) to, respectively, send data to a channel and receive data from it. The fundamental behaviour of a DU is receive data from a producer and send it to a number of consumers. DIMI can handle many simultaneous channels.

4. Concluding Remarks

GRADEp is currently an on-going work. An stable version of the GRADEp middleware is scheduled to be released early in the second quarter of 2005. Such release will be presented to the community in the SBRC2005 event.

References

Chakraborty, D. and Chen, H. (2000). Service discovery in the future for mobile commerce. *ACM Crossroads*, 7(2):18–24.

Foster, I. (2001). The anatomy of the Grid: Enabling scalable virtual organizations. *Lecture Notes in Computer Science*, 2150.

Moraes, M. (2004). Dimi: Um Disseminador Multicast de Informações para a Arquitetura ISAM. Masters thesis, PPGC, Instituto de Informática, UFRGS. (on-going work).

Pendarakis, D., Shi, S., Verma, D., and Waldvogel, M. (2001). ALMI: An application level multicast infrastructure. In *Proceedings of the 3rd USNIX Symposium on Internet Technologies and Systems (USITS '01)*, pages 49–60, San Francisco, CA, USA.

Saha, D. Mukherjee, A. (2003). Pervasive computing: a paradigm for the 21st century. *UNKNOWN*, 36(3):25–31.

Satyanarayanan, M. Pervasive computing: vision and challenges. In *UNKNOWN*, volume 8, pages 10–17.

Schaeffer Filho, A. E., da Silva, L. C., Yamin, A. C., Augustin, I., and Geyer, C. F. R. (2004). PerDiS: Um modelo para descoberta de recursos na arquitetura ISAM. In *VI Workshop de Comunicação Sem Fio e Computação Móvel*, pages 98–107, Fortaleza, Brasil.

Yamin, A. (2004). *Arquitetura para um Ambiente de Grade Computacional Direcionado às Aplicações Distribuídas, Móveis e Conscientes do Contexto da Computação Pervasiva*. Phd thesis, PPGC, Instituto de Informática, UFRGS.