Resource Discovery Support for Time-Critical Adaptive Applications

Carlo Bertolli and Daniele Buono
Dept. of Computer Science
University of Pisa, Italy
{bertolli,d.buono}@di.unipi.it

Gabriele Mencagli and Massimo Torquati
Dept. of Computer Science
University of Pisa, Italy
{mencagli,torquati}@di.unipi.it

Marco Vanneschi
Dept. of Computer Science
University of Pisa, Italy
vannesch@di.unipi.it

Matteo Mordacchini and Franco Maria Nardini
ISTI-CNR, Pisa, Italy
{m.mordacchini,f.nardini}@isti.cnr.it

ABSTRACT

Several complex and time-critical applications require the existence of novel distributed and dynamical platforms composed of a variety of fixed and mobile processing nodes and networks. Notable examples of such applications are crisis and emergency management and natural phenomenon prediction. In this scenario we need the development of applications able to adapt their behavior according to the dynamical platform conditions, such as the presence of specific classes of computing resources and the actual network availability. For these reasons such adaptive applications need to interact with a fast and reliable resource discovery support, which ensures required response times by means of an high-degree of reconfigurability and selectivity. In this paper we present an integrated approach between our programming model for distributed adaptive time-critical computations and a suitable resource discovery support.

Categories and Subject Descriptors

C.2.4 [**Computer Communication Networks**]: Distributed Systems—*Distributed applications.*

General Terms

Design, Experimentation, Performance.

Keywords

Autonomic Computing, High-Performance Computing, Pervasive Grid, Resource Discovery.

1. INTRODUCTION

Over the past few years Grid Computing has emerged as the dominant paradigm for wide-area distributed computing. However, recent technological advances in computing and in particular in communication technologies and the proliferation of pervasive systems are leading to the emergence of a new generation of applications that use context information as an integral part of the application management, to optimize and dynamically adapt the cooperation of the application software components. Notable examples of such applications are emergency management, natural phenomenon prediction, homeland security and i-mobility.

The abstract platform on which this kind of applications are executed is composed of a variety of fixed and mobile nodes, interconnected through multiple wireless and wired network technologies. Such heterogeneous resources are characterized by different and highly variable levels of availability (e.g. they can dynamically change their location and presence) and by different capabilities (e.g. memory and processing capacity and their performance). In this case the term *context* represents the actual conditions of both the surrounding environment and the computing and communication platform. In this scenario, fixed and mobile nodes (e.g. PDA, wearable devices, smart-phones) and networks must be able to capillary provide users with the necessary services in different processing and connectivity conditions.

The Pervasive Grid paradigm [8] implies the development, deployment, execution and management of applications that, in general, are dynamical in nature. Dynamicity concerns the *resource discovery* process (both for processing nodes and software components) and the deployment and composition of the most suitable versions of software components. The main objective is to satisfy the needed Quality of Service (QoS): i.e. a set of metrics reflecting the experienced behavior of an application such as its memory occupation, battery consumption, the estimated performance as well as the user degree of satisfaction, e.g. the precision of computed results. The specification and requirements of this QoS itself are varying dynamically, according to the information produced by sensors and services, as well as according to the monitored state and performance of networks and nodes.

It is clear that Pervasive Grid applications must be characterized by an adaptive behavior in order to efficiently deal with highly variable QoS requirements and with the dynamicity of the surrounding execution platform. Adaptivity means that the application must be able to identify specific conditions which require the execution of proper application reconfiguration activities: e.g. some non-functional application parameters can be modified, such as its memory occupation, its performance and its battery consumption. In a very heterogeneous environment, such as a Pervasive Grid, the relevant differences between the available processing resources can lead to more general application reconfigurations: we can provide a set of different versions of the same software component, each one suitable for specific context situations (e.g. mapping onto new available and discovered computing resources or when some network conditions occur). In this case a crucial issue is how to efficiently discover the presence of new available processing nodes and how to efficiently find and deploy the most-suitable version of a specific application component on these discovered resources.

In time-critical scenarios, such as emergency management, the resource discovery phase plays a central role. Finding computing resources and software components in a fast, predictable and reliable way is an unavoidable requirement in order to exploit an acceptable overhead for the application reconfiguration activities. In this paper we present an integrated approach between resource discovery methodologies and a programming model for high-performance adaptive application for Pervasive Grids. Our focus is mainly based on the development of critical compute-intensive applications which require efficient and configurable run-time mechanisms for exploiting application adaptivity.

This paper is organized as follows. In Section 2 some related works concerning adaptive systems and resource discovery methodologies are introduced. In Section 3 a specific time-critical Pervasive Grid application is presented: a system for flood emergency management. In Section 4 a programming model for high-performance adaptive applications is described focusing on the interaction with the resource discovery phase. In Section 5 the resource discovery runtime support is introduced and, finally, in Section 6 some first experiments are described which have been useful for evaluating our approach.

2. RELATED WORK

Adaptivity has been introduced for mobile pervasive applications by exploiting the concept of context [2]. For instance, in Odyssey [7] an operating system is responsible for monitoring the resource utilization. Significant changes in the resource status are notified to applications, which adapt their execution quality to the new execution environment.

High-performance for pervasive applications is introduced in [6]. Anyway, in this work high-performance computations are executed on centralized servers, while mobile nodes are only demanded to result collection. Several research works are focused on adaptivity for parallel programs in grid environments. In [1] it is shown how a hierarchical management can be defined in the case component-based applications are developed according to well-known parallelism paradigms.

Many resource discovery solutions have been investigated for grid platforms and for pervasive heterogeneous scenarios. Interesting approaches are related to the distribution and sharing of resource information across federated registries (i.e. distributed approaches). Typically these systems are built on Peer-to-Peer technologies as in Schmidt and Parashar [9]: they present a P2P indexing system and associated P2P storage that supports large-scale, decentralized and real-time search capabilities. In Li et al. [5] the authors use a Chord P2P protocol as overlay, consisting of Service Peers (SP). Each SP is mapped onto several Logical Machines (different machines corresponding to the same hardware). Each Logical Machine maintains the necessary interfaces to map Web Services onto the P2P network.

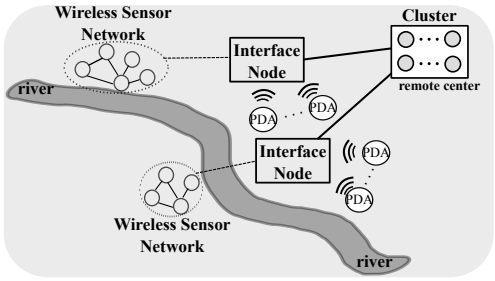

Figure 1: Logical scheme of a Pervasive Grid for flood emergencies.

3. PERVASIVE GRID SCENARIO

We consider a fluvial flood prevention system whose Pervasive Grid infrastructure is depicted in Figure 1. It exploits a large number of sensors spread across the river, monitoring the water level and other environmental parameters. During "normal" situation those parameters are periodically acquired and used by a forecasting model, which is a time-critical compute-intensive processing. During the emergency itself, shot-term predictions for a limited area of the river are also required. In this scenario a human operator could ask the execution of the model for a specific river area. In "normal" situations, the forecasting processing is executed on a remote cluster architecture, so we imagine a network infrastructure made of dedicated links (wireless and wired) between the cluster and mobile devices geographically spread along the river basin. In critical situations the network connectivity with the central servers could be down or unreliable. To manage this potential crisis in real-time, we can execute the forecasting model on a set of decentralized resources whose interconnection is currently reliable. Thus, the logical interconnection structure of the application has to be mapped onto the surrounding physical infrastructure in the most efficient way, to make it possible the fastest and most reliable prediction with the currently available nodes. Therefore the application must be designed with several levels of adaptivity, to be useful both on normal and critical situations, and on a broad and dynamical range of devices.

Just limiting to the above scenario, it is clear that there is a complex problem in dynamical allocation of software components to processing and communication resources. Some resources may have specific constraints in terms of storage, processing power and power consumption: the same version of the software components may be not suitable for them, or even may be impossible to run on them. Therefore the application needs an updated view of the execution environment, in order to choose the best component allocation over the available nodes. Historically, especially in grid environ-

ments, this part of the application has been exploited by a resource discovery support or service.

Though many research efforts have been performed, classical grid approaches are not suitable in a very dynamical and heterogeneous environments as Pervasive Grids. We started from a previous work on resource discovery algorithms [4], to make them suitable for time-critical applications characterized by strict real-time constraints. As a consequence, a key-issue is the development of resource discovery mechanisms featuring predictable and settable response times. For instance, in emergency management systems we are interested in quick responses by the resource discovery service, even at the cost of loosing some levels of accuracy. The trade off between accuracy and fast responsiveness, in finding resources within a very heterogeneous and dynamical environment, is not an easy task to solve.

4. THE ASSISTANT PROGRAMMING MODEL

Our evaluation framework for the development of high-performance adaptive applications is the ASSISTANT programming model [3]. In ASSISTANT an adaptive application is described as a graph of distributed cooperating software components, each one exploiting a possible high-performance computation expressed by using well-known parallelism schemes (i.e. the Structured Parallel Programming approach as in our previous work [11]). An ASSISTANT component is also a complete high-performance, adaptive and context-aware unit, so it also exploits its own management and context logics in order to express an adaptive behavior in response to environmental and platform changes.

Referring to the emergency management application presented in Section 3, an adaptive component is responsible for executing the flood forecasting model characterized by specific performance constraints. As stated before, this part of the application could be executed on very different resources, depending on the current conditions of the Pervasive Grid platform. For obtaining the required performance, in the model solving phase, we have to use different sequential algorithms and different parallelization techniques, depending on the hardware on which it is currently executed. This means that the programmer must specify different versions of the single component, that will perform better on the various available and discovered computing resources.

Obviously, the alternative versions of the same component have different but compatible semantics: i.e. they can exploit different algorithms, different parallelization schemes or optimizations, but preserving the component interfaces in such a way that the selection of a different version does not modify the behavior of the global application. Of course only a specific version of a component is selected for the execution, according to the actual context conditions and the adaptation strategy performed by its management logic.

As depicted in Figure 2 ASSISTANT features proper programming constructs to define, for each application component, its adaptive behavior by expressing:

- **Computations**: this represents all the possible versions performed by the component. One of them can be selected for the execution according to the current context conditions (e.g. QoS measurements and different availability of computing and network resources);
- **Manager**: it is responsible for executing the adaptation strategies performed to adapt the component behavior in response to specific events;
- **Context**: this includes all the aspects which link the module behavior with the surrounding context. Sensors, monitoring services and in general context interfaces can generate events that are received by the component. This information can be utilized to trigger specific reconfigurations according to the exploited management strategy.

In many time-critical application, like emergency management, we need to adapt the application behavior in order to respect and maintain a user-defined QoS level, e.g. we could require a minimum throughput for the execution of a parallel computation like the flood forecasting model. For this reason we can exploit proper *performance models* of well-known parallelization schemes (e.g. a task-farm or a data-parallel schemes) in such a way as to have a reasonable expectation of the performance in function of specific parameters (e.g. the execution time of a sequential code and the communication time between different processes).

In response to platform changes, we need to modify the allocation of the application components over the physical nodes, i.e. changing the parallelism degree or, if it is necessary, to completely deploy a different version of the same component on a specific set of discovered computing resources. In this phase the Manager cooperates with a resource discovery service to obtain a list of nodes that could be used for the next configuration. From our point of view this interaction is not only limited to the deployment phase. Resource information likes network latencies, energy levels, number of available nodes can be continuously monitored and acquired by the Manager of each component, in order to dynamically instantiate the performance model parameters in such a way as to maintain an updated view of the surrounding execution platform. For these reasons the interaction with a distributed resource discovery service is a key-point in our approach, both for component deployment reasons but also for updating the adaptation strategies of each application component.

5. RESOURCE DISCOVERY SERVICE

In our approach the resource discovery service must be part of the ASSISTANT run-time support running on each device of the Pervasive Grid. It uses a distributed repository containing all the information regarding the available resources. Figure 2 shows how the resource discovery service is internally structured. It consists of two main parts:

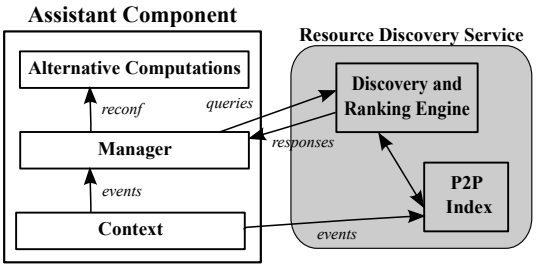

Figure 2: Resource discovery interaction with an ASSISTANT adaptive component.

i) a Discovery and Ranking Engine (DRE) that acts as an "intelligent" interface to Managers, and ii) a P2P meta-data index storing all the resource information.

The DRE implements the provided interfaces for Managers to use the resource discovery service, which works in a flexible, configurable and fast way. Efficiency is an important feature due to the importance of information provisioning during normal or emergency situations. The DRE engine has been implemented to manage an XML description of the resources by indexing their features. In this approach resource is a general term which corresponds to hardware computing resources but also software application components. For this reason, each resource has two main features: (i) functional, and (ii) non-functional parameters. Functional features are basically for software resources, i.e. the name of the service and a short textual description of the service features. Non-functional ones consist in QoS parameters or other properties related to a computing resource. Functional and non-functional parameters are stated in a resource description shared through the distributed resource repository. Descriptions use the XML language to enable a portable representation of resources.

The DRE [4] has been developed using Information Retrieval techniques to enable efficiency over big indexes. The Vector Space Model consists in representing an object as a *vector* in $\mathbb{R}^n$, where each dimension corresponds to a separate term. If a term occurs in the object, its value in the corresponding vector entry is non-zero. Resource descriptions are modeled as vectors in $\mathbb{R}^n$, where n is the number of possible terms. For simplicity we consider only unit vectors. The normalization is done in a way that all the vector coordinates will range between 0 and $1/n$.

DRE uses a well-known Information Retrieval *similarity function* called *cosine metric*. Given a query q and a set of resources S, the equation in (1) measures the similarity degree between the query and the resource. By operating in this way the DRE is able both to find resources on the Pervasive Grid by means of the P2P index, and to rank the obtained resources. Managers asking for a particular resource will be provided with a list of resources similar to what they have required.

$$sim\,(q, s \in S) = \sum_{i=1}^{n} q_i s_i \qquad (1)$$

The P2P index relies on a Voronoi-based overlay network [5]. In this network, objects are mapped using their features as coordinates in the space. The space is divided in regions using the Voronoi tessellation technique. Each peer is assigned to a different region and is responsible for the points falling into it. Each point of the space is assigned to the closest peer according to some notion of distance. The set of points that are assigned to a peer p_i constitutes the cell of p_i. If two cells are adjacent, their related peers are linked. The resulting network is also called the Delaunay triangulation of the objects. The system does not have any a-priori knowledge of the topology like for Pervasive Grid platforms. Thus, it can be adapted to different dynamical situations and scenarios. Since objects with similar attributes are close in the network, data locality is preserved and can be used to find all/some objects that are located into a region, also called Area of Interest (AoI). Moreover, locality constitutes an advantage for fault tolerance, e.g. with respect to the "classical" Distributed-Hash-Table based P2P approaches, since object insertion/removal perturbs only the neighborhood of the object. As a further advantage, the routing mechanism of Voronoi networks makes it possible to reach the requested area even if the network contains some "wholes", i.e. regions with disconnected or malfunctioning peers. As a consequence we think that a Voronoi approach is a suitable solution for Pervasive Grid platforms.

6. TEST RESULTS

In the fluvial flood management application described in Section 3, the most critical part is the flood forecasting model, that involves the numerical resolution of a number of partial differential equations. In many approaches (like MIKE 21 [10]) this phase requires the resolution of a large sequence of tridiagonal linear systems whose size determines the forecast precision. In this paper we limit to the tridiagonal system resolution, which is the most compute-intensive processing of the overall forecasting model.

In this experiment our objective is to respect a QoS constraint for the service time of the tridiagonal solver, in order to simulate a real case of emergency where an human operator requires a short-term forecast for a limited area. Existing parallel techniques for solving tridiagonal systems make it possible the achievement of reasonable service times in a scalable manner. In this example we select the Cyclic Reduction algorithm providing different parallelization schemes used according to different resource availability. We consider the presence of different classes of computing resources:

- **Low-end interface node**: a multicore IBM Cell processor, with six cores and 256 MB of main memory, which interconnects mobile devices with a centralized architecture;
- **High-end Mobile nodes**: in our tests we have used a set of PDAs with 300 Mhz ARM processor and 64 MB of main memory;
- **Low-end Mobile nodes**: a set of PDAs equipped with a 300Mhz ARM processor and a limited amount of main memory (i.e. 16MB).

For brevity we only consider the scenario in which discovered mobile resources are utilized. During the development of the alternative versions of the forecasting component, we are interested in the memory occupation which must be different depending on the resources selected for the execution. For this reason we have developed two different parallel task-farm versions: the first one exploits an higher throughput but with a larger memory occupation; the second one is characterized by a lower throughput but with a more limited memory utilization.

On the interface node we can solve up to six systems simultaneously, so we must reserve space in main memory for the resolution of six tridiagonal systems; unfortunately the faster version does not fit in the main memory, so we have to use the slower one which requires a less memory utilization. Memory occupation is also important for the mobile nodes. In this case the parallelism degree of the task-farm scheme is the number of utilized mobile nodes, each one solving the systems one at a time only. Therefore, PDAs with 64 MB of main memory can execute the faster version, while for the mobile nodes equipped with only 16 MB we have to use the slower version. Figure 3 shows how important

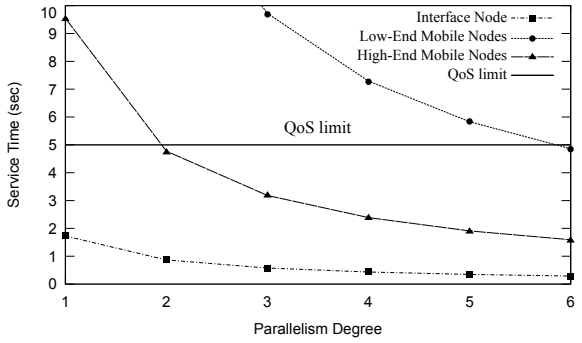

Figure 3: Service time of the Tridiagonal Solver mapped onto different platforms, compared with the required QoS level.

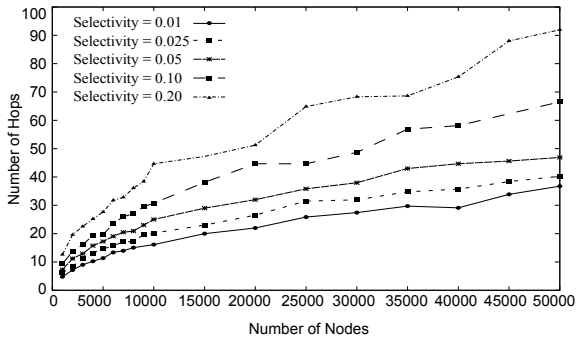

Figure 4: Number of hops for the P2P index to obtain a response with several levels of selectivity.

is a good knowledge of the current execution environment: having more high-end PDAs, or an unloaded interface node can change the service time significantly, and allows us to remain under the required QoS level (i.e. in the example a specific maximum threshold for the average service time).

In this experiment we are interested in dynamically discover computing resources. In Figure 4 is reported our preliminary study on the P2P index performance, presenting the number of hops required to discover nodes on the overlay network. The selectivity level represents the amount of P2P network covered by the query. With an higher selectivity we can obtain better results, but at the cost of more hops on the network and thus a slower response time. This means the Manager can require faster response times from the RD service at the cost of lower result quality by changing the selectivity level. Another important point is that the number of hops is limited (10-30 with ten thousand nodes); in our experiments using a 802.11n Wi-Fi network we have a point-to-point transfer delay of 0.06 seconds for 1 MB of data. Indeed the P2P traffic for a query is much lesser than 1 MB of data, so we can estimate a response time of less than 1 second for 15-30 hops, and thus a fast enough response time for our emergency scenario.

7. CONCLUSIONS AND FUTURE WORKS

In this paper we have shown the importance of predictable and limited response times for each part of the system, especially the resource discovery service for adaptive and time-critical applications. From our preliminary results the Voronoi-based P2P index exploits all the required features: high-performance and, more important, a reconfigurable selectivity level to adjust the query response times. We are working on a better integration of this service in our ASSISTANT programming model, in particular we plan to use the resource discovery service not only for finding hardware resources, on which the application components could be deployed, but also the software components themselves that can be found on a distributed remote software repository.

8. ACKNOWLEDGMENTS

This research has been funded by the IST FP7 European Project S-Cube Grant Agreement no. 215483.

9. REFERENCES

[1] Danelutto, M., and Zoppi, G. Behavioural skeletons meeting services. In *ICCS '08: Proceedings of the 8th international conference on Computational Science, Part I* (Berlin, Heidelberg, 2008), Springer-Verlag, pp. 146–153.

[2] Dey, A. K. Understanding and using context. *Personal Ubiquitous Comput. 5*, 1 (2001), 4–7.

[3] Fantacci, R., Vanneschi, M., Bertolli, C., Mencagli, G., and Tarchi, D. Next generation grids and wireless communication networks: towards a novel integrated approach. *Wirel. Commun. Mob. Comput. 9*, 4 (2009), 445–467.

[4] Laforenza, D., Nardini, F. M., and Silvestri, F. Collaborative ranking of grid-enabled workflow service providers. In *HPDC '08: Proceedings of the 17th international symposium on High Performance distributed computing* (New York, NY, USA, 2008), ACM, pp. 227–228.

[5] Li, Y., Zou, F., Wu, Z., and Ma, F. Pwsd: A scalable web service discovery architecture based on peer-to-peer overlay network. In *APWeb* (2004), pp. 291–300.

[6] Lillethun, D. J., Hilley, D., Horrigan, S., and Ramachandran, U. Mb++: An integrated architecture for pervasive computing and high-performance computing. In *RTCSA '07: Proceedings of the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 241–248.

[7] Noble, B. System support for mobile, adaptive applications. *Personal Communications, IEEE 7*, 1 (Feb 2000), 44–49.

[8] Priol, T., and Vanneschi, M. *From Grids To Service and Pervasive Computing*. Springer Publishing Company, Incorporated, 2008.

[9] Schmidt, C., and Parashar, M. A peer-to-peer approach to web service discovery, 2003.

[10] Thompson, J., Sorensen, H. R., Gavin, H., and Refsgaard, A. Application of the coupled mike she/mike 11 modelling system to a lowland wet grassland in southeast england. *J. of Hydrology 293*, 1-4 (2004), 151–179.

[11] Vanneschi, M. The programming model of assist, an environment for parallel and distributed portable applications. *Parallel Comput. 28*, 12 (2002), 1709–1732.