Rendiconti
Accademia Nazionale delle Scienze detta dei XL
Memorie di Scienze Fisiche e Naturali
122° (2004), Vol. XXVIII, pp. 267-284

MARCO VANNESCHI *

Grid.it: National Research Project on Next Generation Grid Platforms

Abstract – Grid.it is a FIRB Strategic Project in the area of Enabling Technologies for Information Society, coordinated by the National Research Council, granted by the Italian Ministry for Education and Research for three years (November 2002 - November 2005). This project, having a strong interdisciplinary character, is aimed at defining, implementing and applying innovative solutions for network computing enabling platforms, oriented towards scalable Virtual Organizations and based on the Grid Computing paradigm. The research topics of Grid.it span from high performance networks, innovative middleware services, high-performance programming environments and application testbeds.

This paper is organized as follows:

- Part 1: introduction to Grid Computing technology and concepts,
- Part 2: overview of Grid.it project,
- Part 3: description of the Grid.it approach to programming environments for developing Grid-aware applications.

1. Basic concepts on Grid Computing technology

Grid Computing is the area that studies computing and data management infrastructures providing an abstraction for resource sharing and collaboration across multiple administrative domains [2, 3]. Grid platforms are characterized by the ability to dynamically link together resources as an ensemble to support the execution of large-scale, resource-intensive and distributed applications for a global society in business, government, research and science [4, 5, 6].

This research area originated a large interest growth, not only in USA but also in Europe, as proved by different research projects, e.g. the e-science project in UK

* Dipartimento di Informatica, Università di Pisa, Via F. Buonarroti, 2 - 56100 Pisa. E-mail: vannesch@di.unipi.it

and the European Community actions in the VI Research Framework. These new technologies are aimed at drastically improving the impact of ICT in the economics, commerce and industry, as well as to play an important role in the evolution of computational sciences.

In the last years there have been large improvements in computer systems, in distributed platforms, in WEB computing (client/server and peer-to-peer), in application service and storage service products, in Enterprise Computing Systems (CORBA, Java Beans, Dcom, etc.) and in the programming and development technologies (objects, components). Despite the results achieved in all these fields, the ability to actually *integrate* these technologies to meet the requirements of multidisciplinary applications (either compute or data intensive) did not grow consequently: these requirements concern performance, security, reliability and quality of service, according to an Open Software / Open Standard approach. In other terms, new ITC platforms are required to support, efficiently and in a secure and controlled way, resource sharing and integration in a scalable Virtual Organization (VO) context. VOs (sets of individuals or organizations that need to share resources to solve a given complex problem) have a strong dynamic structure, form, lasting and composition: this poses new strategic challenges to ITC technology. The paradigm and the infrastructure having scalable VO as the main target is usually called *Grid Computing.*

From the high-performance architectures point of view, the trends in ICT platforms are clearly focused in the direction of integrating computing resources into *large-scale platforms* [2]: the high-performance computing resources to be integrated include parallel machines such as SMP shared memory systems, previous-generation supercomputers, homogeneous and heterogeneous Beowulf clusters of PCs and/or workstations, as well as Beowulf clusters of parallel (SMP) nodes. The interconnection of such resources, according to a client-server or a peer-to-peer approach, is not limited to state-of-the art distributed systems at the Internet level: high-speed communication technologies are emerging, not only at the cluster and LAN level, but also at the Intranet and MAN level, e.g. very high-bandwidth optical networks. In this context, the Grid Computing paradigm is emerging as an outstanding approach to realize new large-scale platforms to support high-performance applications efficiently and in a secure and controlled way.

An outstanding goal of Grid technology is to allow the users/applications to achieve the desired level of *Quality of Service* – in terms of *performance*, *fault tolerance*, *security* –, and to guarantee that such QoS level is preserved in spite of the *heterogeneou*s and *dynamic* nature of Grid resources. With respect to other platforms (e.g. PC/WS clusters, Virtual Private Networks), this goal is a novel and very interesting R&D challenge. Besides the heterogeneous and dynamic nature of Grid platforms, the *security* and *administration* issues are crucial for the success of the Grid technology.

For the above reasons, the development of Grid applications requires capabil-

ities and properties beyond those needed in both sequential programming and in parallel/distributed programming, as it requires the management of computations and environments that are typically dynamic and heterogeneous in their composition, and include resource hierarchies with different features (e.g. memory and network). Thus *Grid programming environment* is an outstanding research issue for next generation Grids. There is a growing agreement in the scientific community that: 1) current programming environments, languages and tools are not sufficient to support the development of Grid applications, 2) Grid-aware applications should be designed and implemented exploiting *component technology* [7, 8, 9, 10, 11]. The component based implementation will alleviate the effort required to develop applications, as different *components* can be reused in different combinations to achieve better results. Moreover, the component based implementation will allow single items functional to the whole environment to be reused in different contexts.

The static and dynamic support tools for Grid-aware application development must, in turn, be implemented as a *Grid Abstract Machine* on top of a, possibly standard, *Middleware* layer, such as versions of Globus Toolkit, Unicore, and others. This layer offers the users a view of Grid resources and services which is independent of the underlying basic platforms («fabrics»), such as .NET, J2EE, Websphere and others. Though fundamental, the Middleware layer should not be considered as the machine directly visible to the application designer/programmer: instead, the application designer/programmer should have a more high-level, abstract view of the Grid platform through the assistance of the programming environment tools. The task of Grid Abstract Machine, mentioned above, is just to hide all the very complex details of Grid platforms.

At the middleware level, a fundamental role is played by the *Grid Information Service* (GIS), that is the subsystem that takes care of publishing and searching the information concerning the resources available on the network. The main goal of GIS is to support optimal choices of the computational resources used to schedule a computation, but also to identify the data and the software components available on the Grid. Notable research issues are: *i)* to achieve a high level of accuracy of the dynamic information distributed by GIS without requiring the synchronizations needed to guarantee a consistent view of the global status, *ii)* tools and infrastructures that will allow end users and GIS services to collect information on the different Grid resources status, on faults or possible error situations, *iii)* policies and tools needed to allocate, co-allocate, reserve and schedule Grid resources to fulfill applications requests.

2. The Grid.it project on next generation Grid technology and applications

Grid.it [1] is a FIRB Strategic Project in the area of Enabling Technologies for Information Society, coordinated by the National Research Council, granted by the

Italian Ministry for Education and Research for three years (November 2002 - November 2005). This project, having a strong interdisciplinary character, is aimed at defining, implementing and applying innovative solutions for network computing enabling platforms, oriented towards scalable Virtual Organizations and based on the Grid Computing paradigm. The research topics of Grid.it span from high performance photonic networks, innovative middleware services, high-performance programming environments and application testbeds.

Ahead of the distributed platform based on a distributed infrastructure aspect, in Grid.it a special emphasis is placed on the *high performance* requirement of the applications developed on Grids. This means that the study of the integration of systems and resources, as well as heterogeneity and dynamic situations management, must explicitly handle the case of Grid nodes (which in general are geographically distributed or placed on private virtual networks) being high performance systems, such as parallel machine architectures or clusters. The research on high performance extends to *all the platform levels*, from high bandwidth network to middleware services, and, in particular, resource management, as well to tools and programming environments, as shown in Fig. 1.

At the *programming tools and environment* level the high performance requirement implies that, in the design of scalable VOs, a unifying approach in the development of application should be used. Such approach is able to take into account both the aspects related to the distribution of computations and resources, and of those related to parallelism. The programming environment must therefore be characterized by high degree of portability on different hardware-software systems (or different hardware-software combinations) in an heterogeneous and dynamic context. Portability must be guaranteed not only for code: portability must mainly ensure that the performance matches the configuration of the target system at hand. Fundamental to the programming environment are tool interoperability and high performance application reuse.

The research on *resource management*, at Middleware level, includes aspects of maximal importance as discovery, brokering, scheduling, monitoring and performance evaluation/prediction.

High speed networks needed to support enabling Grid platforms for scalable VOs is an internationally recognized «hot» topic. Within this research activity an important role is played by experiments on very high bandwidth optical networks, based on photonic technology for Grid platforms with high performance sites that extend to metropolitan area.

Beyond the aspects concerning programming environments and resource management, the software technology studied in Grid.it includes some fundamental aspects related to Middleware:

- *security*: secure Grid environments and cooperation among Grid environments belonging to different organizations;

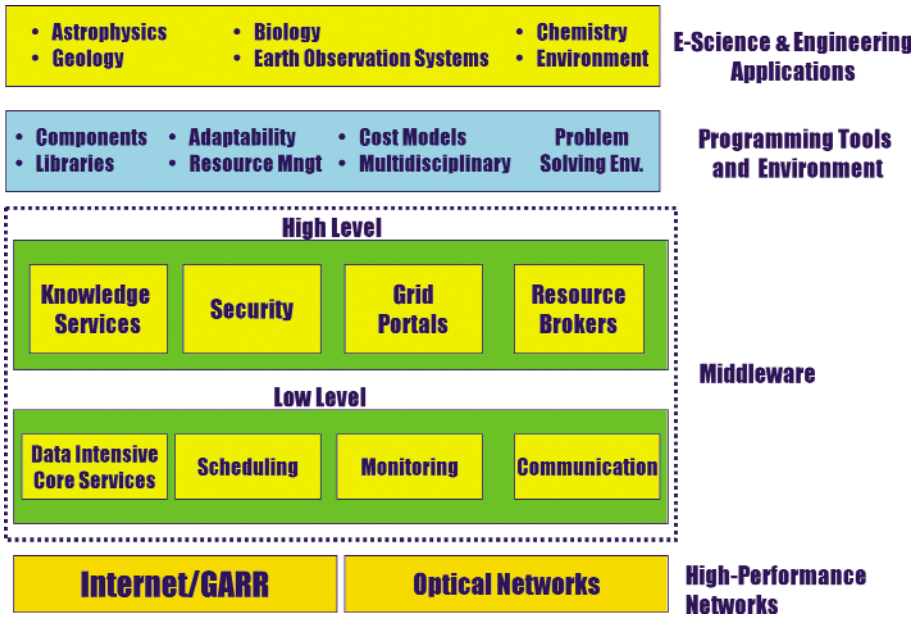

Fig. 1 - Research issues of Grid.it and system levels.

- *data intensive services*: federated database services, visualization and hierarchical management of data and meta-data according to advances and high-performance technique;
- *knowledge discovery services*: Grid services (data mining, search engines, etc.) that provide consistent, efficient and pervasive access to high end computational resources;
- *Grid portals*: Grid enabled applicative services, e.g. the possibility provided to the user to submit tasks and collect results to remote jobs via Web interface.

The design and implementation of *scientific libraries* suitable to be used in an heterogeneous and dynamic context, such as the one of Grid, completes the research on programming environments.

Grid.it includes the development of some *demonstrators* selected within applicative fields that are of maximum interest, not only for their scientific value, but also as testbeds for high performance Grid platforms:

- *Earth observation*
- *Geophysics*
- *Astronomy*
- *Biology*
- *Computational chemistry*

In order to be able to implement and experiment the ideas and the results of the project, some Grid infrastructures (two by CNR, one by INFN, one by ASI)

are implemented, on a national scale, based on the GARR network. This project result provides the community with national Grid resources to be used in different computational science research sectors, and also for commerce, industrial and social service applications.

Six High Qualification Centres (Research Units) participate to the project:

1. ISTI-CNR (D. Laforenza)
2. ISTM-CNR (M. Rosi)
3. ICAR-CNR (A. Murli)
4. INFN (M. Mazzucato)
5. CNIT (G. Prati)
6. ASI (G. Milillo).

A large number of University departments (Pisa, Cosenza, Padova, Perugia, Rome Tor Vergata, Rome La Sapienza, Bologna, Naples, Genova, Milano Bicocca, Turin, Venezia, Trieste, Bari, Lecce) is coordinated in the context of these six Research Units.

The principal investigator is Marco Vanneschi, Dipartimento di Informatica, University of Pisa and ISTI-CNR, Pisa. The Technical Board is coordinated by Domenico Laforenza, ISTI-CNR, Pisa.

The research Workpackages and respective coordinators are listed below:

WP1. Grid Oriented Optical Switching Paradigms (P. Castoldi, CNIT, Pisa)
WP2. High Performance Photonic Testbed (S. Giordano, University of Pisa)
WP3. Grid Deployment (M. Mazzucato, INFN, Padova)
WP4. Security (M. Talamo, University of Rome «Tor Vergata»)
WP5. Data Intensive Core Services (M. Mazzucato, INFN, Padova)
WP6. Knowledge Services for Intensive Data Analysis, Intelligent Searching, and Intelligent Query Answering (Franco Turini, University of Pisa)
WP7. Grid Portals (G. Aloisio, University of Lecce)
WP8. High-performance Component-based Programming Environment (M. Danelutto, University of Pisa)
WP9. Grid-enabled Scientific Libraries (A. Murli, University of Naples and ICAR-CNR)
WP10. Grid Applications for Astrophysics (L. Benacchio, INAF, Padova)
WP11. Grid Applications for Earth Observation Systems Application (G. Milillo, ASI, Matera)
WP12. Grid Applications for Biology (A. Apostolico, University of Padova)
WP13. Grid Applications for Molecular Virtual Reality (A. Laganà, University of Perugia)
WP14. Grid Applications for Geophysics (A. Navarra, INGV, Bologna)

The Grid.it project is aimed at playing an important role in the training of highly qualified young people. A relevant part of the project cost is reserved to contracts for young researchers.

3. The Grid.it software technology for Grid-aware applications development

In this section, we will describe in more detail one of the research tracks of Grid.it, concerning the software technology for application development. In the context of Grid.it organization, WP8 is the responsible for this activity, however there is a strong coordination with activities of several WPs around the approach and tools of WP8, namely WP1, 2, 4, 6, 9, 11, 13.

As introduced in Section 1, a *Grid-aware application* must be able to deal with heterogeneity and dynamicity in the most effective way (*adaptive* applications), in order to guarantee the *specified* level of performance in spite of the variety of run-time events causing modifications in resource availability (load unbalancing, node/network faults, administration issues, emergencies, and so on). With respect to traditional platforms, now it is much more important to rely on application development environments and tools that guarantee *high-level programmability* and *application compositionality*, software *interoperability* and *reuse*, and, at the same time, to be able to achieve *high-performance* and *capability to adapt to the evolution of the underlying technologies* (networks, nodes, clusters, operating systems, Middleware, and so on) [12, 13, 14, 15].

Our view of Grid application development is summarized by the level structure shown in Fig. 2.

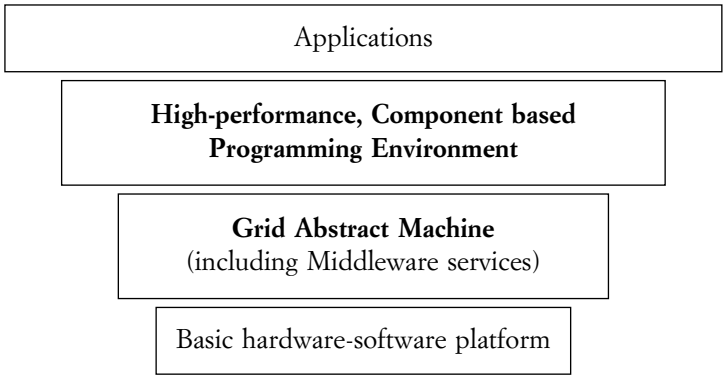

Fig. 2 - The role of Programming Environment in Grid application development.

The Programming Environment is centered on the existence of a high-level, high-performance programming model and related development tools. A high-level view of compositionality, interoperability, reuse, performance and application adaptivity characterizes the Programming Environment we advocate. Applications are expressed entirely on top of this level. The level denoted by *Grid Abstract Machine* includes all the functionalities to support the preparation, loading and execution of the applications expressed in the formalism of the programming environment and

transformed by the compiling tools. The Grid Abstract Machine includes the functionalities that, in the current view of Grids, are provided by the *Middleware tools and services*, e.g. moving bottom-up: the Connectivity (micro-kernel), Resource (resource management services) and Collective (collective and dynamic resource control and allocation) levels. This Middleware may be one of the current/standard products (Globus Toolkit and its evolutions), or a subset of the services performed by them.

The Grid Abstract Machine exploits a subset of the Middleware services and adds very critical functionalities that support the programming model and the development tools, including all the strategies for resource management and scheduling and re-scheduling, allocation and re-allocation, as well as all the actions concerning the application structuring and re-structuring. By replacing the old-fashion OS-like view – according to which the application development occurs directly on top of the Middleware – by the view centered upon the Programming Environment and the Grid Abstract Machine, we wish to stress the *programming-model based approach* to system design, and, at the same time, to minimize the amount and variety of functionalities that are present in the underlying levels: i.e. these functionalities must be limited just to the support to the programming model and tools used to build Grid-aware, adaptive applications. Potentially, this approach leads to achieve a much better trade-off between programmability and interoperability, on one side, and performance, on the other side.

From the discussion above, it follows that the fundamental research issues, to design innovative platforms for Grid-aware applications, are the programming model and its implementation strategies. Other notable research projects namely GraDS [16, 12, 13, 14, 15], ProActive [17] and Ibis [18], propose to follow a similar approach, each one with its own characterization. See also [23].

The Grid.it software technology is an evolution of ASSIST (A Software development System based upon Integrated Skeleton Technology), a programming environment, developed by the Department of Computer Science at University of Pisa, oriented to the development of parallel and distributed high-performance applications according to a unified approach [19, 20, 21, 22]

3.1. *Programming model: distribution, parallelism, interoperability and adaptivity*

Currently, Grid applications are often designed according to a low-level approach (i.e., by relying on the Middlware services directly, possibly through a Grid portal) and, in many cases, they consist in single jobs or in limited forms of job composition (e.g. DAGs). Parallelism, where present, is limited inside single jobs, in a way that does not affect the external structure of the application (e.g. a job may be a MPI program). The result is that rarely Grid applications are Grid-aware and high-performance.

As discussed in the previous section, our point of view is radically different. It is based on the definition and realization of a programming model with the following features:

1. applications are expressed as compositions of high performance components,
2. a uniform approach is followed for distributed and parallel programming: in general components exploits internal parallelism and are executed in parallel with each other,
3. the strategies to drive the dynamic adaptation of applications are expressed in the same high-level formalism of the programming model.

Fig. 3 summarizes the interrelationships of these features. Such interrelationship forms the conceptual frameworks on which we found our research approach.

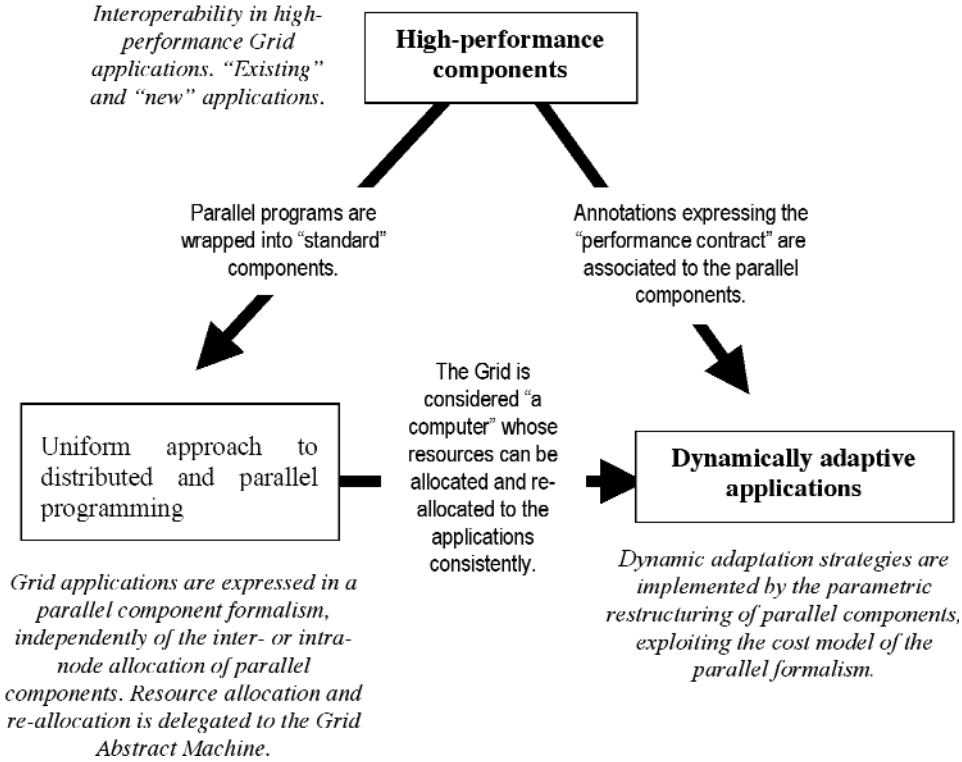

Fig. 3 - The conceptual framework for Grid-aware programming environments.

3.2. *Grid-aware applications as compositions of high-performance components*

Feature 1 is based on the proper exploitation of the *component* technology. In our view, components are the basic mechanism to achieve compositionality by guaranteeing software interoperability and reuse. Here, we assume that the basic features of this software technology are known to the reader.

Achieving high-performance in component technology is currently an impor-

tant research issue. Currently, we are evaluating how the existing standards (CCA [8, 9,], Java Beans, CCM [10], Web Services [11]) can be assumed as starting points to define and realize a robust component-based high-performance programming model, that can be widely accepted and that is able to interoperate in many application areas.

ASSIST provides the *abstraction* of high-performance components and high-performance composition of components, independently of any commercial standard. This allows us to understand the basic features that high-performance components should possess, in particular from the point of view of computation structuring, parallelism exploitation and modularity. These features will be properly merged with one or more commercial standard, or their future high-performance versions, in order to achieve extensive interoperability and reuse. The merging of high-performance programming and component technology must allow the designer to structure the application as the proper *composition of «existing» and «new» components*, i.e. some of them may be already existing (possibly in binary form), other ones are programmed from scratch (e.g. written in ASSIST) or as the combination of existing software into new parallel structures.

The current version of ASSIST (ASSIST 1.2) supports heterogeneity and the interoperability with several currant standards, in particular the *CORBA interoperability* [10]: that is, not only an ASSIST program can act as a client of a CORBA server, but ASSIST programs can be easily defined as, and automatically transformed into, CORBA servers invoked by any CORBA client. Though referred to an object-oriented approach, this experience proves that interoperability features can be merged into the ASSIST model, in order to design applications as composition of components, some of which are possibly parallel.

3.3. *Uniform approach to distributed and parallel programming for Grid-aware applications*

Despite the current limitations in Grid application development, Grid applications have to be *distributed* in the real meaning of the word, as known in theory since many years. With Feature 2 of the conceptual framework we further characterize this concept: we design a Grid application as a *parallel program* described by the parallel composition of parallel components (and possibly existing components). *No distinction is made a priori between parallelism and distribution*, i.e. between modules to be executed in the same (possibly parallel) Grid node or in distinct Grid nodes. In the same way, we do not restrict the application to be a single (sequential or internally parallel) job or a DAG of jobs. In general, the structure of the application can be any graph whose nodes are (parallel) components and the arcs are the mechanisms for their composition and interaction. The programming model of ASSIST is based on this concept.

At this point, it is important to clarify that modeling a Grid application as a

parallel program does not necessarily mean that we are considering a Grid merely as a parallel machine, though in some cases this is a meaningful and effective view. There may be applications in which we could not be interested in inter-component parallelism or in optimizing such potential parallelism, possibly exploiting the parallelism at the intra-component level and forcing distinct components to be allocated onto distinct Grid nodes.

However, there are strong reasons in support to a *uniform view of distributed programming and parallel programming*. Provided that the module granularity is determined properly, there are many applications that can greatly benefit from inter-node parallelism, while additional performance is gained at the intra-node level. We point out that, because of the heterogeneous and dynamic nature of Grid platforms, the a-priori distinction between inter- and intra-node parallelism may be difficult or, by forcing it in some way, it may cause a sensible degradation in performance or fault-tolerance. Instead, we believe that the distinction between inter- and intra-node parallelism must be *delegated to the programming tools*, either at compile- and at run-time: it is for this reason that the resource management, scheduling and allocation functionalities must belong to the Programming Environment support (the Grid Abstract Machine), and in particular to the support of the programming formalism.

As a consequence, an approach that does not limit the parallelism opportunities is characterized by much more flexibility and performance: notably, it must be possible to adapt applications, without sensible or no modifications, to changes and evolutions in the underlying platform, such as in node architecture and multiplicity, communication latency or bandwidth, processor power, operating system facilities, and so on. This aspect is consistent with the trends in component technology [7], e.g. *application versioning* according to different requirements of users and/or availability of system resources.

3.4. *Grid-awareness: dynamically adaptive applications*

The considerations above are generalized to the possibility of developing dynamically adaptive applications, i.e. applications whose resource allocation varies at run-time to guarantee a desired level of performance. Re-scheduling and re-allocation of resources should occur because of node unavailability or node unbalancing, or because an increase in performance is required in response to an emergency (e.g. in an Earth Observation application for landslip detection, the response to some events may require a very large increase in computing power that can be rendered available by a large collection of Grid-connected machines).

Currently, when this problem is addressed only partial solutions are mentioned: notably, dynamic code/data movement.

In general, the problem does not consist merely in finding a better allocation of the same code and data, instead we need to take into account other more com-

plex actions that imply a *transformation* of the executable version of the program, such as

- a different degree of parallelism,
- different data distribution and partitioning,
- and also alternative versions of the program implementing the same functionality, i.e. a different implementation of the same component or composition of components.

A rigorous approach to the adaptivity problem can be based upon the following points:

- several *modalities of expressing the structuring and restructuring of a computation* must be available in the programming formalism,
- these modalities must be characterized by a *cost model* (performance model) that can drive the structuring and restructuring phases with reasonable complexity and overhead.

These modalities can correspond to the usage of diferent (combinations of) *parallelism forms*, or *parallelism paradigms*, as it normally happens in *structured parallel programming* models [19]. In such models a consistent set of parallelism forms is provided to the programmer to structure/restructure the application at hand: for example, pipeline, farm or divide&conquer are typical task-parallel (stream-parallel) paradigms, while map, reduce, prefix, scan, stencil are typical data-parallel paradigms. In structured parallel programming, a *coordination language* is adopted that acts as a metalanguage used to compose codes expressed in any standard language (C, C++, Java, Fortran). These codes may be already existing: for example they may be existing programs, libraries, or components themselves.

Parallelism forms have associated a semantic model and a cost model, that make this approach very promising also for Grid programming: because of the existence of the cost model, the static and dynamic implementation of each parallelism form is *parametric* with respect to few parameters. For example, the actual degree of parallelism or the actual number of data partitions can be varied dynamically without affecting the code of the run-time support.

ASSIST is based on the *structured parallel programming* approach. Beyond the «classical» parallelism forms, the ASSIST programming model contains several features (graphs, parallel modules, external objects) that sensibly increase flexibility and expressive power, including the possibility to design *adaptive program structures* (see the previous consideration about the need for alternative versions of the same computation).

Summing up:

- parallelism (and structured parallelism in particular) is not only useful *per se* (i.e. to exploit higher performance of a certain code, possibly allocated onto the same Grid node), but also it has an utilization which is much more consistent with the dynamically adaptive nature of Grid-aware applications: in fact, in our model structured parallel programming is *a way to specify the strategy for structuring and*

for restructuring a component or a composition of components. Components are internally expressed in ASSIST, with the addition of proper scripting *annotations* for specifying the «performance contract» [15] of the component (e.g. performance metrics, critical events, and so on). Notice that, in general, processes of the same component could be rescheduled onto *different* Grid nodes.

Grid.it will develop, on top of available standard services (GlobusToolkit), a Grid Abstract Machine based on an Application Manager (AM) for parallel and distributed applications according to the Grid-awareness principles. AM, that logically is a centralized entity whose implementation may be decentralized, will exploit the functionalities made available by

- *Performance Model*
- *Monitoring*
- *Resource Discovery*
- *Scheduling strategies, both local to single nodes and global to Grid*
- *Allocation strategies of codes and data.*

The current implementation of ASSIST for heterogeneous networks and Grids will be extended in order to support the dynamic allocation of ASSIST programs: this affects the run-time support of ASSIST modules (called *parmod*), so that parts of the same parallel components can be re-allocated dynamically to different nodes according to the decisions of AM.

Modules of ASSIST programs will be wrapped into standard components and, in general, made interoperable with other non-ASSIST components in order to build Grid applications. Moreover, each component will provide a scripting annotation about the «performance contract» to be established with the Grid Abstract Machine.

The following example could serve to clarify the ASSIST-based approach to the design of dynamically adaptive applications. The applications consists of the component composition shown in Fig. 4.

Component C1 is an interface towards a Grid memory hierarchy, that virtualizes and transforms data sets available on the Grid into two streams of objects, the one (whose elements have an elementary type) is sent to C2, and the other (whose elements have array type) is sent to C3. C1 may be an existing component available on the Grid, virtualized by an ASSIST program.

C2 is a component encapsulating an ASSIST program. The «performance contract» of C2 specifies that

- by default C2 is a sequential module executing a certain function F;
- when the Monitoring and Performance Model services generate the event that signals the need or opportunity to adjusting the current performance level ("on restructuring"), C2 is transformed into a *farm* computation whose workers execute the same function F. AM of the Grid Abstract Machine determines the actual number of workers and their allocation to Grid resources: these may belong to the same Grid node (cluster) or to different Grid nodes. This is con-

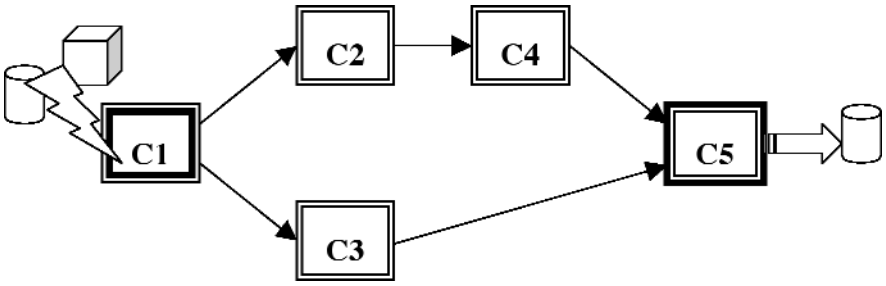

Fig. 4 - Example of an adaptive application expressed by parallel components.

sistent with our conceptual framework, according to which the high-level version of the application is expressed by the structured parallel formalism with annotations, and all the allocation strategies are delegated to the Grid Abstract Machine.

C3 is a component encapsulating an ASSIST *data-parallel* program operating on each stream element of array type. Similarly to the approach described for C2, the «performance contract» of C3 specifies that, by default, the ASSIST program has to be executed on a single Grid node with cluster internal architecture, while "on restructuring" it can modify (increase) the parallelism degree (amount of real processors onto which the data-parallel virtual processors are mapped). The re-allocation may exploit resources belonging to one Grid node or to distinct Grid nodes.

C4 is a component encapsulating an ASSIST program which, by default, is a sequential module, while "on restructuring" it is transformed into a parallel module operating on the input stream according to a data-parallel or a farm style, depending on the values of the module state and on the input values themselves. In this case the adaptation principle is applied at two levels: at the program level and at the allocation level.

C5 is a component encapsulating an ASSIST program operating nondeterministically on the input values received from C3 or C4, and transforming the two streams into a data set. The «performance contract» of C5 specifies that C5 can be allocated and executed only on a certain Grid node and that no reconfiguration can occur. This may be due to security, or privacy, reasons, or to requirements related to the specific resource kinds needed to operate on the data set.

Let us assume that at a certain time C2 is becoming a bottleneck that causes a substantial degradation of performance of the whole application. AM provides to transform C2 into a version with the proper parallelism degree and to re-schedule and re-allocate this new version, assumed that, interacting with the Grid Resource Management services, the necessary resources can be found. In case of restructuring of data-parallel components, the AM strategy must be applied also to the re-distribution of the data constituting the internal state of ASSIST modules. As a

consequence of C2 restructuring, AM could decide to restructure other modules (C4, C5) consistently in order to optimize the global performance.

3.5. *Interrelationships of programming model features*

As shown in Fig. 3, *features 1, 2 and 3, that we advocate for the definition of a Grid programming model, are strongly interrelated*. Feature 1, that implies interoperability, is fundamental for being able to structure complex application that include existing and /or predefined software components, and their «glue» is made possible and easy by the structured parallel programming approach. This feature is also fundamental to allow «legacy code» usage in Grid programs.

Feature 3 requires that components of an application can be rescheduled and restructured dynamically: in turn, *this requires feature 2 (uniform approach to distributed and parallel programming)* because processes of the same parallel component could be restructured and reallocated onto different and distinct nodes, even in the case that at launch time this component has been allocated onto the same node in a sequential or differently parallelized fashion. The parametric feature of structured parallel programming makes the realization of a performance model for the dynamic restructuring of applications feasible.

Summing up:

- a Grid-aware application can be designed as a parallel program, properly «wrapped» into a components structure (together with some possibly pre-existing components), without distinguishing between inter- or intra-node parallelism at the implementation level. Provided that an initial allocation of the components is done at launch time, the allocation of parts of the same components can be modified at run-time (both in identities of nodes and in amount of nodes) to deal with the dynamic adaptation strategies expressed in the same parallel formalism.

Finally, we observe that dealing with the complexity of the Grid programming model has beneficial effects on the same parallel programming principles *per se*. In fact, the possibility to express dynamically adaptive computations also contributes to the solution of *irregular and dynamic problems in parallel programming*, i.e. computations that cannot efficiently be expressed according to predefined paradigms and/or that need substantial modifications according to some data values known at run-time (e.g. parallel Barnes-Hut algorithm), including some *interactive* applications. ASSIST aims to be a solution to this problems too, since it goes beyond the typical limitations of «classical» parallelism forms in dealing with irregularity, dynamicity and interactivity.

4. Conclusive remarks

Grid.it is an Italian research project that aims to contribute to the development of an innovative technology for Next Generation Grids in cooperation with

the main European Research Centres and projects. In Section 3 of this paper we have sketched the features of one of the Grid.it research track, namely the application development environment. Detailed information about all the research tracks of Grid.it can be found at www.grid.it.

Currently several Grid Research initiatives are ongoing or planned at national and European Community level. Though these initiatives provide a rich set of advanced technologies, methodologies and applications, these various endeavours are presently uncoordinated and look rather disparate and fragmented. In the 2002-2006 timeframe, the funding of Grid research and deployment at EU level (275 M€) more than doubles passing from FP5 to FP6. During the same period national funding of 500 M€ for Grid research and development projects has been expended (UK, France, Italy, The Netherlands, Germany, Hungary, Spain, Poland, Czech Republic, Sweden). The totality of these initiatives could provide the EU with the potential to play a world leadership role in Grid technologies and applications. National and EU collaborations have been established with other international players (in the US and Asia-Pacific) and with international standards organisations. However, if Europe wishes to compete with leading global players, it would be sensible to attempt to better coordinate its various, fragmented efforts towards achieving critical mass and the potential for a more visible impact at an international level. Achieving such a coordinated approach will require co-ordination among the funding authorities, collaboration among the individual researchers, and a visionary research agenda.

These are the goals of GridCoord (www.gridcoord.org), an IST Special Support Action of the European Community VI Framework Programme: an ERA Pilot on a co-ordinated Europe-wide initiative in Grid Research, coordinated by the University of Pisa, Department of Computer Science.

Acknowledegments

I wish to thank all the people working in Grid.it: the coordinators of Research Units, Workpackages and activities, and the many young researchers acting on the various tracks. Special thanks are due to Domenico Laforenza of ISTI-CNR – since the beginning, Grid.it has been built and managed through a strong collaboration with him – and to Marco Danelutto of Dipartimento di Informatica, University of Pisa, coordinator of Workpackage 8 on Grid Programming Environment.

REFERENCES

[1] Grid.it Project: *Enabling Platforms for High-Performance Computational Grid Oriented to Scalable Virtual Organizations*. MIUR, FIRB National Research Programme, November 2002. www.grid-it.
[2] I. Foster and C. Kesselman, eds, *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, San Francisco, CA, 1999.
[3] Fran Berman, Geoffrey Fox and Tony Hey, eds, *Grid Computing: Making the Global Infrastructure a Reality*, 2002.
[4] I. Foster, C. Kesselman, S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organization. *The International Journal of High Performance Computing Applications*, 15(3): 200-222, Fall 2001.
[5] I. Foster, C. Kesselman, J. Nick, S. Tuecke. *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*, 2002. http://www.globus.org/research/papers/ogsa.pdf
[6] I. Foster et al. Grid Services for Distributed System Integration. *Computer*, vol. 35, n. 6, 2002, 37-46.
[7] G. Beneken, U. Hammerschall, M. Broy, M.V. Cengarle, J. Jürjens, B. Rumpe, and M. Schoenmakers. Componentware - State of the Art 2003. Background paper for the *Understanding Components* Workshop, Venice, 7-9 October 2003.
[8] R. Armstrong, D. Gannon, A. Geist, K. Keahey, S. Kohn, L. McInnes, S. Parker, and B. Smolinski. Toward a common component architecture for high performance scientific computing. *Proc. 8th High Performance Distributed Computing (HPDC'99),* 1999
[9] *CCA Forum home page*. http://www.cca-forum.org/
[10] *The CORBA & CCM home page*. http://ditec.um.es/~dsevilla/ccm/
[11] W3C.*Web Services home page*. http://www.w3.org/2002/ws/
[12] H. Dail, F. Berman, H. Casanova. A modular scheduling approach for grid application development environments. *Journal of Parallel and Distributed Computing*, 63 (5), 2003.
[13] F. Berman, R. Wolski, H. Casanova, et al. Adaptive Computing on the Grid using AppLeS. *IEEE Trans. On Parallel and Distributed Systems*, 14 (5), 2003.
[14] K. Kennedy, M. Mazina, J. Mellor-Crummey, K. Cooper, L. Torczon, F. Berman, A. Chien, H. Dail, O. Sievert, D. Angulo, I. Foster, D. Gannon, L. Johnsson, C. Kesselman, R. Aydt, D. Reed, J. Dongarra, S. Vadhiyar, and R. Wolski. Toward a framework for preparing and executing adaptive Grid programs. In *Proc. of NSF Next Generation Systems Program Workshop (IPDPS 2002)*, 2002.
[15] F. Vraalsen, R. Aydt, C. Mendes, D. Reed. *Performance contracts: predicting and monitoring grid application behaviour*. TR, Comp. Sc. Dept, Univ. Illinois at Urbana-Champaign, 2001.
[16] GrADS project, /hipersoft.cs.rice.edu/grads. 2004.
[17] ProActive project, /www.sop.inria.fr/oasis/ProActive. 2004
[18] H.E. Bal. Ibis: a Java-based grid programming environment. *Proc. of EuroPar 2003* (invited talk), *LNCS* n. 2790, 2003.
[19] M. Vanneschi. The programming model of ASSIST, an environment for parallel and distributed portable applications. *Parallel Computing* 28(12), 2002, 1709-1732.
[20] M. Aldinucci, S. Campa, P. Ciullo, M. Coppola, S. Magini, P. Pesciullesi, L. Potiti, R. Ravazzolo, M. Torquati, M. Vanneschi, C. Zoccolo. The Implementation of ASSIST, an Environment for Parallel and Distributed Programming. In *Proc. of Euro-Par2003: Parallel and Distributed Computing*, *LNCS* n. 2790, Springer, August 2003.
[21] R. Baraglia, M. Danelutto, D. Laforenza, S. Orlando, P. Palmerini, P. Pesciullesi, R. Perego, and M. Vanneschi. AssistConf: a Grid configuration tool for the ASSIST parallel program-

ming environment. In *Proc. of 11-th Euromicro Conference on Parallel Distributed and Network based Processing (Euro-PDP'03)*, pages 193–200, Genova, Italy, February 2003.

[22] M. Aldinucci, M. Coppola, M. Danelutto, M. Vanneschi, C. Zoccolo. ASSIST as a Research Framework for High-performance Grid Programming Environments, in *Grid Programming Environments*, Research Books, Cunha, & Rama eds, 2004.

[23] *Proc. of EuroPar 2004*, Marco Danelutto, Domenico Laforenza and Marco Vanneschi eds, *LNCS* n. 3149, September 2004.