Scientific Programming 13 (2005) 1–13
IOS Press

Design and implementation of a grid extension for Maple

Dana Petcu[a], Marcin Paprzycki[b] and Diana Dubu[a]
[a]*Computer Science Department, Western University of Timişoara, and Institute e-Austria Timişoara, Romania*
E-mail: petcu.ddubu@info.uvt.ro
[b]*Computer Science Department, Oklahoma State University, USA, and SWPS, Warsaw, Poland*
E-mail: marcin@cs.okstate.edu

Abstract. One of the important issues facing the development of the grid as a computational framework of the future is availability of grid-enabled software. In this context, we discuss possible approaches to constructing a grid-enabled version of a computer algebra system. Our case study involves Maple: the proposed Maple2g package allows the connection between Maple and the computational grids based on the Globus Toolkit. We present the design of the Maple2g package and follow with a detailed discussion of its implementation. Finally, we illustrate performance of Maple2g in a number of experiments.

Keywords: Grid computing, computer algebra system, grid middleware, Maple

1. Introduction

One of the developments that can lead to a wider practical usage of computational grid technologies is grid-enabling of computer algebra systems (CAS). These systems are routinely used by mathematicians and/or engineers to perform complex calculations and are dully seen as an important source of their productivity. However, currently, it is often the case that useful functionalities are implemented in only one particular CAS or as stand-alone programs; sometimes running best on special hardware such as a parallel computer. In either case, it is desirable to be able to augment the CAS with functionality from another software module. It is the grid technology that should facilitate the necessary infrastructure to support this process [8]. We thus begin this paper by reviewing, in Section 2, the state of the art of network and grid-aware CAS.

In Section 3 we follow by summarizing the most important issues in designing a grid-enabled CAS and possible approaches to addressing them. We proceed with a practical example of how a CAS can be made grid enabled and for this purpose Maple became our CAS of choice. The main reason is that, despite its robustness and ease of use, we were not able to locate efforts to link Maple with the grid. Furthermore, it is well known that Maple excels other CASs in solving selected classes of problems e.g. systems of nonlinear equations or inequalities [28]. Finally, Maple has already a build-in socket library for communicating over the Internet, and a library for parsing XML. These capabilities match very well with our goal as they suffice to make Maple a client for an external computational service (in this context one should note a recent trend to use the XML syntax as a de-facto standard in the grid community).

In Section 4 we describe the functionality of Maple2g, the grid-wrapper for Maple. Maple2g consists of two parts a CAS-dependent and a grid-dependent one. Therefore, any change in the CAS or in the grid will be reflected only in one part of the proposed system. Furthermore, the CAS-dependent part is relatively simple and could be easily ported to support another CAS or legacy software.

We complete our description of Maple2g, in Section 5, with details of implementation of access to grid services, in particular, of the grid service search facility.

Finally, experimental results provided in Section VI illustrate the performance of Maple2g, while future research directions are outlined in the last section.

2. Current network and grid-enabled CAS

Grid enabling and augmenting mathematical software tools with functionality from an external software module(s) is the subject of a number of recent and on-going research projects. We will thus summarize the most important developments in this area. A more detailed overview of parallel, distributed and grid enabled symbolic algebra software can be found in [21].

2.1. Accessing external services

Several projects (e.g. NetSolve [3], Nimrod/G [1], Ninf [17]) aim at providing simple ways (APIs, GUIs) to execute software modules available in scientific libraries and/or as stand-alone programs over the Internet/grid. This approach has been commonly labeled "network-enabled server" (NES) [14]. Fully developed NES systems are expected to follow the basic tenets of the grid framework and change the RPC model by incorporating resource discovery, dynamic problem solving capabilities, load balancing, fault tolerance, security, etc.

Currently, the NetSolve system [3] seems to be the most developed NES. It is a grid based server that, among others, supports Matlab and Mathematica as native clients for grid computing. NetSolve provides a web tool that users can query for information concerning all available software modules within the NetSolve system. Furthermore, NetSolve searches for computational resources across the network, chooses the best one available to solve the assigned problem. Then, using retry for fault-tolerance NetSolve facilitates solution of the problem, and returns an answer/answers to the user. A load-balancing policy is used by the NetSolve system to ensure good performance by enabling the system to use available computational resources as efficiently as possible. Recently, a proxy was built for the NetSolve client that is capable of interacting with and making use of Globus resources.

MathLink [29] enables Mathematica to interface with external programs via an API interface. Such an external program sends its arguments to a mathematical computation service and returns result directly into Mathematica.

MathGridLink [27] permits access to the grid service and deployment of new services entirely from within Mathematica. It allows two ways of interaction: one from the view-point of a Mathematica user who wants to use an existing service, and the second one from the viewpoint of a grid user, who wants to access Mathematica as a grid service.

Finally, the Geodise toolkit [6] is a suite of grid-services which are presented to the user as Matlab functions. User of the Geodise toolkit acts as a client to the remote computational resources. Users are authenticated, and then authorized to access resources for which they have rights. They are able to discover available resources, to decide where to run a job, to monitor its status, and to retrieve obtained results. Functions implemented in the "language of Matlab" call Java classes which in turn access the Java CoG API [10].

Note that CASs like Matlab and Mathematica are used mainly as interfaces for grid services (e.g. in Geodise or Netsolve) and not as tools offering services on grids, while MathGridLink envisages support of both types of activities.

2.2. Availability of interactive mathematical web services

Let us now consider a particular situation of interactive access to web enabled computational resources. This scenario can be achieved in a limited way using applets in a web browser. Observe however that computing even the most fundamental mathematical operations such as an integral can require a complicated software module and thus, it is usually necessary to incorporate existing mathematical software into a web application to achieve the required functionality. To implement interactive mathematical web content the following steps are required [25]: install/maintain the "external" computational component, write the wrapper for this component to enable it to be called from another program, write an applet to present the interactive element together with a (most likely Java) servlet which will interact with the wrapper, and write the content and embed the applet or form into the text (with appropriate parameters). JavaMath SDK [26] can assist the user in this process by enabling development (in Java) of conglomerate systems from existing components. It gives a template for writing wrappers and an API for creating and using sessions utilizing these components. For example the code would be part of a servlet on a server, and it would make use of Maple running on a JavaMath server.

To achieve a single generic mechanism that could then be used for all requests for computations, with no extra software that needs to be loaded into the CAS to interface with each new online service, it is necessary to establish a standard for the request-response exchanges. Part of this development would be a standard for the representation of mathematical objects to be exchanged. MathML [13] is one of the possible ways of addressing this problem.

As directly related to our work, where one of our goals is to explore the possibility of adding Maple modules to the set of grid available services, we note that MapleNet [12] offers a software platform for effective large-scale deployment of comprehensive content involving live math computations. MapleNet client is an applet which encapsulates mathematical content. Its publisher offers tools to create applet-based exploration tools, while the server coordinates all the essential software infrastructure, including the general web server, math engines, content, and other databases. Furthermore, MapleNet server manages concurrent Maple instances as necessary to serve client requests for computational and display services, and provides auxiliary services including user authentication.

2.3. *Parallel/distributed CAS versions*

While thus far we have mostly discussed the possibility of making the CAS available as a part of grid services, obviously it can be beneficial if the CAS is capable of utilizing the computational capabilities of the grid itself. In this context observe that, if applicable to a given problem, coarse grain parallelism can be very efficient in an interpreted computation environment such as the CAS. To be able to facilitate development of coarse-grain parallel grid distributed CAS applications, a CAS interface to a message-passing library is needed.

gridMathematica [30] allows the distribution of Mathematica tasks among different kernels in a distributed environment. It is built on a PVM-like architecture. A typical installation of gridMathematica has one master kernel and several computational kernels: the master kernel handles all inputs, divides computations into independent subtasks, schedules calculations for the computation kernels, and collects the results.

There exist more then 30 parallel projects involving Matlab (for more details and a list of projects see [4]). They use diverse approaches to achieving their goal(s): compile Matlab script into a parallel native code, provide a parallel backend to Matlab using Matlab as a graphical frontend, or coordinate multiple Matlab processes to work in parallel. For example, Matlab*P 2.0 [4] is a parallel Matlab environment using the backend support approach. MatlabMPI [7] implements basic MPI routines like send, recv, size and rank entirely in Matlab scripts. PVMTB [2] is a complete Matlab interface to PVM, by means of which Matlab users can prototype applications in the usual high-level programming environment, while retaining the ability to make PVM calls.

Distributed version of Maple have been recently reported in [18] and [22]. For example, Parallel Virtual Maple [18] (PVMaple), was developed to allow several independent Maple kernels on various machines connected by a network to cooperate in solving a problem. This is achieved by wrapping Maple into an external system which takes care of the parallel execution of tasks. Here, a special binary, the command-messenger, is responsible for the message exchanges between the Maple processes, coordinates the interaction between Maple kernels via PVM daemons, and schedules tasks among nodes. Initial experiments show sufficient efficiency in solving large problems to follow this path in Maple2g which has a number of similar functionalities with the PVMaple [19].

In the context of this paper it has to be stressed that while there exist attempts at developing *distributed* Maple, there were no attempts at developing *grid-enabled* Maple (and these are somewhat similar, but different goals), which is the goal of our current research.

In summary, there exist a large number of projects that attempt at grid enabling known computer algebra systems. Their main goals are: (1) to make CAS modules available through the grid, (2) to allow CAS to utilize the grid, (3) to provide direct, web-based access to CAS modules, and (4) to develop parallel and/or distributed CAS by utilizing the networked/grid environment and message-passing parallelism. Out of these goals 1, 2 and 4 are of particular interest to us in this paper. As indicated above, in the case of Maple, goal 3 has been already mostly achieved and thus will be omitted.

3. Developing a grid-aware CAS extension

Let us now look in a bit more details into main issues involved in developing grid enmeshed CAS systems.

Our analysis of the grid aware CAS systems presented in the previous section indicates, that any such system must have at least the following facilities (Fig. 1):

Ability to accept services from the grid: the CAS must be opened to augment its facilities with external modules, in particular it should be able to

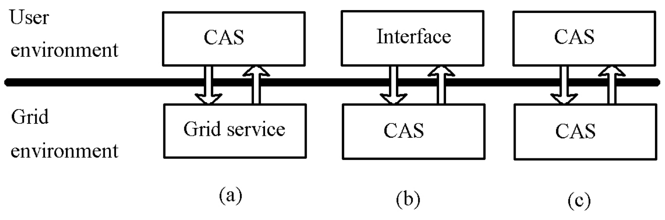

Fig. 1. Operating modes between a CAS and a computational grid: (a) CAS as an interface for the grid services; (b) CAS as grid service; (c) multiple CAS kernels on user and grid sides.

explore computational grid facilities, to connect to a specific grid service, to use the grid service, and to translate its results for the CAS interface;

Being a source of grid or web services: the CAS or some of its facilities must be seen as grid or web services and allowed to be activated by remote users under appropriate security and licensing conditions; furthermore, deployment of the services must be done in an easy way from the inside of the CAS;

Ability to communicate and cooperate over the grid: similar or different kernels of CASs must be able cooperate within a grid in solving general problems; in order to have the same CAS on different computational nodes a "grid-version" of the CAS must be available; in the case of different CASs, appropriate interfaces between them must be developed and implemented or a common languages for inter-communication must be adopted.

There exist multiple ways of achieving the above described functionalities. Rewriting a CAS kernel in order to grid-enable it is likely to be a complicated, time-consuming and high-cost solution. Wrapping the existing CAS kernel in a special code acting as the interface between the grid, the user and the CAS can be done relatively easily as an added-functionality to the CAS. Moreover it can be adapted "on-the-fly" when new versions of the grid software or the CAS in question become available. It is therefore the latter solution that we advocate and pursue here. Let us now describe each of the three functionalities in more details.

3.1. *CAS input from grids – importing grid services into a CAS*

Most CASs have the possibility to launch system commands or to call external modules written in non-native languages. Using these facilities special libraries can be constructed in the CAS language describing, in an user-friendly manner, calls to the grid middleware tools (e.g. those provided by the Globus environment). On the user side, only some minimal facilities to access the computational grid and the CAS must be installed. The grid facilities which can be provided to the user are those currently supported by the grid middleware.

The interface between the grid middleware and the CAS can be written entirely in the CAS language or partially in the CAS language and partially in some other language, more appropriate for the grid middleware (for example, in the case of Globus, such a language would be Java CoG). In the first approach the added-code is oriented towards a particular CAS and is not portable (Fig. 2). The second approach can be more flexible in integrating a new CAS (or multiple CASs) within the user environment and this approach will be pursued here.

It is worth mentioning, that the Geodise project [6] has already adopted the second approach. The Geodise toolbox includes a Java-grid client and a special library mapping current Globus line commands into the Matlab environment. Java-grid client interacts with the Globus server, sending and receiving information from and to the location service(s), authorization service and metadata archive/query service(s). Acting on user request, via the special Matlab functions, it sends to the Matlab interface data concerning the available grid services, and then makes connections to the specific service(s).

3.2. *CAS output into grids – deploying CAS services on grid*

Obviously, access to the CAS facilities must be available to users of the computational grid.

MapleNet [12], allowing the secure access of a thin-client to a Maple server, gives a good example for grid-enabling the CAS: the entire functionality of the CAS can be exposed to the computational grid (while respecting the license conditions).

Full access to the CAS functionality can further facilitate access to other grid services. To achieve this the

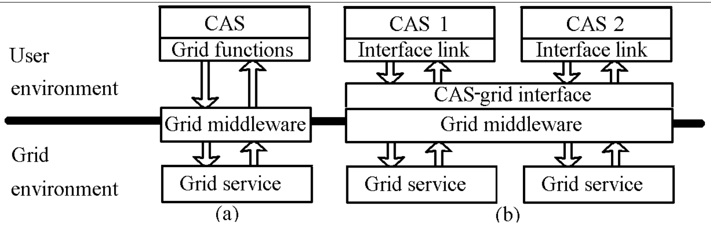

Fig. 2. CAS openness towards grid services: (a) using specific grid-aware function library; (b) using a general CAS-grid interface.

entire CAS kernel has to be rebuild to construct a multi-threaded version or a wrapper is to be provided in order to launch separate CAS kernels to support individual user request.

The CAS functionality can be also exposed only partially by exploring a possibility to implement a visibility hierarchy with different levels of security (Fig. 3) on each level.

A CAS installed within the user or within the grid environment can be used to deploy services in other languages than the one provided by the CAS. Here, facilities to export codes, existing in most CASs, are utilized.

3.3. CAS over the grids – grid-aware distributed CAS version

The computational power provided by the CAS can be augmented by using several other CAS kernels (the same or different CASs). Obviously, this can be utilized only when the problem to be solved can be split between these kernels or a distributed-memory parallel method is used in order to solve it. The usage of a standard message-passing interface for inter-kernel communication allows the portability of the parallel version of a CAS and, in particular, an easy deployment on clusters and grids (Fig. 4).

The two extreme approaches to design the interaction with the message-passing interface are minimal, respectively full, access to the functions of the message-passing interface. In the first case the set of functions is restricted to those allowing to send commands and receive results from remote kernels. In the second case it is possible to enhance the CAS with parallel or distributed computing facilities, allowing the access of the CAS to other parallel codes than the ones written in the CAS language (the message-passing interface can be used as an interpreter between parallel codes written in different languages, including those of different CASs).

3.4. A functional approach

In the next section we describe a prototype of a grid-enabling wrapper for the Maple. Having in mind the above summarized approaches to grid-enabling CASs, we have considered the following roles as the most appropriate for our prototype:

1. the CAS-grid-interface from Fig. 2(b),
2. the CAS-grid-service from Fig. 3(b),
3. the master/worker interfaces from Fig. 4.

The first selection was made so that any change in the CAS or in the grid will be reflected only in the corresponding part of the wrapper. Moreover this allows the CAS-dependent part to be relatively simple and easy to be ported to support another CAS or a legacy software artifact. The same idea is motivating also the second selection. Finally, grid enabling should also lead to the capability for large-scale distributed computing and thus the last choice.

4. Case study: Maple2g

We proceed with a practical example of how a CAS, in our case Maple, can be made grid enabled. Maple2g package allows the connection between Maple and computational grids based on the Globus Toolkit.

The prototype of a grid-enabling wrapper for Maple, consists of two parts a CAS-dependent and a grid-dependent one:

– *m2g*, the Maple library of functions allowing the Maple user to interact with the grid/cluster middleware;
– *MGProxy*, the middleware, a package of Java classes, acting as interface between m2g and the grid environment.

The m2g functions are implemented in the Maple language, and they call the MGProxy which accesses the Java CoG API. MGProxy acts as intermediary be-

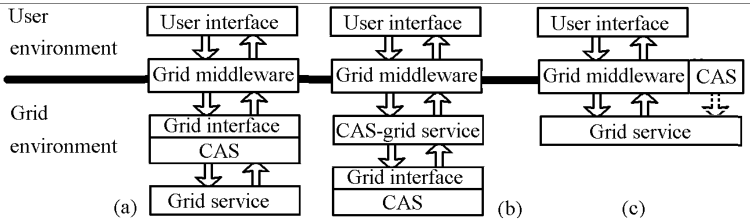

Fig. 3. CAS as grid service: (a) entire functionality exposed as grid-service; (b) partial exposing; (c) using CAS to create stand-alone grid-services.

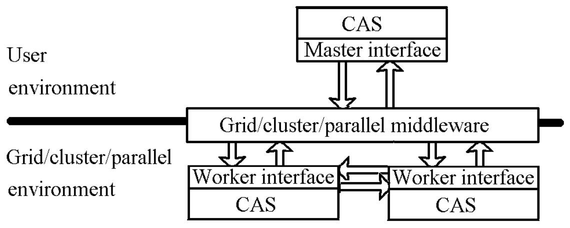

Fig. 4. Grid-version of a distributed CAS.

tween the Maple and the Globus middleware (as the CAS-grid interface from Fig. 2(b)).

Maple2g has three operating modes:

1. *user mode* for external grid-service access;
2. *server mode* for exposing Maple facilities as grid services;
3. *parallel mode* for supporting Maple kernels co-operating over the grid.

Let us now discuss them in more details.

4.1. User mode: grid-service access in Maple2g

In order to make grid services available to the user of the CAS the coupling with the services exposed within the grid has to be performed in a transparent way. This implies that call to service methods should be done only in the CAS native language syntax. In order to achieve this we have incorporated a suite of Maple functions which allow the m2g package communication with the services available within the grid.

In the current version of Maple2g we have implemented a minimal set of functions (described in Table 1) allowing access to the grid services.

MGProxy is activated from inside the Maple environment by the m2g command m2g_MGProxy_start. The user command(s) from the user interface are send to the MGProxy via a socket interface, when m2g_getservice and m2g_jobsubmit are invoked. MGProxy contacts the grid services, queries the contacted services, and sends to the Maple interface the results of performed queries. The m2g_results command provides users with results of computations. Maple commands are passed in the system as strings and results are presented in the MathML format.

4.2. Server mode: Maple services on grid

Concerning access to Maple as a service, Maple2g is similar to the MapleNet [12]. The main difference is that instead of implementing a "new version" of Maple, we have used the classical kernel and a wrapper.

In the current version of the Maple2g prototype, the access to the fully functional Maple kernel is available from the grid (MGProxy acting as CAS-grid interface in Fig. 3(b)): we have implemented only an account check procedure in order to verify the user rights to access the licensed version of Maple provided on the grid. Obviously, our system can be modified to restrict user-access to a subset of Maple commands or function libraries, but this type of enhancement is outside of focus of our current interest.

The user interface activates a simple Java applet which allows the user to send Maple commands as strings via a socket connection to a local Maple2g process awakened in the user mode by the Java applet initialization.

The connection with the remote Maple kernel is established at the initialization stage by sending a specific string in the format in which m2g_jobsubmit sends the information, specifying in this case the remote MGProxy as a grid-service. MGProxy activates a Maple process (which enters an infinite cycle of interpreting

Table 1
M2g functions enabling Maple to use grid services

Function	Description
m2g_connect()	Connection via Java COG to the grid
m2g_getservice(c, l)	Search for a service c and give a link to it, retrieve its location l
m2g_jobsubmit(t, c)	Based on the service location retrieved in the previous step, perform a job submission, in the grid environment, labeled t: the command from the string c is send to the MGProxy which treats it as a grid-service request
m2g_jobstop(t)	Stop the job labeled t
m2g_status(t)	Queries the status of the submitted job labeled t
m2g_results(t)	Retrieve the results of the submitted job labeled t
m2g_MGProxy_start()	Start the MGProxy
m2g_MGProxy_end()	Stop the MGProxy and the grid connection

commands incoming via the socket interface from the MGProxy), acts as a server waiting for external calls, interprets the requests, sends the authentications requests to the Maple twin process, waits for the Maple results and sends them back to the user. After the connection is established, Maple commands can be send to the remote Maple kernel (via the MGProxy), returned in the MathML format.

4.3. Parallel mode: Message passing interface in Maple

Parallel codes using MPICH as the message-passing interface can be easily ported to grid environments due to the existence of a MPICH-G2 version which runs on top of the Globus Toolkit. On other hand, the latest Globus Toolkit is build in Java, and the Java clients are easier to write. This being the case, we selected mpiJava [16] as the message-passing interface between Maple kernels. It requires an already installed MPI version, either MPICH or MPICH-G2.

In Maple2g a small number of commands have been implemented and made available to the user, for sending commands to other Maple kernels and for receiving results from them (Table 2).

MGProxy is activated from user's Maple interface with several other MGProxy copies by the m2g_maple command. The copy with the rank 0 enters in user mode and runs in the user environment, while the remaining copies enter in a server mode. Communication between different MGProxy copies is supported via the mpiJava.

These facilities are similar to those introduced in the PVMaple [18] and in the Distributed Maple [22]. The user run copy of Maple is seen as the master process, while the other Maple kernels are working in a slave mode. Note that command sending is possible not only from the user's Maple interface, but also from one kernel to another (i.e. user issued commands can include send/receive commands exchanging messages between slaves).

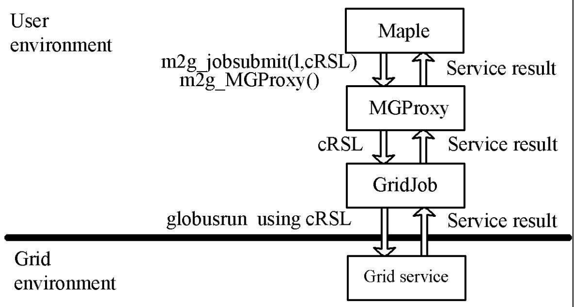

Fig. 5. Communication flow when accessing grid services from Maple.

As a side-note, we have tested the feasibility of PVMaple type approach to development of distributed Maple applications on a small PC cluster. We have observed a reasonable speedup obtained when splitting time-consuming computations. Detailed report as well as a complete description of functionality of this component of Maple2g can be found in [20].

5. Accessing grid services from Maple: Implementation details

Maple functions made available through the Maple2g package m2g allow the programmatic access to Globus grid enabled resources. The m2g package translates internally functions from the syntax familiar to the Maple user into commands, allowing the initiation and further communication with the MGProxy middleware.

MGProxy acts as an intermediary between Maple and the grid and was written in Java, due to its portability and to the fact that libraries supporting Globus-based grid computing have been already implemented in Java. Java Commodity Grid kit [10] integrates software for grid computing developed by Globus and the Java commodity framework, thus facilitating the development and deployment of grid services, while also permitting the use of web services as parts of the grid.

Table 2
M2g functions for remote process launch/communications

Function/const.	Description
m2g_maple(p)	Starts p processes MGProxy in parallel modes
m2g_send(d, t, c)	Send at the destination kernel labeled d a message labeled t containing the Maple command c; d and t are numbers, c is a string; when 'all' is used in destination field, c is send to all Maple kernels
m2g_recv(s, t)	Receive from the source kernel labeled s a message containing the results from the a previous Maple command which was labeled with t; when 'all' is used in source field, a list is returned with the results from all Maple kernels which have executed the command labeled t
m2g_probe(s, t)	Test if a message labeled t has arrive from the source kernel labeled s; the result is 'true' or 'false'; 'all' can be used in the source field to test if a message labeled t has arrive no matter from which source
m2g_exit()	Kill all MGProxy processes, shutdown the twin Maple kernels
m2g_rank	MGProxy rank in the MPI World, can be used in a command
m2g_size	Number of MGProxy processes, can be used in a command

The procedural steps to communicate within the grid starting from the user's Maple interface are depicted in Fig. 5:

1. *Activate MGProxy.* User commands in Maple syntax, parsed within the m2g package, initiate the communication with the middleware which acts as an intermediary between Maple and the grids. This is performed via the commands for external code invocation (including Java) which are already available in Maple 8.
2. *Grid services invocation.* The user invokes the remote services by issuing commands in RSL syntax.
3. *Job submission.* MGProxy activates GridJob, a Java class encapsulating GRAM job that deals with job submission over the grids.
4. *Results retrieval.* Results can be requested either during the communication or after closing the grid connection.

Table 3 enumerates the main classes of the MGProxy package. MGProxy can be viewed as the entry point to the grid. The commands issued by the user from her Maple interface are passed as strings to the MGProxy which forwards these messages further to the MapleListener class responsible for parsing the messages and calling an appropriate tool for their management.

For the invocation of grid services MapleListener activates, according to the request, either the MDSService or the GridService. MDSService is responsible for the retrieval of information regarding grid-available resources, including software resources. Current version of MapleListener is based on the first of the two approaches for the discovery of services described below. GridService acts as a server for the GridJob which is the client performing the actual task of submitting the job; the client-server communication is being established via sockets. GridService receives commands to be send over the grid to the available services (from the MapleListener) until an 'end of job' is signaled, meaning that the connection with the grid is no longer necessary.

Table 3
Java classes in MGProxy package

Name	Description
MGProxy	Activate the Maple link
MapleListener	Parse the Maple messages
MDSService	Retrieve information regarding the grid resources
GridService	Server for GridJob
GridJob	Client performing the requested job
MapleService	Used for Maple as grid service
MPIMaple	Used for parallel Maple

The underlying principles for service retrieval in Globus, referred to as Monitoring and Discovery Service , can be implemented in two alternative ways. Either, the Metacomputing Directory Service can be used, or the facilities from the new GT3 can be utilized. In Maple2g we have implemented the first approach.

The Metacomputing Directory Service (MDS) [11] provides a directory service which has the functionality of the white and yellow pages directories. It makes available the information regarding the computational resources within the grid and the grid network, i.e. information about the hardware, the software and the system status. While the white pages offers the information concerning the hardware performance, yellow pages deal with the computers of a particular class or with a particular property. This later organizational principle is what we are interested in.

MDS is based on LDAP (Lightweight Directory Access Protocol, part of the X.500 standard), a software protocol for enabling localization of organizations, individuals, and other resources such as files and devices in a network, without knowing their particular location.

An LDAP directory is organized in a simple tree hierarchy and can be distributed among many servery. An LDAP server that receives a request from a user takes responsibility for the request, passing it to other LDAP servers as necessary, but ensuring a single coordinated response for the user.

In order to make use of the LDAP principles within Globus, several steps have to be performed, namely:

1. *Initialization.* LDAP schema files must be updated with the description of the attributes associated with grid-specific entry classes. The directory structure described above has therefore to be adapted such that it would contain the information regarding the grid resources.
2. *Population.* The LDAP directory has to be further populated with information according to the hierarchies established at the initialization phase.
3. *Querying.* Is the essential step as it represents the request for information. The information is retrieved from the directory service.

For the Initialization step, there exists a naming schema for the MDS reported in [15]. The MDSService class was implemented starting from the MDSService class proposed in [23].

We have used a combination of LDAP and Globus commands in order to perform the above operations. Alternatively, Java APIs can be used. The suite of commands and their use is depicted in Table 4.

Once the information regarding the existing services has been obtained the subsequent step is to deploy such service in order to retrieve the result. Information is passed in the form of strings, the GridJob class being responsible for the job submission to the service provider which in turn sends back the result of computation, again in the String format. This result is further retrieved by the Maple user, which can use it in subsequent operations. The results are valid even after the connection is closed, thanks to the label which identifies them.

The GridJob class incorporates the methods described in Table 5. It was written starting from the class proposed in [24]. The GridJob class is responsible for the job submission. Requests received from the Maple's user interface in the RSL syntax are send over the grid to remote resources. The underlying framework used here is Java CoG. The connection is established with the remote server, referred to as the 'gatekeeper', which is responsible for the execution of the job (i.e. a binary executable or command to be run remotely). Both the host and the gatekeeper must comply with the authentication requirements. The Grid Security Infrastructure (GSI) is used for enabling secure authentication and communication over an open network. Globus uses GASS for porting and running the applications requiring I/O files to the Grid environment. Therefore, GridJob starts the GASS server and submits all GRAM job requests to this server. The request is formatted accordingly in the RSL format. Output of the processed job is returned through the MGProxy intermediary to Maple into the user's interface.

6. Experimental results

In order to investigate the performance of the grid-wrapper we have performed several tests on a small Globus-based grid environment:

- *cluster*: 6 local PCs each with a P4 processor running at 1.5 GHz and 256 Mb of memory, connected in a cluster via a Myrinet switch at full 2Gb/s (located in Timişoara, Romania);
- *radio:* 2 remote PCs located in Timişoara, with P4 processors running at 1.2 GHz and 512 Mb memory, and connected with the cluster via a radio connection at full 1 Mb/s;
- *Internet:* 1 remote PC located in Linz, Austria, with P4 processor running at 2.4 GHz and 512 Mb of memory, and connected with the cluster at full 2Mb/s.

Furthermore, in the third experiment, a remote web service was used.

Note, that our experiments are not investigating the efficiency of problem solving. To this effect we are not trying to implement the most efficient ways of solving test problems; we are not trying to optimize task scheduling, or match the underlying hardware architecture of components of our mini-grid. Rather, we are interested in establishing general performance characteristics of Maple2g.

6.1. *Experiment 1: Improve Maple's numerical facilities using grid services*

Maple is a tool for symbolic computations. Dealing with numerical computations is not its strength. In order combine symbolic computations with fast numerical computations it is useful to access external numerical codes. If such a code is registered as a grid service, using Maple2g one can access it through the sequence of steps presented in Fig. 6. We have experimented with two codes:

Table 4
Sequence of steps followed in order to populate the LDAP database and search for available services

Command	Result
slapd -f <conf file>	Activate the LDAP server with the configuration from slapd.conf
ldapadd -h localhost -a -w <passw> -x -D <binddn> -f <.ldif file>	Populate LDAP database with the entries specified in the LDIF file
ldapsearch -x -s <scope> -b <baseDN> filter -p <ldapport>	Search for the objects specified within the filter starting in the directory from the baseDN. The scope restricts the search level
grid-info-mds-core	Retrieve the information for the above queries such as Globus directory, base DNs of servers, slapd process ids.
ldap stop	Stop the ldap server

Table 5
GriJob methods

Method	Description
GridJob(C, p, b)	Contructor responsible for the initialization of the contact string variable C, gatekeeper port p and submission mode b (i.e. whether batch or not)
startGassServer(*credent*)	Starts the Globus GASS Server. Retrieves the output from the GASS server and sends it to the client via GridService and MapleListener as a string
initJobOutListeners()	Initiate/register listeners for non-batch mode jobs
statusChanged(*job*)	Used to notify the implementer when the status of a GramJob has changed. A waiting thread is notified when a job is finished and when this is the case the URL is returned and output
outputChanged(*output*)	When the *output* is modified, performs an update
GlobusRun(RSL)	The default Globus proxy is loaded and user credentials are setup properly. The GASS server is started. The RSL is formatted accordingly to the expected structure. A GramJob instance is created and the object sends a request to the remote host

```
> with(m2g);
[m2g_connect,m2g_exit,m2g_getservice,
 m2g_jobstop,m2g_jobsubmit,m2g_rank,
 m2g_maple,m2g_probe,m2g_rank,m2g_recv,
 m2g_results, m2g_status,m2g_send,m2g_size,
 m2g_MGProxy_end,m2g_MGProxy_start]
> m2g_MGProxy_start(); m2g_connect();
 Grid authorization checked
 Grid connection established
> m2g_getservice("newton",`service_locate`);
["&(resourceManagerContact="myri1.info.uvt.ro")
 (count=1)(label="subjob 0")(directory/home/
  Diana)(executable=/home/Diana/newton)",
 "&(resourceManagerContact="myri8.info.uvt.ro")
 (count=1)(label="subjob 0")(directory/home/
  Dana/)(executable=/home/Dana/newton)",]
> m2g_jobsubmit(3,service_locate[2]);
 Job 3 submitted
> m2g_results(3);
 Solving nonlinear system with Newton method:
 Input in.txt, Output out.txt
> m2g_MGProxy_end();
 Grid connection closed
```

Fig. 6. Accessing in Maple an external numerical nonlinear solver, available as grid service.

Newton package: the Maple2g code from Fig. 6, a solver for system nonlinear equations based on Newton's method, written in C, and the Maple's *fsolve* function;

Gauss package: the same Maple2g code was utilized and a linear system solver based on Gauss' elimination written in Java, as well as the Maple's *linsolve* function.

The test problem was: solve $\sum_{j=1}^{n} a_{ij}x_j^{f(i)} = b_i, i = 1,\ldots,n$, where $f(i) = 1$ in the linear case, $f(i) = i$ in the nonlinear case, while A and b are random matrices.

Table 6 presents the most significant results. The "user-controlled" Maple kernel was executed on a local PCs. Note that in each case code solving the problem was executed on a single computer. The grid-service was launched on:

Local: the same computer as the Maple main kernel;
Cluster: on the other PC in the cluster (see above);
Internet: on the remote PC (outside of the local network);
Maple: only the local Maple was invoked (no grid).

The results indicate that for large numerical problems, Maple2g user can efficiently utilize the external code(s) residing on the grid. The apparent inefficiency of Maple is related to the particular approach to solve our problems and should be ignored.

Times obtained for small problems estimate the overhead introduced by the Maple external code launcher, the Maple2g, and the Globus middleware and the network. The overhead is almost independent of the problem size (small differences result from the size of exchanged messages).

Table 6
Time results (mean values for 5 runs)

Package	Dimension	Local	Cluster	Internet	Maple
Newton	5 eqs.	18 s	38 s	74 s	0.3 s
	20 eqs.	435 s	460 s	496 s	934 s
Gauss	5 eqs.	18 s	37 s	73 s	0.01 s
	100 eqs.	535 s	557 s	596 s	822 s

```
>#Maple code: time to multiply 2 big integers
> L:=[2^25,2^26,2^27]:
> for p in L do a:=2^p-1:
> s:=time(): a*(a+7): print(time()-s): end do:
```

```
//C++ code multbi: time to "*" the integers
#include <cln/cln.h>
 using namespace cln;
 int main() { int i; cl_I p=((cl_I)1<<25;
  for(i=0;i<3;i++){ a=((cl_I)1<<p)-1;
   {CL_TIMING; a*(a+7);} p=p<<1;}
 return(0); }
```

```
>#Maple2g code/results using the * service
>with(m2g): s:=time():
>m2g_MGProxy_start(): m2g_connect():
>m2g_getservice("multbi",`serv_loc`):
>m2g_jobsubmit(10,serv_loc[1]):
>m2g_results(10);
 real time: 5.012 s, run time: 4.486 s
 real time: 25.565 s, run time: 20.470 s
 real time: 23.541 s, run time: 19.910 s
 > m2g_MGProxy_end(): time()-s;
```

Fig. 7. Multiplying three pairs of big integers of the order of tens millions decimal digits – Maple code crashing due to memory and time limitations; C++ code based on CLN library and used as grid service to multiply the same big integers; Maple2g code using the grid service to check the multiplication time.

6.2. *Experiment 2: Faster arbitrary precision arithmetic by accessing grid services*

In order to confirm the results obtained in the first series of experiments we considered a second scenario. Maple works with large integers, which make it useful, among others, in applications involving encryption/decryption. Its multiplication algorithm is a fast one, but still there is a limit as to how large are the two numbers that can be multiplied and how fast this goal con be achieved.

We present an example of multiplication of two large integers consisting of 10 million, 20 million and 40 million decimal digits.

We have attempted running the Maple code presented in Fig. 7 on a cluster PC and encountered the following situations.

When Maple 7 was used, the multiplication used the Karatsuba algorithm and the computation time was around 770 seconds for the first pair and would have taken several hours for the subsequent pairs.

When Maple 9 was used, the multiplication used the Schönhage-Strassen algorithm and the multiplication time was around 9 seconds for the first pair, 21 seconds for the second pair, but the third multiplication could not have been performed (an error “Stack limit reached” occurred, caused by the recursive nature of the algorithm). In the second case, four times more memory was needed to complete the multiplication.

A solution to overcome this limit of Maple is that of using an appropriate grid service which can complete the multiplications.

For testing purpose we have selected the CLN package [9]; a C++ library, which allows computations with integers with unlimited precision and which uses the Schönhage-Strassen multiplication for integers larger than 12 thousands decimal digits. The C++ code for the multiplication of the three pairs of big integers is also included in Fig. 7. The time of multiplications is somewhat shorter, but comparable to the time obtained with Maple 9, with the difference, that this time we were able to complete the task for the largest integer pair.

Time tests have shown that if the CLN package is registered as a grid service, the Maple2g code presented in the same figure can activate it, and the overhead is approximately 16 seconds if the service is local. Additional 12 seconds are required if the service is located remotely somewhere in the cluster, and another 18 seconds are required if the service is located remotely on computers in the other networks (similar results for the radio and the Internet connection).

6.3. *Experiment 3: Accessing a web service for symbolic computations*

The third scenario involved a web service, the Online Gröbner Basis (OGB) [5].

Gröbner basis are intensively used in symbolic computations, e.g. to find the solution of systems of polynomial equations or to find the greatest common divisor of a set of univariate polynomials. The "Groebner" package was introduced in the latest versions of Maple. The gbasis function computes a reduced Gröbner basis. Maple's internal algorithm for gbasis is one of the fastest available for this task, but it still requires a large amount of time for a small number of polynomials. Alternative algorithms can be used by accessing web services like OGB.

OGB is written in PHP. Any valid request from a client containing a sequence of numbers in a specific form result in a computation performed by the server. Finally the results, i.e. the polynomials of the Gröbner basis, are returned in an HTML format.

We adopted the following approach. A grid service, OGBInterface, has been written in Java CoG to interact with the OGB server. It receives, as an argument, a string containing polynomials for which the reduced Gröbner basis is requested. Coefficients and the degrees of the unknowns are extracted from the input string and transformed into the sequence required by the OGB. For example, the Maple input $[a*b*c*d-1, a*b*c+b*c*d+c*d*a+d*a*b, a*b+b*c+c*d+d*a, a+b+c+d]$, for which a Gröbner basis is computed with Maple gbasis on a cluster PC in around 3 seconds, is transformed into "http://grobner.nuigalway.ie/grobner/grobner1.php?tf= $1,1,1,1,2,1,3,1,4,1;-1,1|1,1,1,1,2,1,3,1;1,1,2,1,3,1,4,1;1,1,1,1,3,1,4,1;1,1,1,1,2,1,4,1|1,1,1,1,2,1;1,1,2,1,3,1;1,1,3,1,4,1;1,1,1,1,4,1|1,1,1,1;1,1,2,1;1,1,3,1;1,1,4,1$&&type=2&& $\text{time}=100$" (for which the OGB computation lasts less than one second). Then a Java HttpURLConnection is open and the prepared string is send to the OGB server. The results returned in the HTML format are extracted by the grid service, with its output being the string representing polynomials forming the reduced Gröbner basis.

The overhead introduced by accessing the OGB via the grid service OGBInterface located on the cluster is bigger than in the previous experiments due to the OGB server being located on the Internet.

6.4. Experiment 4: Parallel computations in the grid environment

While additional details and examples of parallel usage of Maple2g can be found in citePVMMPI, here, we continue with the integer multiplication example. We consider the case of using three Maple 7 kernels to speedup multiplication of two integers with 10 million decimal digits. As stated above, the Maple 7 code from

```
>#Maple code: time to multiply 2 big integers
>p:=2^25: a:=2^p-1: s:=time(): a*(a+7): time()-s:
```

```
>#Maple2g code
>Karatsuba:=proc(A,B)
>local N,k,A0,A1,B0,B1,T0,T1,T2,E1,E2:
>N:=max(length(A),length(B)): k:=floor(N/2):
>if(N<10^6) then RETURN (A*B) end:
>with(m2g): m2g_MGProxy_start():
>m2g_connect(): m2g_maple(2):
>A0:=irem(A,10^k): A1:=iquo(A,10^k):
>B0:=irem(B,10^k): B1:=iquo(B,10^k):
>m2g_send(1,100,
 cat("A0:=",A0,":B0:=",B0,":A0*B0;"):
>m2g_send(2,200,
 cat("A1:=",A1,":B1:=",B1,":A1*B1;"):
>T1:=(A0+A1)*(B0+B1):
>T0:=m2g_recv(1,100): T2:=m2g_recv(2,200):
>E1:=parse(cat(T2,T0)):
>E2:=parse(cat(T1-T0-T2,
>substring(convert(10^k,string),2..k+1))):
>m2g_MGProxy_end();
>RETURN(E1+E2) end;
>p:=2^25:a:=2^p-1:
>s:=time():Karatsuba(a,a+7):time()-s;
```

Fig. 8. Multiplying two integers of 10 millions decimal digits – Maple code used for efficiency comparisons; Maple2g code using Karatsuba algorithm and three Maple kernels to speedup the computation.

Fig. 8 running on a cluster PC uses approximately 770 seconds to finish the multiplication.

Using three Maple kernels running the MAPLE2G code from Fig. 8 on a homogeneous cluster consisting of similar PCs connected at 2 Gbs, the running time was reduced to approximately 320 seconds (efficiency of 80%). A 7% loss in efficiency was due to the use of grid environment over the cluster instead of the cluster environment (MPICH-G2 instead MPICH). Another loss of 6% of efficiency was registered when one kernel has run remotely over the Internet outside of the local network (respectively 5% when the radio connection was used).

Finally we modified the code to simulate the first recursion level of the classical Karatsuba algorithm. Here, $T0$, $T1$ and $T2$ defined in Fig. 8 are computed using three distinct processors each; following the algorithm described in Fig. 8. We used the grid environment consisting of all 9 PCs described at the beginning of this section. The total execution time has decreased from 770 seconds in the sequential case to 167 seconds in the grid-parallel case, i.e. an efficiency of 51%. We expect that when additional resources in the grid environment are used, the total execution time will decrease further.

7. Concluding remarks

Developing grid-enabled computer algebra systems is a necessary part of the emergence of true value of grid computing. Several approaches to construct such systems were discussed in this paper.

Following one such path, we developed Maple2g, a wrapper for Maple, enabling it to access the grid services and to be accessed as a grid service. Furthermore, Maple2g allows distribution of computational effort to several Maple kernels running on a parallel computer, a cluster, or a grid.

Currently, Maple2g exists as a demonstrator system with all of the functionalities described above implemented. In the near future it will be further developed to include facilities existing in other systems, in order for it to become comparably robust as NetSolve (in issues like load balancing, fault tolerance, security) or Geodise (in issues like monitoring and authentication).

We will also perform experiments on the grid on a large domain of problems. Experimental results will help guide further development of the system. Deployment of grid services from Maple in other languages than Maple using the code generation tools must be take also into consideration. The next MGProxy version will allow the cooperation between different CAS kernels residing within the same or on different sites of the computational grid.

References

[1] D. Abramson, J. Giddy and L. Kolter, High performance parametric modelling with Nimrod/G: A killer application for the global grid?, in *Proc. IPDPS*, 2000, 520–528, http://www.csse.monash.edu.au/˜davida/papers/ipdps.pdf

[2] J.F. Baldomero, Parallel Virtual Machine Toolbox, in: *Proc. MATLAB*, S. Dormido, ed., 1999, pp. 523–532, http://atc.ugr.es/ javier-bin/pvmtb_eng.

[3] H. Casanova and J. Dongarra, NetSolve: a network server for solving computational science problems, *Inter.J. Supercomputer Appls. & HPC* **11**(3) (1997), 212–223, http://icl.cs.utk.edu/netsolve/.

[4] R. Choy and A. Edelman, Matlab*P 2.0: a unified parallel MATLAB, *Proc. 2nd Singapore-MIT Alliance Symp.*, 2003, in print.

[5] M. Gettrick, OGB: Online Gröbner Bases, http://grobner.it.nuigalway.ie.

[6] M.H. Eres, G.E. Pound, Z. Jiao, J.L. Wason, F. Xu, A.J. Keane and J.S. Cox, *Implementation of a grid-enabled problem solving environment in Matlab*, Proc. WCPSE, 2003, in print, http://www.geodise.org.

[7] J. Kepner, *Parallel programmimg with MatlabMpi*, Proc. HPEC, 2001.

[8] I. Foster and C. Kesselman, *The Grid. Blueprint for a new computing infrastructure*, Morgan-Kaufmann, 1999.

[9] B. Haible, CLN, a class library for numbers, 2004, ftp://ftp.ilog.fr/pub/Users/haible/gnu/ cln-1.18.tar.bz2.

[10] Java CoG Kit, http://www-unix.globus.org/cog/java/.

[11] G. von Laszewski and I. Foster, *Usage of LDAP in Globus*, CSE 225 (High Performance Distributed Computing), http://www. globus.org/mds/globus_in_ldap.html.

[12] MapleNet, http://www.maplesoft.com/maplenet/.

[13] MathML, The W3C's Math Homepage, http://www.w3.org/Math/.

[14] S. Matsuoka and H. Casanova, *Network-enabled server systems and the computational grid*, in *Proc.GF4-WG3*, 2000, http://www.eece.unm.edu/˜apm/WhitePapers/GF4-WG3-NES-whitepaper-draft-000705.pdf.

[15] MDS 2.2 Schemas. Definition of Schema, http://www.globus.org/mds/Schema.html.

[16] mpiJava, http://www.npac.syr.edu/projects/pcrc/HPJava/mpiJava.

[17] H. Nakada, M. Sato and S. Sekiguchi, Design and implementations of Ninf: towards a global computing infrastructure, *Future Generation Computing Systems, Metacomputing Issue* **15**(5–6) (1999), 649–658.

[18] D. Petcu, PVMaple:A distributed approach to cooperative work of Maple processes, in: *LNCS* 1908, J. Dongarra et al., Springer, 2000, pp. 216–224.

[19] D. Petcu, D. Dubu and M. Paprzycki, Towards a Grid-aware Computer Algebra System, in: *LNCS* 3036, eds. M. Bubak et al, Springer, 2004, pp. 490–494.

[20] D. Petcu, D. Dubu and M. Paprzycki, A Grid-based Parallel Maple, in: *LNCS* 3241, D. Kranzlmüller et al., eds, Springer, 2004, pp. 215–223.

[21] D. Petcu, D. Ţepeneu, M. Paprzycki and T. Ida, Symbolic Computations on Grids, in: *Engineering the Grid: status and perspective*, B. di Martino, J. Dongarra, A. Hoisie, L. Yang and H. Zima, eds, Nova Science Publishers, Inc, to appear in Spring 2005.

[22] W. Schreiner, C. Mittermaier and K. Bosa, Distributed Maple – parallel computer algebra in networked environments, *J. Symbolic Computation* **35**(3) (2003), 305–347.

[23] V. Silva, *Querying the Grid with the Globus Toolkit Monitoring and Discovery Service*, http://www-106.ibm.com/developerworks/grid/library/gr-mds.

[24] V. Silva, *Grid Job submission using the Java CoG Kit*, http://www-106.ibm.com/developerworks/library/gr-gridcog.

[25] A. Solomon, Distributed computing for conglomerate mathematical systems, in: *Integration of Algebra and Geometry Software Systems*, M. Joswig and N. Takayama, eds, http://www.illywhacker.net/papers/webarch.ps.

[26] A. Solomon and C.A. Struble, *JavaMath: an API for internet accessible mathematical services*, in Proc. 5th Asian Symp.on Computer Mathematics, World Scientific, 2001, http://javamath.sourceforge.net/.

[27] D. Tepeneu and T. Ida, *MathGridLink – Connecting Mathematica to the Grid*, in Procs. 6th Intern. Mathematica Symposium (IMS 2004), Banff, Alberta, Canada, August 2004.

[28] M. Wester, A critique of the mathematical abilities of CA systems, in: *Computer Algebra Systems: A Practical Guide*, M. Wester, ed., John Wiley & Sons, 1999, http://math.unm.edu/˜wester/cas_review.html.

[29] Wolfram Research, MathLink, http://www.wolfram.com/solutions/mathlink/.

[30] Wolfram Research, gridMathematica, http://www.wolfram.com.