Towards a Grid-aware Computer Algebra System

Dana Petcu[1,2], Diana Dubu[1,2], Marcin Paprzycki[3]

[1] Computer Science Department, Western University, [2] Institute e-Austria, Timişoara,
[3] Computer Science Department, Oklahoma State University
{petcu,ddubu}@info.uvt.ro, marcin@cs.okstate.edu

Abstract. One of the developments that can lead to a wider usage of grid technologies is grid-enabling of application software, among them computer algebra systems. A case study described here involves Maple. The proposed maple2g package allows the connection between the current version of Maple and the computational grid based on Globus.

1 Introduction

Computer algebra systems (CASs) are frequently used by mathematicians or engineers to perform complicated calculations and are rightfully seen as one of major sources of user's productivity. In practice it is often desirable to be able to augment the CAS with functionality from an external software artifact (e.g. pakage, application etc.). Nowadays, in this process one can rely on already available solutions, such as the grid technology.

Several projects aim at providing APIs to execute scientific libraries or programs over the grid. NetSolve [1] is a grid based server that supports Matlab and Mathematica as native clients for grid computing. MathLink [8] enables Mathematica to interface with external programs via an API interface. MathGridLink [6] permits the access to the grid service within Mathematica, and the deployment of new grid services entirely from within Mathematica. The Geodise toolkit [3] is a suite of tools for grid-services which are presented to the user as Matlab functions, calling Java classes which in turn access the Java CoG API. MapleNet [4] offers a software platform to effective large-scale deployment of comprehensive content involving live math computations.

To be able to facilitate development of parallel grid distributed CAS applications, a CAS interface to a message-passing library is needed. There exist more then 30 parallel Matlab projects [2]. gridMathematica [8] allows the distribution of Mathematica tasks among different kernels in a distributed environment.

Maple is utilized as the CAS of choice in our attempt to couple a CAS and a computational grid. The main reason for this choice is that, despite its robustness and ease of use, we were not able to locate efforts to link Maple with grids. Second, it is well known that Maple excels other CAS in solving selected classess of problems like systems of nonlinear equations or inequalities [7]. Furthermore, Maple has already a sockets library for communicating over the Internet, and a library for parsing XML (a data-exchange standard widely utilized in the grid community). Finally, distributed versions of Maple have been recently reported in [5].

To obtain our goal we proceeded by developing Maple2g: the grid-wrapper for Maple. It consists of two parts: one which is CAS-dependent and the other, which is grid-dependent and thus any change in the CAS or the grid needs to be reflected only in one part of the proposed system. The CAS-dependent part is relatively simple and can esily be ported to support another CAS or a legacy code.

2 Developing a grid-aware Maple extension

Our analysis of the grid aware CAS systems indicates that any such a system must have at least the following facilities:

Ability to accept inputs from the grid: the CAS must be opened to augment its facilities with external modules, in particular it should be able to explore 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 an input for the grid: the CAS or some of its facilities must be seen as grid services and activated by remote users in appropriate security and licensing conditions; furthermore, deployment of grid 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 to 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.

Rewriting a CAS kernel in order to improve its functionality towards grids can be a complicated and high-cost solution. Wrapping the existing CAS kernel in 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. In addition, it can also be adapted on-the-fly when new versions of the CAS in question become available.

Maple2g is a prototype of a grid/cluster-enabling wrapper for Maple. As described below it consists of two components, `MGProxy`, a Java interface between Maple and the grid/cluster environment, and `m2g`, a Maple library of functions allowing the Maple user to interact with the grid/cluster middleware.

MGProxy has three operating modes:

1. *user mode*: activated from inside of the Maple environment (by the m2g_MGProxy_start command), receives the user command from the user's Maple interface via a socket interface, contacts the grid/cluster services (including also other MGProxy processes), queries the user requests to the contacted services, and sends the results of the queries to the main Maple interface.
2. *server mode*: activates a Maple twin process (which enters in a infinite cycle of interpreting commands incoming via the socket interface from MGProxy), acts as a server waiting for external calls, interprets the requests, sends the authentications requests to the Maple twin process, receives the Maple results, and sends them back to the user.

Table 1. Functions available in m2g library

Function	Description
m2g_connect()	Connection via Java COG to the grid
m2g_getservice(s, l)	Search for a service s and retrieve its location l
m2g_jobsubmit(t, c)	Allows a job submission in the grid environment labeled with the number t: the command c is a string in the RSL format
m2g_results(t)	Retrieve the results of the submitted job labeled t
m2g_maple(p)	Starts p processs MGProxy in parallel modes
m2g_send(d, t, c)	Send to the destination kernel d a message labeled t containing the command c; d – 'all' or a number, t – number, c – string
m2g_recv(s, t)	Receive from the kernel labeled s results from the command labeled t; s – 'all' or a number, t – number
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

3. *parallel mode*: is activated from user's interface with several other MGProxy copies; the copy with the rank 0 enters in user mode and runs in the user environment, while the others enter in server mode; the communication between different kernels is established through a standard message passing interface.

The current version of Maple2g has a minimal set of functions (described in Table 1) allowing access to the grid services. These functions are implemented in the Maple language, and they call MGProxy which accesses the Java CoG API. For example, accessing a grid-service can be done in the steps described in Fig.1.

The component responsible for accessing Maple as a grid-service is similar to that of the MapleNet [4]. In the current version of the Maple2g prototype, the access to the fully functional Maple kernel is allowed from the grid: MGProxy acting as CAS-grid interface implements only an account check procedure in order to verify the user rights to access the licensed version of Maple residing on the grid.

Parallel codes using MPICH as their message-passing interface can be easily ported to grid environments due to the existence of a MPICH-G version which runs on top of the Globus Toolkit. On other hand, the latest Globus Toolkit is build on Java, and the Java clients are easier to write. This being the case, we selected the mpiJava as the message-passing interface between Maple kernels.

```
> with(m2g): m2g_MGProxy_start(); m2g_connect();
  [m2g_connect, m2g_getservice, m2g_jobstop, m2g_jobsubmit, m2g_maple,
   m2g_MGProxy_end, m2g_MGProxy_start, m2g_rank, m2g_recv, m2g_results,
   m2g_send,m2g_size]
  Grid connection established
> m2g_getservice("gauss",`service_location`);
   ["&(executable=/home/Diana/m2g/Gauss.sh)","&(executable=/tmp/gauss)"]
> m2g_jobsubmit(3,service_location[1]);
    job submitted
> m2g_results(3); m2g_MGProxy_end();
  Solving linear syst. with Gauss method: Input in.txt, Output out.txt
  Grid connection closed
```

Fig. 1. Accessing in Maple an external linear solver, available as a grid service

```
>with(m2g): m2g_MGProxy_start(); m2g_maple(4): d:=`all`:
>m2g_send(d,1,"f:=(x,y)->(x^2-y^2+0.32, 2*x*y+0.043):"):
>m2g_send(d,2,"g:=(x,y)->x^2+y^2:"):
>m2g_send(d,3,"h:=(x,y)->if g((f@@130)(x,y))<1 then 0 else 1 fi:"):
>m2g_send(d,4,"plot3d(`h(x,y)`,grid=[400/mg_size,400],y=-1.15..1.15,
>  x=-1+2*mg_rank/mg_size..-1+2*(mg_rank+1)/mg_size,style=point,
>  view=[-1..1,-1.15..1.15,0..0.1],orientation=[90,0]);"):
>plots[display3d](m2g_recv(`all`,4)); m2g_MGProxy_end();
```

Fig. 2. A Julia fractal: the plotting time of order $O(10^3)$ s in the sequential case can be reduced by a speedup factor of 3.5 using 4 Maple kernels treating equal vertical slices

In Maple2g a small number of commands is available to the user, for sending commands to other Maple kernels and for receiving their results (Table 1). These facilities are similar to those introduced in PVMaple [5]. The user's Maple interface is seen as the master process, while the other Maple kernels are working in a slave mode. Command sending is possible not only from the user's Maple interface, but also from one kernel to another (i.e. a user command can contain inside a send/receive command between slaves).

To test the feasibility of this approach to developing distributed Maple applications, tests have been performed on a small PC cluster (8 Intel P4 1500 MHz processors, connected by a Myrinet switch at 2Gb/s). When splitting the time-consuming computations we have observed an almost linear speedup. While a detailed report on parallel Maple2g is outside of the scope of this note, in Fig.2 we give an example of a parallel Maple2g code.

At this stage Maple2g exists as a demonstrator system; however it already shows its potential. 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 or Geodise. Tests on grid on a large domain of problems will help guide further development of the system. Deployment of grid services from Maple in other languages than Maple using the code generation tools will be also taken into consideration. Finally, the next version of MGProxy will allow the cooperation between different CAS kernels residing on the grid.

References

1. Casanova H. and Dongarra J.: NetSolve: a network server for solving computational science problems. Inter.J. Supercomputer Appls. & HPC, 11-3 (1997) 212–223
2. Choy R., Edelman A.: Matlab*P 2.0: a unified parallel MATLAB, In Procs. 2nd Singapore-MIT Alliance Symp. (2003), in print.
3. Eres M. H. et al: Implementation of a grid-enabled problem solving environment in Matlab. In Procs. WCPSE03 (2003), in print, www.geodise.org
4. MapleNet. www.maplesoft.com/maplenet/
5. Petcu D., PVMaple: A distributed approach to cooperative work of Maple processes. LNCS 1908, eds. J.Dongarra et al., Springer (2000) 216–224
6. Tepeneu D. and Ida T.: MathGridLink - A bridge between Mathematica and the Grid. In Procs. JSSST03 (2003), in print.
7. Wester M.: A critique of the mathematical abilities of CA systems. In CASs: A Practical Guide, ed. M.Wester, J.Wiley (1999), math.unm.edu/~wester/cas_review
8. Wolfram Research: MathLink & gridMathematica, www.wolfram.com.