

NWP on a GRID compute environment

Gerard Cats

The concept of a GRID is explained. Its applicability for NWP, and in particular for HIRLAM, both in the long and the short term, is discussed. It is concluded that within a few years the HIRLAM system may be substantially simplified by application of a simple GRID.

Introduction

Over the past few years, the concept of GRID computing has had lot of interest in the research of high performance computing and networking. The general idea is that it should be possible to combine all available compute resources into one big compute facility (including data handling, storage, *etc.*). Such a facility is called a GRID computer, GRID environment, or simply GRID. The user would submit a job to the GRID, which would then decide on which hardware the job should run, with due care of access rights, security, data availability, and so on.

One of the first GRID environments was developed in the GLOBUS project (www.globus.org). As a demonstration project it was used for MM5 (www.globus.org/research/applications/mm5.html). So there has already been an application of a GRID for a numerical weather code. Of course, this was only a very simple implementation; security issues were fairly well covered, but most other aspects of a GRID, like data availability, prioritisation, *etc.*, had to be taken care of manually by the user.

Several projects are now furthering the GRID architecture (*e.g.* www.teragrid.org). Within the list of projects supported by the European Commission DataGRID (www.eu-datagrid.org) and EuroGrid (www.eurogrid.org) have a meteorological component. At the recent Workshop on System Collaboration (Météo-France, 13 to 15 May 2002) there were presentations of these two EC projects. These projects build on the American GLOBUS and the German UNICORE (www.unicore.de) GRID computers, *resp.*

In this document I will give my views on the applicability of a GRID environment for numerical weather prediction, and in particular, on how HIRLAM can profit from implementing it on a GRID within the time range of a few years. In my discourse I will forsake most of the interesting features of a GRID, so as to avoid any suggestion that it is unrealistic to expect GRID to deliver benefits for HIRLAM. In stead, I will show the advantages of one feature that has already been realised, namely the ability of a GRID to keep track of in- and output files. Yet, it is not my intention to suggest that this is the most important feature of a GRID. On the contrary, A GRID consists of a number of software layers, of which I estimate the most important those that stimulate cooperation between hard- and software providers (see Foster's (2002) paragraph on "Collaborative work" on page 44). HIRLAM, in particular, would profit from these features; in the first place because the project is distributed by nature; but also the fact that most HIRLAM members rely heavily on remote (*e.g.* ECMWF) compute resources for research makes HIRLAM a particularly good target for a GRID.

The GRID concept

The idea that led to the concept of a GRID is that many a user has access to many computers, and often spends a lot of time in finding the machines on which the best turn-around time is achieved. Would it be possible to automate this decision process? The designers of the GRID concept thought that the way the user would want to use computer resources is probably similar to the way the power grid (220/240V) is used for electrical appliances: plug in on an outlet, and get the power that you require. You do not have to worry about where the power was generated. You are charged for what you use, and your “access rights” are limited by your fuses and your budget. The power is mostly supplied by large-scale plants, but if you put a windmill on your roof, you may become a power supplier yourself, and sell electricity to the grid.

Similarly, if you require compute resources, simply plug in onto the grid of compute resources and you should get the resources you require. This explains why this compute environment is called a “GRID”.

Of course, the situation with compute resources is far more complicated than with the electricity supply. A GRID has to know for each user what machine(s) the user has access to, and at which priority. The user’s identity must be certified. For each submitted job, the GRID has to find the most appropriate hardware, balancing compute and network costs and turn-around time. For this balance, it has to be able to judge the efficiency of the job on available hardware (*e.g.*, does it vectorise?) and the costs of getting the input data to the executing hardware and getting the output back. This requires that the GRID have a database containing the location of the input data, and of copies of the input data that possibly are closer to the target architecture. A separate issue is to identify which input data the job requires, and to insert format conversions, where needed. A particular kind of input file is the executable to be run; the format conversion in this case is the compilation from Fortran (*etc.*) sources. The GRID computer has to know how to create it from the sources; for the estimation of turn-around time and computing costs the GRID must be able to estimate the cost of the compilation process.

NWP on a GRID

Most issues listed in the last paragraph of the previous section have not been solved yet. They are the subjects of a range of research projects, as mentioned in the introduction. Some scepticism about whether all issues will be resolved, even in the long run, seems justified. The question is whether a GRID may become useful for NWP, even if some issues are still outstanding. The main requirements of production NWP are a guaranteed turn-around and security. In the following, I will concentrate on the turn-around, because the prospects are that the security issues can be solved satisfactorily using modern techniques. Initially, I consider the operational, production, implementation of NWP.

The current situation for NWP in Europe is that all institutes that require a guaranteed turn-around of their NWP systems have guaranteed access to some computer system during certain hours of the day. This provides a near 100% guaranteed turn-around, but some back-up mechanism is needed to cover the few tenths of a percent loss of data that is unavoidable.

Consider the simplest non-trivial implementation of a GRID, in which a number of NWP institutes would join their computers into a GRID. If the home institute has exclusive access to the local hardware at the same hours of the day as before, the GRID computer would provide at least the same guarantee of turn-around as the original configuration. But this GRID would provide a better back-up to the local system than what we have now. So provided that the security issues are solved, there is at least one GRID configuration that beats the current situation for NWP in Europe in reliability, namely the simplest non-trivial one. (The trivial configuration is to have just one computer in the GRID).

This “base-line” configuration of a GRID is not very user-friendly, though, when it comes to making use of the additional facilities, unless use is made of the ability of a GRID to make in- and output files available where they are needed; this ability has *e.g.* been implemented in DataGRID. If this feature is not used, access to back-up facilities will usually be limited by the amount of work the user is prepared to spend on them – and thus comes close to the current situation, where the back-up is explicitly configured.

The further the research on GRID computing progresses, the more facilities will become available. In particular, GRID computing will probably allow more user-friendly access to the available computers, sooner or later. Because operational NWP runs are usually configured infrequently, the advantage of increased user-friendliness weighs heavier for experiments done by possibly less experienced users than for the production runs. The big, prospect, advantage of a GRID is that a user can submit a job, at any time, and the GRID will find out where to run the job. At times that the local machines are busy with the production runs, the job will be executed remotely; at other times the job may stay in-house. To be realistic, I suppose that initially the GRID will choose from a limited number of machines, on which the system has been implemented. Depending on the progress of the research projects, the GRID may become cleverer, and the submission of an NWP run, be it for experiments or for operational production, easier. Ultimately, by better usage of the available hardware, NWP compute requirements may become cheaper.

HIRLAM on a simple GRID

I estimate that realistically one may assume that within a few years time a GRID will be available that supports the following features:

- Access control (authentication and authorization) to some systems
- Automatic transfer of in- and output files, but without automatic insertion of format conversions (including compilations), and without automatic identification of required files
- Hardly any or no capabilities of automatic turn-around optimisation or cost considerations

In the following I will describe how this computer already reduces the complexity of the HIRLAM considerably. I will do this for the HIRLAM system at ECMWF.

The complexity of the system at ECMWF is to a large extent due to the need to distribute the tasks in the HIRLAM suite over several machines. The system uses mini-SMS (compatibly with SMS) to control the suite. The suite definition file is generated from a template. Typically, sections of the template look like:

```

ifeq($ENV{COMP CENTRE},ECMWF) # copy analysis input files
  family XCopy1
    trigger ( Execute == complete )
    task Execute
      edit SCRIPT "perl -S EcnCopy1.pl"
      edit ENVT 'LOF=$CYCLEDIR/xcopyAN'
      edit ARGS "$CYCLEDIR/*.dat $CYCLEDIR/sfcan $CYCLEDIR/CMA01"
    endfamily
  family XCopy2
    trigger ( XCopy1 == complete )
    task EcnCopy2
      edit R_HOST HOST0
      edit LOF '$CYCLEDIR/xcopyAN'
      edit SMSNODE $ENV{HOST1}
    endfamily
endif

task Analyse
trigger ( InFiles == complete ) # family InFiles contains, a.o., the above families XCopy1 and XCopy2
edit SMSNODE $ENV{HOST1}
ifeq($ENV{COMP CENTRE},ECMWF) # run on remote host
  task CopyFiles
    trigger ( Analyse == complete )
    edit R_HOST HOST0
    edit tarfiles '$CYCLEDIR/bl$ENV{FCINT} $CYCLEDIR/ae$ENV{FCINT} $CYCLEDIR/an $CYCLEDIR/cof'
    edit SMSNODE $ENV{HOST1}
  endfamily
endif

```

In here, the 3 lines in bold print instruct (mini-)SMS to run the task Analyse, after the task to create the input files has completed, and to run it on the host whose domain name is in environment variable HOST1. The other lines, only active if the computer centre is ECMWF, copy the input files to that host, and copy the output files back to HOST0. For performance, the copying of the input files takes place only if they do not exist on the target host, or if they are obsolete.

On a GRID that cannot identify the in- and output files, or the preferred target, the user must take those responsibilities. Assuming that the GRID knows where the files are, and using generic names for the files (“logical file names”, as opposed to the “physical file names” that identify a file on a disk), the sections of the template, as shown above, would reduce on such a computer to:

```

task Analyse
trigger ( InFiles == complete )
edit SMSNODE $ENV{HOST1}
edit inputfiles 'config.dat sfcan CMA01'
edit outputfiles 'next-bl next-ae an cof'

```

This example clearly shows how much simpler the system becomes even by the use of a simple GRID. (Most likely even a simple GRID would know that a task should not be submitted before its input files are ready, so probably the trigger line could also be discarded). Note that in this example security issues do not play a role, so even if the GRID is not trusted to handle those reliably, all advantages would still be there.

The example above is typical, in the sense that the sequence of <obtain input | execute | write output> occurs several times. In the real sequences, there is additional information on the requirements of the jobs, such as CP time and memory. For clarity, this information has been left out from the example as shown, but the information is essential when the GRID becomes clever enough to use it to launch the job to the host on which those requirements may be met. In that case, there is no need for explicit allocation of hosts by the edit SMSNODE line: the GRID will determine the target node

from the job requirements. This feature of a GRID depends on the availability of a standard way to specify job requirements, which seems to be a dream as yet.

Conclusions

The use of a GRID compute environment for NWP, and in particular for HIRLAM, would substantially reduce the complexity of the system. On the longer term, when the GRID concept has matured, the user will not have to decide where jobs will be run. In the shorter term the GRID will probably lack many of the capabilities of a mature system. Yet, a simple system would already bear advantages like providing a cheap back-up of NWP production runs.

It is most probable that within a year or two from now the European DataGRID project will have resulted in a GRID that is able to keep track of the location of in- and output files at least. Even on a GRID computer with such limited scope, the HIRLAM system would become much simpler than currently. Hence, in the context of the system overhaul, developments on GRID should be closely followed, and whenever useful, applied.

In this paper, I have assumed that the GRID computer will be used to interpret a script that specifies which jobs have to be executed in which order; in the case of HIRLAM, this script is the (mini-)SMS suite definition file. However, we should not take it for granted that the GRID is going to work this way. To match GRID developments with HIRLAM requirements it is essential that HIRLAM takes an active role in those developments; such an active role is certainly warranted, given the great benefits that a GRID can offer us. Although the benefits would be available to a large range of applications, the HIRLAM project in particular may be expected to profit from GRID developments because of the nature of the project and its relation with ECMWF.

Acknowledgments

I thank Nils Wedi (ECMWF), John van de Vegte (KNMI), and Lex Wolters (Leiden University) for their contributions through discussions.

Reference

Foster, I, *The GRID: A New Infrastructure for 21st Century Science*, Physics Today **55**, February 2002, 42-47.