Demonstration of P-GRADE Job-Mode for the Grid[1]

P. Kacsuk[1], R. Lovas[1], J. Kovács[1], F. Szalai[1], G. Gombás[1], N. Podhorszki[1], Á. Horváth[2], A. Horányi[2], I. Szeberényi[3], T. Delaitre[4], G. Terstyánszky[4], and A. Gourgoulis[4]

[1] Computer and Automation Research Institute (MTA SZTAKI)
Hungarian Academy of Sciences, 1518 Budapest, P.O. Box 63, Hungary
{kacsuk, rlovas, smith, szalai, gombasg, pnorbert}@sztaki.hu

[2] Hungarian Meteorological Service, H-1525 Budapest, P. O. Box 38, Hungary
{horvath.a, horanyi.a}@met.hu

[3] Department of Control Engineering and Information Technology, Budapest University of Technology and Economics, H-1117 Budapest, XI., Pázmány Péter sétány 1/D, Hungary
szebi@iit.bme.hu

[4] Centre for Parallel Computing (CPC), Cavendish School of Computer Science, University of Westminster, 115 New Cavendish Street, London W1W 6UW, UK
{delaitt, terstyg, agourg}@cpc.wmin.ac.uk

Abstract. The P-GRADE job execution mode will be demonstrated on a small Grid containing 3 clusters from Budapest and London. The first demonstration illustrates the Grid execution of a parallel meteorology application. The parallel program will be on-line monitored remotely in the Grid and locally visualized on the submitting machine. The second demonstration will use a parallel traffic simulation program developed in P-GRADE to show the usage of the P-GRADE job mode for Grid execution. The parallel program will be check-pointed and migrated to another cluster of the Grid. On-line job and execution monitoring will be demonstrated.

1 Introduction

MTA SZTAKI has developed a parallel program development environment (P-GRADE) [1] for hiding the low-level PVM and MPI APIs from the users and providing a much higher-level graphical programming model with full support for parallel program debugging, monitoring, performance visualization, mapping, load-balancing, etc. P-GRADE is now extended towards the Grid and it stands for Parallel Grid Run-time and Application Development Environment. This environment enables the Grid application programmer to develop a parallel program that can be executed as a Grid job on any parallel site (like a supercomputer, or a cluster) of a Grid in a transparent way.

[1] This work was partially supported by the following grants: EU DataGrid IST-2000-25182, EU GridLab IST-2001-32133, Hungarian Scientific Research Fund (OTKA) no. T032226, Hungarian OMFB-02307/2000, and Hungarian IKTA-00075/2001.

2 Current Version of P-GRADE

P-GRADE supports the interactive development of a parallel program as well as their job execution mode. The interactive execution can be on a single processor system, on a supercomputer or on a cluster. The recommendation is that the editing, compiling and debugging activities should be on a single processor system while mapping, and performance monitoring should take place on parallel systems like supercomputers or clusters. If the program is correct and delivers the expected performance on parallel systems, the user can switch to the job mode. Here the user should specify the resource requirements, input files, output files and error file of the job. Then P-GRADE automatically generates the appropriate job from the parallel program developed in the interactive working mode. Currently two job types can be generated:

- Condor job
- PERL-GRID job

3 Condor Job Mode of P-GRADE

By integrating P-GRADE and Condor our goal was to provide a Grid-enabled runtime system for P-GRADE. In Condor mode, P-GRADE automatically constructs the necessary job description file containing the resource requirements of the parallel job. The mapping function of P-GRADE was changed according to the Condor needs. In Condor the user can define machine classes from which Condor can reserve as many machines as it is defined by the user in the job description file. When the user generates Condor job under P-GRADE this is the only Condor-related task. P-GRADE supports this activity by offering a default set of machine classes for the user.

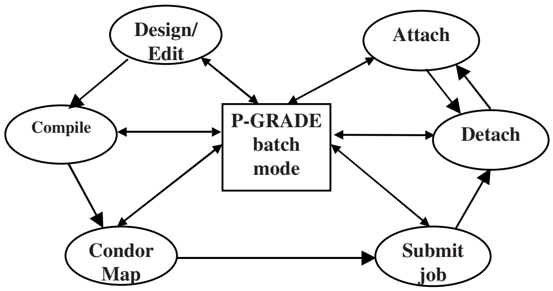


Fig. 1. P-GRADE services in job mode

Finally, after submitting the Condor job the user can detach P-GRADE from the job. It means that the job does not need the supervision of P-GRADE when it is executed in the Grid (for that purpose we use Condor). Meanwhile the P-GRADE generated

Condor job is running in the Grid, P-GRADE can be turned off or it can be used for developing other parallel applications. However, at any time the user can attach again P-GRADE to the job to watch the current status and results of the job. The program development and execution mechanism of P-GRADE for the Grid is shown in Fig. 1.

4 PERL-GRID Job Mode of P-GRADE

The Condor job mode can be used only if the submit machine is part of a Condor pool. However, such a restriction is too strong in the Grid where there are many Condor pools as potential resources and these should be accessed from the submit machine. This is the case, for example, in the Hungarian ClusterGrid where the Grid itself is a collection of Condor pools (clusters) but the user's submit machine is typically not part of the ClusterGrid.

In such situation some new functionalities should be provided for the user. In order to provide these missing functionalities we have created a thin layer, called PERL-GRID, between P-GRADE and Condor with the following tasks:

- The selection of the high-performance computing site (Condor pool) is the task of a Grid resource broker. Currently a random selection algorithm is applied by PERL-GRID. (This will be replaced by a Grid resource broker in the future.)
- PERL-GRID takes care of contacting the selected site and executing the mutual authentication based on the SSH technology.
- PERL-GRID also takes care of file staging by transferring the input files and the code file into a temporary directory at the reserved Grid site. Finally, it passes the job to the local job manager. Currently it is Condor but soon others, like SGE will be supported.
- If the selected Grid site becomes overloaded, the whole parallel application is check-pointed and migrated to another Grid site by PERL-GRID.

An automatic parallel check-point mechanism has been elaborated for parallel programs developed under P-GRADE. By this check-point mechanism, all those features of Condor that are provided for sequential jobs in the Standard Universe (job migration, fault-tolerance, guaranteed execution) are supported even for parallel jobs under P-GRADE.

Running a parallel job in the Grid requires the on-line monitoring and steering possibility of that job. Using the Grid resource and job monitoring techniques developed in the GridLab project we can on-line monitor and visualize the status of P-GRADE jobs. More than that, the processes of a parallel application can be monitored by integrating the GRM monitor of P-GRADE either with the GridLab Grid monitor or with the R-GMA monitoring infrastructure of the DataGrid project. In either case using PROVE of P-GRADE enables the on-line visualization of the process interactions of a parallel application running anywhere in the Grid.

5 Description of the Demonstration

All the features of the PERL-GRID job mode of P-GRADE described in the previous section will be demonstrated by two scenarios. During the demonstration we will use a demonstration Grid consisting of 3 clusters: SZTAKI cluster in Budapest, cluster of the Budapest University of Technology and Economics, cluster of the University of Westminster in London.

5.1 Scenario 1

Scenario 1 will demonstrate the Grid execution of the MEANDER nowcast program package of the Hungarian Meteorology Service. The goal of the MEANDER package is to provide ultra-short range (up to 1 hour) weather forecast of dangerous situations like storm and fog on a high resolution regular mesh (10km -> 1km). To achieve this goal members of OMSZ and SZTAKI have jointly parallelised the six most time consuming calculation module of MEANDER by P-GRADE [2].

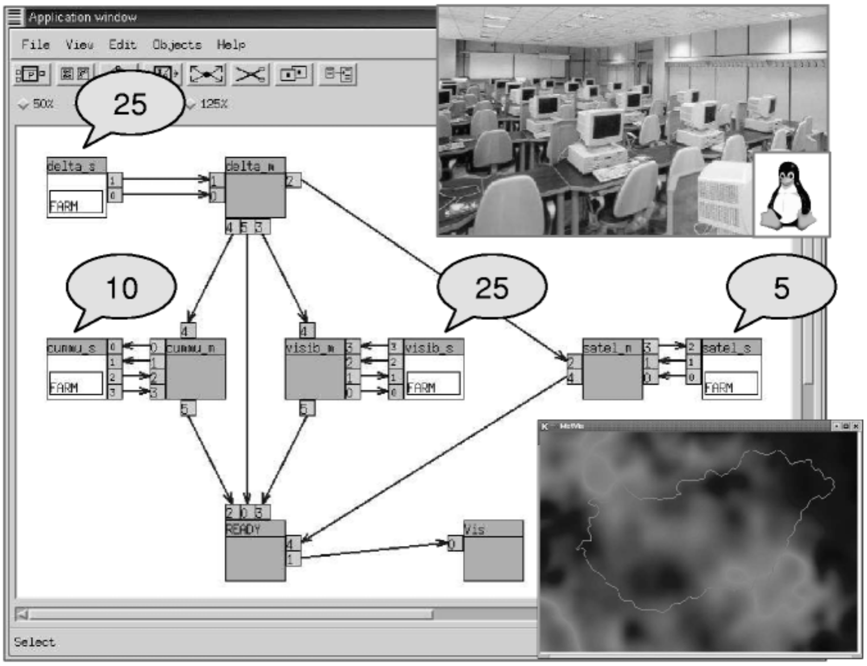
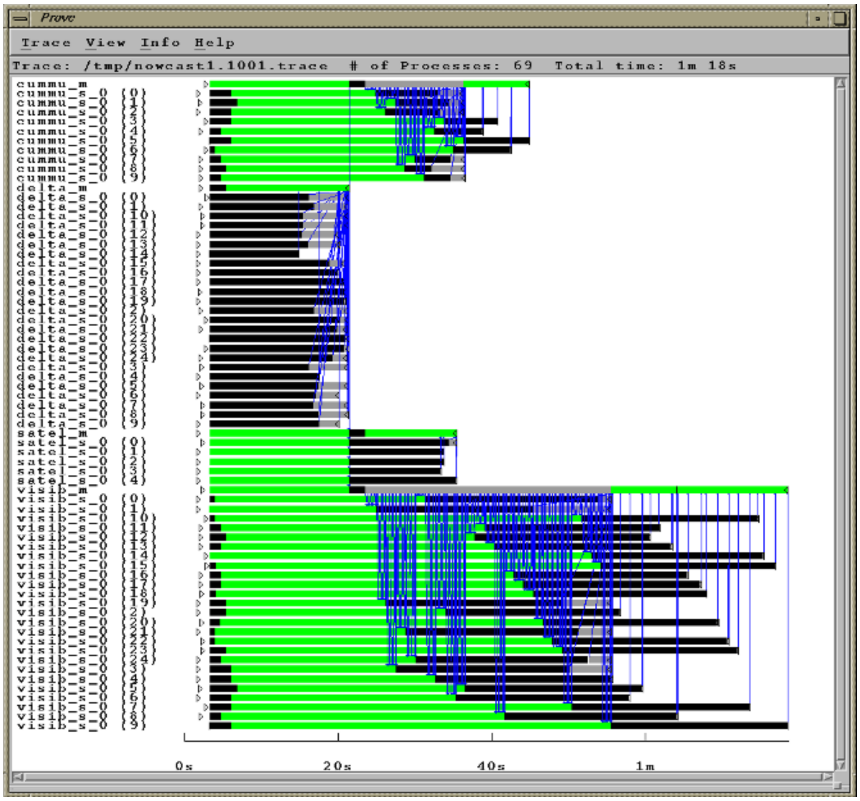


Fig. 2. The P-GRADE demonstration version of the MEANDER program

We will use a simplified version of MEANDER containing only 4 algorithms for the demonstration. The P-GRADE view of this MEANDER version is shown in Fig. 2. It is clear on Fig. 2 that the four algorithms are realized by the processor farm concept. The numbers in the clouds represent the number of processors to be used in the demonstration. It can also be seen that these algorithms are connected like a workflow. First the delta algorithm should be executed, then in the second phase, the other three algorithms can be executed in parallel. At that time 40 processors will be used in parallel. Finally, a visualization process is applied to draw the weather map shown in the right bottom of Fig. 2.

The parallel job will be generated by P-GRADE and passed to PERL-GRID which will transfer the executable code to the 58-processor Linux cluster of SZTAKI (shown in the right top of the picture) to execute the job. Then PERL-GRID collects the necessary meteorology database input file from the Hungarian Meteorology Service and passes the job with all the files to Condor at the SZTAKI cluster. Condor takes care of the parallel execution of the job at SZTAKI. When monitoring is requested, PERL-GRID delivers the local monitor code, too and the collected trace file is sent back and visualized by PROVE on the submit machine at Klagenfurt whenever we request a trace collection by PROVE. The final trace file visualization picture is shown in Fig. 3.



```
Trace: /tmp/nowcast1.1001.trace   # of Processes: 69   Total time: 1m 18s
```

Fig. 3. Process space-time diagram of the MEANDER program

The picture clearly shows that first the delta algorithm is running on 25 processors and when it is finished, it triggers the execution of the other three algorithms that are executed simultaneously. When the whole job is finished, PERL-GRID takes care of transferring the result file back to Klagenfurt and removing the temporary directory it created for the job at the SZTAKI cluster. Finally, the weather-forecast map of Hungary will be displayed at Klagenfurt.

5.2 Scenario 2

Scenario 2 will demonstrate the Grid execution of an urban traffic simulation program developed at the University of Westminster [3]. The program will be started as a PERL-GRID job under P-GRADE. PERL-GRID will select one of the three clusters of the demonstration Grid and transfers the necessary files to the selected cluster where it passes the job to Condor. When the program starts we will increase the load of the selected cluster by starting other higher priority jobs there. The application will be check-pointed and PERL-GRID will select another cluster and migrates the job there and passes it to the local Condor job manager. Using the GridLab Grid monitor system we will on-line monitor and visualize the job status as well as its processes and their interactions by PROVE.

6 Conclusions

P-GRADE provides a high-level graphical environment to develop parallel applications both for parallel systems and the Grid. One of the main advantages of P-GRADE is that the user has not to learn the different APIs for parallel systems and the Grid, simply by using the same environment will result in a parallel application transparently applicable either for supercomputers, clusters or the Grid. The current version of P-GRADE supports the interactive execution of parallel programs as well as the creation of a Condor or PERL-GRID job to execute the parallel program in the Grid. Remote monitoring and performance visualization of Grid applications are also supported by P-GRADE.

P-GRADE will be used in the Hungarian ClusterGrid that connects the Condor pools of the Hungarian higher educational institutions into a high-performance, high-throughput Grid system. Though the Grid execution mode of P-GRADE is currently strongly connected to Condor, the use of PERL-GRID enables the easy connection to other local job managers like SGE and PBS. This work is planned as part of the Hungarian Grid activities.

References

1. P. Kacsuk: Visual Parallel Programming on SGI Machines, Invited paper, Proc. of the SGI Users' Conference, Krakow, Poland, pp. 37–56, 2000
2. R. Lovas, et al: Application of P-GRADE Development Environment in Meteorology, Proc. of DAPSYS'2002, Linz, pp. 30–37, 2002
3. Gourgoulis, et al: Using Clusters for Traffic Simulation, Proc. of Mipro'2003, Opatija, 2003