Unified Development Solution for Cluster and Grid Computing and Its Application in Chemistry*

Róbert Lovas[1], Péter Kacsuk[1], István Lagzi[2], and Tamás Turányi[2]

[1] Computer and Automation Research Institute, Hungarian Academy of Sciences (MTA SZTAKI), 1518 Budapest, P.O. Box 63.
{rlovas,kacsuk}@sztaki.hu
[2] Department of Physical Chemistry, Eötvös University (ELTE), H-1518 Budapest, P.O. Box 32, Hungary
lagzi@vuk.chem.elte.hu, turanyi@garfield.chem.elte.hu

Abstract. P-GRADE programming environment provides high-level graphical support to develop parallel applications transparently for both the parallel systems and the Grid. This paper gives an overview on the parallelisation of a simulation algorithm for chemical reaction-diffusion systems applying P-GRADE environment at all stages of parallel program development cycle including the design, the debugging, the execution, and the performance analysis. The automatic checkpoint mechanism for parallel programs, which supports the migration of parallel jobs between different clusters, together with the application monitoring facilities of P-GRADE enable the long-running parallel jobs to run on various non-dedicated clusters in the Grid while their execution can be visualised on-line for the user. The presented research achievements will be deployed in a chemistry Grid environment for air pollution forecast.

1 Introduction

Computational Grid systems [1] are becoming more and more popular in natural science. In such systems, large number of heterogonous resources can be interconnected in order to solve complex problems. The aim of a joint national project, "Chemistry Grid and its application for air pollution forecast" is to investigate three important aspects of Grids and to find practical results in the areas as follows:

1. Establishment of a chemistry Grid, i.e. using Grid technologies for *supporting a specific scientific research area*; this new infrastructure provides access for chemists to both the Hungarian computational Grid resources

* The research described in this paper has been supported by the following projects and grants: Hungarian IHM 4671/1/2003 project, Hungarian OTKA T042459 and T043770 grants, OTKA Instrument Grant M042110, Hungarian IKTA OMFB-00580/2003, EU-GridLab IST-2001-32133, and COST Action D23/0003/01 project.

A. Laganà et al. (Eds.): ICCSA 2004, LNCS 3044, pp. 226–235, 2004.

(e.g. Hungarian Cluster Grid [2]) and the European-wide chemistry Grid infrastructure being established as a result of the EU funded COST Action D23 project called SIMBEX.

2. The Grid as a *high performance computational infrastructure*; Computer and Automation Research Institute of the Hungarian Academy of Sciences (MTA SZTAKI) has elaborated an integrated parallel program development environment, called P-GRADE [3,4] that supports the parallelisation process of sequential applications in an efficient and clear way by means of its high level graphical approach [5,6] and special correctness and performance debugging and analysing tools [3]. In the chemistry Grid project, we are developing further P-GRADE to provide dedicated support for efficient execution of parallel programs in Grids, such as the migration of applications [4] across the Grid resources according to actual load and availability conditions. The P-GRADE system is currently used by all the participating chemists to parallelise their sequential simulations having high computational demands and afterwards to make them run on the chemistry Grid.

3. The Grid as *a computer system for supporting complex collaborative work* and its application for air pollution forecasting (elaboration of smog alarm plans); the partners will elaborate a collaborative application that will run on a supercomputer to forecast air pollution in Hungary in an operative manner. The same application will run on Grid as well to simulate earlier smog events and to analyse the efficiency of smog alarm plans and the prospective effects of various potential measurements against air pollution.

As a joint effort of MTA SZTAKI and Department of Physical Chemistry, Eötvös University (ELTE), P-GRADE environment has been applied to parallelise an existing simulator for chemical reactions and diffusions in the frame of the chemistry Grid project.
In this paper we introduce briefly the fundamentals of the basic problem of reaction-diffusion systems (see Section 2) and its parallelisation with P-GRADE programming environment in details through the design, debugging (see Section 3), and the performance analysis (see Section 4) phases of program development. Finally, we summarise the related and future works (see Section 5), and our achievements (Section 6).

2 Reaction-Diffusion Equations

A variety of spatiotemporal pattern formation arises from the interaction of chemical reaction and diffusion, such as chemical waves [7], autocatalytic fronts [8], Turing structures [9] and precipitation patterns (Liesegang phenomenon) [10]. Evolution of pattern formation can be described by second-order partial differential equations:

$$\frac{\partial c_i}{\partial t} = \nabla(D_i \nabla c_i) + R_i(c_1, c_2, \ldots, c_n), \quad i = 1, 2, \ldots, n, \tag{1}$$

where c_i is the concentration, D_i is the diffusion coefficient and R_i is the chemical reaction term, respectively, of the ith chemical species, and t is time. The chemical reaction term R_i may contain non-linear terms in c_i. For n chemical species, an n dimensional set of partial differential equations is formed describing the change of concentrations over time and space. These equations are coupled through the non-linear chemical reaction term. Assuming that the diffusion coefficient is constant in space, equation (1) can be rewritten as

$$\frac{\partial c_i}{\partial t} = D_i \nabla^2 c_i + R_i(c_1, c_2, \ldots, c_n), \quad i = 1, 2, \ldots, n, \tag{2}$$

Here

$$\nabla^2 \equiv \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2},$$

where x and y are the spatial variables. The operator splitting approach is applied to equations (2), decoupling transport (diffusion) from chemistry, i.e.

$$c_{i,\hat{t}+\Delta t} = T_D^{\Delta t} T_C^{\Delta t} c_{i,\hat{t}}$$

where T_D and T_C are the diffusion and the chemistry operators, respectively, and $c_{i,\hat{t}+\Delta t}$ and $c_{i,\hat{t}}$ are the concentration of the ith species at time $\hat{t}$ and $\hat{t} + \Delta t$, where Δt is the time step.

The basis of the numerical method for the solution of the diffusion operator is the spatial discretisation of the partial differential equations on a two-dimensional rectangular grid. In these calculations, the grid spacing (h) is uniform in both spatial directions. This approach, known as the 'method of lines', reduces the set of partial differential equations (PDEs) of three independent variables (x, y, t) to a system of ordinary differential equations (ODEs) of one independent variable, time. A second order Runge-Kutta method is used to solve the system of ODEs arising from the discretisation of diffusion with no-flux boundary conditions on a 360×100 grid. The Laplacian is approximated as

$$\nabla^2 c_i^{j,k} \approx L_{c,i}^{j,k} = \frac{1}{6h^2} \sum_{p=-1,q=-1}^{p=1,q=1} a_{p,q} c_i^{i+p,j+q}.$$

The coefficients $a_{p,q}$ are taken according to the matrix below for a nine-point approximation:

$$a_{p,q} = \begin{pmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{pmatrix}$$

resulting in an error of $O(h^2)$ for the Laplacian.

The equations of the chemical term have the form

$$\frac{dc_i}{dt} = R_i(c_1, c_2, \ldots, c_n), \quad i = 1, 2, \ldots, n. \tag{3}$$

The time integration of system (3) is performed with the BDF method using the CVODE package [11,12], which solves stiff chemical kinetics equations. In

the present study, a simple three variable, stiff chemical mechanism was used as proposed by S. C. Cohen and A. C. Hindmarsh [12]. The following parameter set was used in the simulation: $D_1 = D_2 = D_3 = 0.8$. The grid spacing and the time step were $h = 0.4$ and $\Delta t = 0.001$, respectively.

3 Parallel Implementation with P-GRADE

In order to parallelise the sequential code of the presented reaction-diffusion simulation the domain decomposition concept was followed; the two-dimensional grid is partitioned along the x space direction, so the domain is decomposed into horizontal columns. Therefore, the two-dimensional subdomains can be mapped onto a one-dimensional logical grid of processes. An equal partition of subdomains among the processes gives us a well balanced load during the solution of the reaction-diffusion equations. During the calculation of the diffusion of the chemical species communications are required to exchange information on the boundary concentrations between the nearest neighbour subdomains. In the rest of this chapter we illustrate how this idea can be implemented in P-GRADE.

The graphical language of P-GRADE consists of three hierarchical design layers [5]: (i) **Application Layer** is a graphical level, which is used to define the component processes, their communication ports as well as their connecting communication channels. Shortly, the Application Layer serves for describing the interconnection topology of the component processes or process groups (see Figure 1, *Application window*). (ii) **Process Layer** is also a graphical level where different types of graphical blocks are applied: loop construct (see Figure 1, in window labeled *Process: sim* $\rightarrow$ *sim_2*), conditional construct, sequential block, input/output activity block and macrograph block. The graphical blocks can be arranged in a flowchart-like graph to describe the internal structure (i.e. the behaviour) of individual processes (see Figure 1, *Process windows*). (iii) **Text Layer** is used to define those parts of the program that are inherently sequential and hence only pure textual languages like C/C++ or FORTRAN can be applied at the lowest design level. These textual codes are defined inside the sequential blocks of the Process layer (see Figure 1, at bottom of Process window labelled *Process: sim* $\rightarrow$ *sim_1*).

We defined a common process, called 'master' (see Figure 1, *Process: master*), which sets up the initial conditions in a sequential code block, *'init_cond'* and sends the necessary information, e.g. the initial concentrations (A, B and C matrices), the diffusion coefficient (*dc*), the time-step (*dt*) to each 'worker' process via the attached communication port with label '0' using collective communication operations (see the selected communication action icon in Figure 1, *Process: master*). The usage of predefined and scalable process communication templates enables the user to generate complex parallel programs from design patterns. A communication template describes a group of processes, which have a pre-defined regular interconnection topology. P-GRADE provides such communication templates for the most common regular process topologies like process farm, pipe, 2D mesh and tree, which are widely used among scientists. Ports of the member processes in a template are connected automatically based on the topology information. In our case the pipe communication template was selected (see Figure 1,

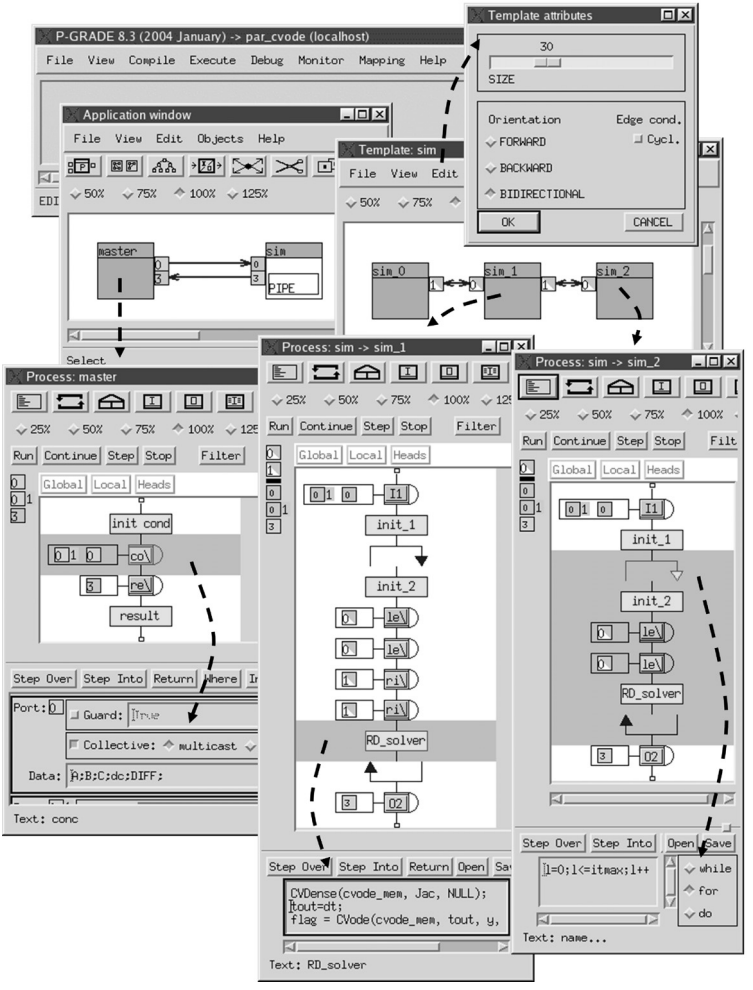

Fig. 1. Parallel code of reaction-diffusion simulation in P-GRADE

Template window) as the most suitable topology since the two-dimensional subdomains should be mapped onto a one-dimensional logical grid of processes and, during the calculation of the diffusion of the chemical species, communications are required to exchange information on the boundary concentrations between the nearest neighbour subdomains. A pipe communication topology consists of a linearly ordered set of processes where each process is interconnected only with its neighbours however, all processes in the pipe may communicate with outsider processes via group ports if such ports are defined for the template at Application level (see Figure 1, *Application window*).

The user must define only the code of the representative processes the number of which depends on the actual template attribute settings (see 'edge condition' below). In a separated dialog window (see Figure 1, *Template Attributes*) the significant attributes of the current template can be set by the user, e.g. in case of pipe:

- *Size*: Actual number of processes within the pipe at runtime.
- *Channel orientation*: Communication channels between neighbour processes can be directed forward, backward or both directions (i.e. bi-directional channels).
- *Edge condition*: Channel pattern can be cyclic if the last process is connected to the first (i.e. ring) one or acyclic otherwise (i.e. pipe).

Without applying that cyclic communication pattern, a pipe is defined by three representative processes since the communication interfaces (i.e. number and types of ports) of the first and last processes differ from that of the middle ones. For illustration purposes we describe only one inner process (see Figure 1, Process window labelled *Process:* $sim \rightarrow sim\_1$). First of all, the process receives the necessary input parameters for the calculation from the 'master' process. After the initialisation phase in each iteration step (applying loop construct) the process exchanges the boundary conditions (a vector of double precision numbers) with its neighbours (see Figure 1, communication actions in window labelled Process: $sim \rightarrow sim\_1$), and calculates the transportation and reaction of chemical species (see Figure 1, sequential code box of RD_solver in window labelled Process: $sim \rightarrow sim\_1$). For the calculation the process invokes external functions (see Figure 1, at the bottom of window labelled Process: $sim \rightarrow sim\_1$), which are available as sequential third-party code [11,12] written in C. Finally, the process sends back the results to the 'master' process, which is responsible for the collection of the results via collective (gather-type) communication. During the debugging stage we took all the advantages of DIWIDE [3] built-in distributed debugger of P-GRADE environment. DIWIDE debugger provides the following fundamental facilities of parallel debugging; the step-by-step execution on both graphical and textual levels, graphical user interface for variable/stack inspection, and for individual controlling of processes.

4 Performance Results

The parallel version of reaction-diffusion simulation with 1000 simulation steps has been tested on two clusters using Condor [13,14] job-mode of P-GRADE [4]: (1) a self-made Linux cluster of MTA SZTAKI containing 29 dual-processor nodes (Pentium III/500MHz) connected via Fast Ethernet, (2) dual mode cluster with 40 nodes (AMD Athlon/2GHz) located at ELTE and connected to the Hungarian Cluster Grid [2]. The simulation has been also tested with 10.000 iterations; the parallel application was able to migrate automatically to another friendly Condor pool when the actual pool had become overloaded, as well as to continue its execution from the stored checkpoint files [4].

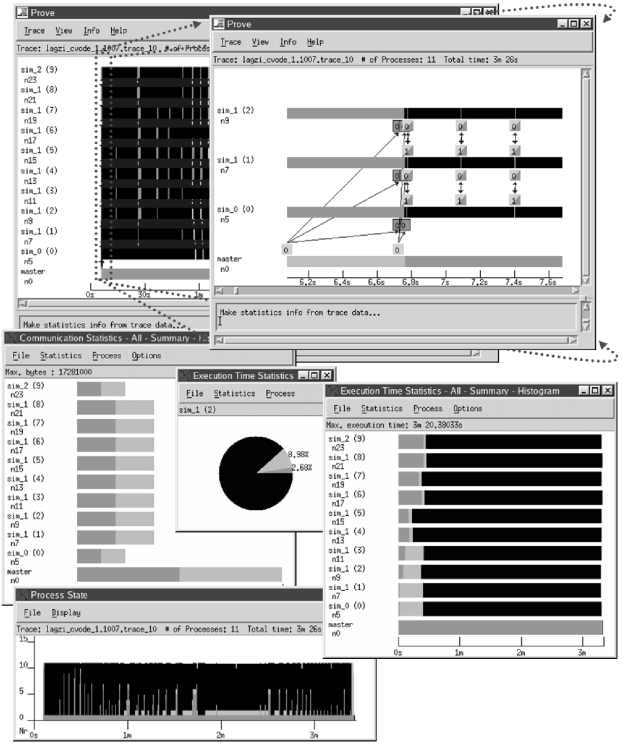

Fig. 2. Performance visualisation with PROVE on MTA SZTAKI's cluster

According to the available computational resources the actual size of the scalable pipe communication topology can be set by a dialog window in P-GRADE. To take an example, the calculation was executed on MTA SZTAKI's cluster within 3 min 26 sec (see Figure 2, PROVE window at the upper-right corner) with 10 worker processes, and it took 1 min 20 sec to calculate it with 40 processes. (The sequential execution time is approximately a half an hour).
In details, PROVE [3] performance analyser (based on the GRM/Mercury monitoring infrastructure [15]) as a built-in tool of P-GRADE system can visualise either event trace data, i.e. message sending/receiving operations, start/end of graphical blocks in a space-time diagram (see Figure 2, PROVE windows), or statistical information about the application behaviour (see Figure 2, Process State, Communication Statistics, and Execution Time Statistics windows). In all the diagrams of PROVE tool, the black colour represents the sequential calculations, and two different colours (green for incoming and grey for outgoing communication) used for marking the message exchanges. The PROVE space-time diagram presents a task bar for each process, and the arcs between the process bars are showing the message passing between the processes. We fo-

cused on some interesting parts of the trace (see Figure 2, PROVE windows) using zooming and filtering facilities of PROVE. The Process 'master' sends the input data to each 'worker' process, and they are starting the simulation and the message exchanges in each simulation steps. The *Process State* window (see Figure 2) is a Gantt chart of the application showing only the state of the processes, sorted by the different types of states. The horizontal axis represent the time while the vertical axis represents the three different states of the processes. For each state, the height of the colored column represents the number of processes in that state. Thus, this graph gives cumulative information about the state of the application. The *Execution Time Statistics* window offers another view; it shows the time of each process state independently for each process in bars or in a pie chart. The *Communication Statistics* (see Figure 2) provide information on the amount of exchanged messages in bytes for each process and for the entire application (see 'Max. bytes'). It is easy to recognise, the first process and the last one in the pipe communicates less (comparing to the other processes of the pipe) since they have only one neighbour. According to our measurements

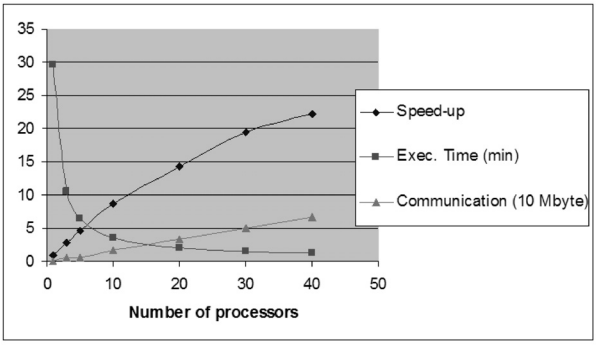

Fig. 3. Performance results on MTA SZTAKI's cluster in Condor job mode

and analysis with PROVE the communication overhead is showing nearly linear characteristics depending on the number of processors, and the curve of speed-up is getting closer and closer to the saturation when the number of processors reaches the 40 (see Figure 3).

5 Related and Future Works

P-GRADE has been applied for the parallelisation of the MEANDER ultra-short range weather prediction package [16] of Hungarian Meteorology Service and for the parallelisation of an urban traffic simulation system [17] by University of Westminster (UK). Institute of Chemistry, Chemical Research Centre of the Hungarian Academy of Sciences has recently parallelised a classical trajectory calculation written in FORTRAN [18] in the frame of joint chemistry

Grid project. Some other development systems, such as ASSIST [19], TRIANA [20], or CACTUS [21], target the same research community (biologist, chemists, etc.), and they can offer several useful facilities similarly to P-GRADE. On the other hand, P-GRADE is able to provide more transparent run-time support for parallel applications without major user interactions, such as code generation to different platforms (Condor [14] or Globus-2 [22] based Grids, PVM [23] or MPI [24] based clusters and supercomputers), migration of parallel jobs across grid sites based on automatic checkpointing facilities [4], or application monitoring of parallel jobs on various grid sites, clusters, or supercomputers [15]. As a new achievement, P-GRADE supports the creation of workflows to execute complex programs in the Grid [6]. In the frame of chemistry Grid project, the developed parallel applications, such as the presented simulation program, will be available via P-GRADE portal, where each component job will collaborate in the Grid based on our workflow concept to provide efficient air pollution forecasting (elaboration of smog alarm plans).

6 Summary

MTA SZTAKI developed a graphical programming environment, called P-GRADE that is able to support the entire life-cycle of parallel program development and the execution of parallel applications both for parallel systems and the Grid. One of the main advantages of P-GRADE is the transparency; P-GRADE users has not to learn the different programming methodologies for various parallel systems and the Grid, the same environment is applicable either for supercomputers, clusters or the Grid. As the presented work illustrates, P-GRADE enables fast parallelisation of sequential programs written in C, C++ or FORTRAN, and it provides an easy-to-use solution even for non-specialist programmers, like chemists, meteorologists, biologists, etc. P-GRADE can generate either PVM [23] or MPI [24] code that can be executed interactively on supercomputers and clusters, or as a job in Condor [14] or Globus-2 [22] based Grids. Moreover, remote monitoring, performance visualisation, fully automatic checkpointing, and migration mechanisms of Grid applications [4] are also supported by P-GRADE.

References

1. Foster, I., Kesselman, C.: Computational Grids, Chapter 2 of The Grid: Blueprint for a New Computing Infrastructure. Morgan-Kaufman, (1999)
2. Stefán, P.: The Hungarian ClusterGrid Project Proc. of MIPRO'2003, Opatija
3. P-GRADE Graphical Parallel Program Development Environment: http://www.lpds.sztaki.hu/projects/pgrade
4. Kacsuk, P., Dózsa, G., Kovács, J., Lovas, R., Podhorszki, N., Balaton, Z., Gombás, G.: P-GRADE: a Grid Programming Environment. Journal of Grid Computing (2004) (accepted for publication)

5. Kacsuk, P., Dózsa, G., Lovas, R.: The GRADE Graphical Parallel Programming Environment Parallel Program Development for Cluster Computing: Methodology, Tools and Integrated Environments (Chapter 10), Editors: P. Kacsuk, J.C. Cunha and S.C. Winter, pp. 231-247, Nova Science Publishers New York, 2001
6. Lovas, R., Dózsa, G., Kacsuk, P., Podhorszki, N., Drótos, D.: Workflow Support for Complex Grid Applications: Integrated and Portal Solutions. Proceedings of 2nd European Across Grids Conference, Nicosia, Cyprus, 2004 (accepted for publication)
7. Zaikin, A. N., Zhabotinsky, A. M.: Concentration wave propagation in two-dimensional liquid-phase self-oscillating system. Nature 225 (1970) 535–537
8. Luther, R.: Raumliche fortpflanzung chemischer reaktionen. Zeitschrift für Elektrochemie 12 (1906) 596–600
9. Turing, A. M.: The chemical basis of morphogenesis. Philosophical Transactions of the Royal Society of London series B 327 (1952) 37–72
10. Liesegang, R. E.: Üeber einige eigenschaften von gallerten. Naturwissenschaflichee Wochenschrift 11 (1896) 353–362
11. Brown, P. N., Byrne, G. D., Hindmarsh, A. C.: Vode: A variable coefficient ode solver. SIAM Journal of Scientific and Statistical Computing 10 (1989) 1038–1051
12. Cohen, S. C., Hindmarsh, A. C.: CVODE User Guide, Lawrence Livermore National Laboratory technical report UCRL-MA-118618 SIAM Journal of Scientific and Statistical Computing (1994) pp. 97
13. Litzkov, M. J., Livny, M., Mutka, M. W.: Condor – A hunter of idle workstations. Proceedings of the 8th IEEE International Conference on Distributed Computing Systems, pp. 104–111, 1988
14. Thain,D., Tannenbaum, T., Livny, M.: Condor and the Grid. in Fran Berman, Anthony J.G. Hey, Geoffrey Fox, editors, Grid Computing: Making The Global Infrastructure a Reality, John Wiley, 2003
15. Balaton, Z., Gombás, G.: Resource and Job Monitoring in the Grid Proceedings of EuroPar'2003 Conference, Klagenfurt, Austria, pp. 404–411, 2003
16. Lovas, et al.: Application of P-GRADE Development Environment in Meteorology. Proceedings of DAPSYS'2002, Linz, pp. 30–37, 2002
17. Gourgoulis, A., Kacsuk, P., Terstyanszky, G., Winter, S.: Using Clusters for Traffic Simulation. Proceedings of MIPRO'2003, Opatija, 2003
18. Bencsúra, Á., Lendvay, Gy.: Parallelization of reaction dynamics codes using P-GRADE: a case study. ICCSA/MPS 2004, Assisi, Italy (accepted for publication)
19. Vanneschi, M.: The programming model of ASSIST, an environment for parallel and distributed portable applications. Parallel Computing 28 (2002) 1709–1732
20. Taylor, I., et al.: Distributed P2P Computing within Triana: A Galaxy Visualisation Test Case. IPDPS'2003, April 2003
21. Goodale, T., et al.: The Cactus Framework and Toolkit: Design and Applications. 5th International Conference on Vector and Parallel Processin, 2002, pp. 197–227
22. www-unix.globus.org/toolkit/
23. Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, B., Sunderam, V.: PVM: Parallel Virtual Machine – a User's Guide and Tutorial for Network Parallel Computing. MIT Press, Cambridge, MA, 1994.
24. Message Passing Interface Forum, MPI: A Message Passing Interface Standard, 1994