DEVELOPMENT OF A GRID ENABLED CHEMISTRY APPLICATION*

István Lagzi[1], Róbert Lovas[2], Tamás Turányi[1]

[1] *Department of Physical Chemistry, Eötvös University (ELTE)*

lagzi@vuk.chem.elte.hu, turanyi@garfield.chem.elte.hu

[2] *Computer and Automation Research Institute, Hungarian Academy of Sciences (MTA SZTAKI)*

rlovas@sztaki.hu

Abstract P-GRADE development and run-time environment provides high-level graphical support to develop scientific applications and to execute them efficiently on various platforms. This paper gives a short overview on the parallelization of a simulator algorithm for chemical reaction-diffusion systems. Applying the same user environment we present our experiences regarding the execution of this chemistry application on non-dedicated clusters, and in different grid environments.

Keywords: programming environment, grid, cluster, computational chemistry

1. Introduction

Beside the widely applied PC clusters and supercomputers, different computational grid systems [1] are becoming more and more popular among scientists, who want to run their simulations (having high computational and storage demands) as fast as possible. In such grid systems, large number of heterogonous resources can be interconnected in order to solve complex problems.

One of the main aims of a joint national project, *Chemistry Grid and its application for air pollution forecast* is to investigate some aspects of Grids, such as their application as high performance computational infrastructure in chemistry, and to find practical solutions.

*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, and EU-GridLab IST-2001-32133.

Department of Physical Chemistry, Eötvös University (ELTE) applied P-GRADE environment to parallelise an existing sequential 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 (see Section 3). We present our experiences in details regarding the execution and performance of this chemistry application on non-dedicated clusters (see Section 4) taking the advantages of the built-in dynamic load balancer of P-GRADE run-time environment. Finally, its successful execution in Condor and Globus based Grids are also presented (see Section 5).

2. Reaction-diffusion equations

Chemical pattern formation arises due to the coupling of diffusion with chemistry, such as chemical waves [3], autocatalytic fronts [4], Turing structures [5] and precipitation patterns (Liesegang phenomenon) [6]. Evolution of pattern formation can be described by second-order partial differential equations:

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

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. Here ∇ is the Nabla operator.
The operator splitting approach is applied to equations (1), 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. A second order Runge-Kutta method is used to solve the system of ODEs arising from the discretisation of partial differential equations with no-flux boundary conditions

on a 360×100 grid. The Laplacian is calculated using nine-point approximation resulting in an error of $O(h^2)$ for the Laplacian.
The equations of the chemical term have the form

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

The time integration of system (2) is performed with the BDF method using the CVODE package [7, 8], which can solve stiff chemical kinetics equations.

3. Parallel implementation in P-GRADE

In order to parallelise the sequential code of the presented reaction-diffusion simulation the domain decomposition concept was followed;

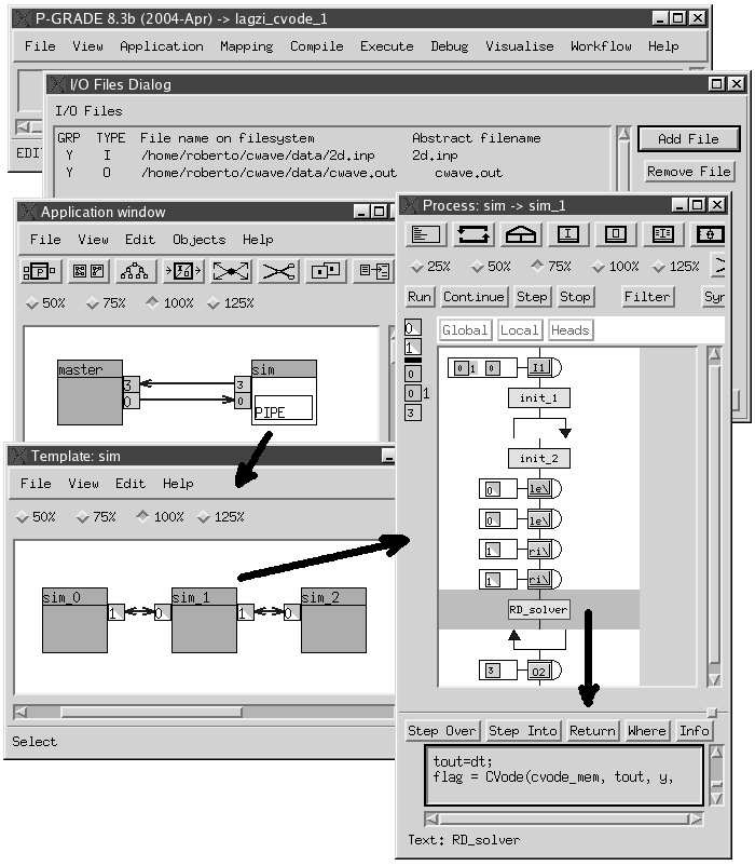

Figure 1. Parallel code of reaction-diffusion simulation in P-GRADE

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 e.g. a pipe of processes (see Figure 1, *Template: sim* window). An equal partition of subdomains among the processes gives us a well balanced load during the solution of the reaction-diffusion equations assuming a dedicated and homogenous cluster or a dedicated supercomputer as the execution platform.
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, which are implemented via communication ports, channels (see Figure 1, *Template: sim* window, arcs between small rectangles), and communication actions (see Figure 1, *Process: sim* → *sim_1*, icons labelled as 'le' and 'ri' in the control flow like description).
For the calculation the process invokes external sequential functions (see Figure 1, bottom of *Process: sim* → *sim_1* windows), which are available as sequential third-party code [7, 8] written in C.
The implementation is published in details in [13].

4. Performance results on non-dedicated cluster

The parallel version of reaction-diffusion simulation has been tested and fine tuned [13] on SZTAKI cluster using it as a dedicated resource. This self-made Linux cluster contains 29 dual-processor nodes (Pentium III/500MHz) connected via Fast Ethernet.
Generally the exclusive access and use of a cluster (e.g. at universities) can not be guaranteed. Sometimes the application is implemented inefficiently, and it may cause unbalanced load (and less effective execution) on the cluster nodes. In both cases the dynamic load balancer [9] of P-GRADE environment can be applied.
In case of the reaction-diffusion simulator the parallel application showed balanced CPU loads [13] on a homogenous and dedicated cluster but we experienced significant slow-down if any of the nodes get an extra calculation intensive task or the node can not deliver the same performance as the other ones. The reason for this phenomenon is that the application must synchronize the boundary conditions at each simulation steps, and they have to wait for the slowest running process. Such situation can be inspected in Figure 2, *Prove* visualisation window when the application was executed on the *n2*, *n3*, *n4*, and *n5* nodes in the first 3 minutes (see the details in Figure 2, smaller *Prove* window in left). The space-time diagram presents a task bar for each process, and the arcs between the

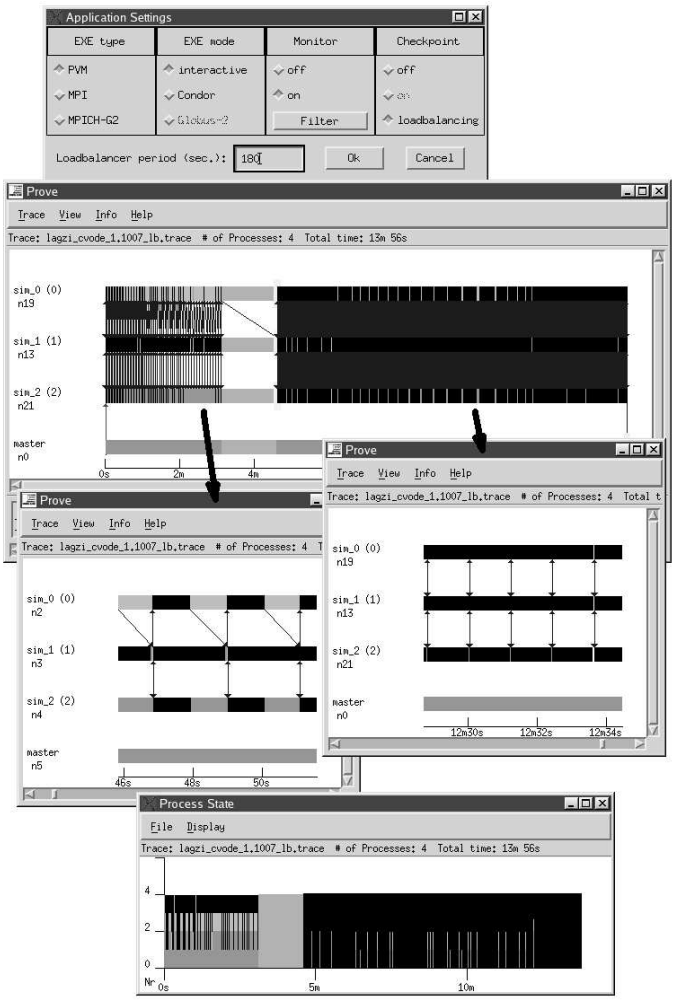

Figure 2. Performance visualisation on non-dedicated cluster

process bars are showing the message passing between the processes. 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.

Thus, we turned on the load balancing support in P-GRADE and recompiled the application under PVM (see Figure 2, *Application settings* dialog window). In our case, the actual period set to *180 sec* when the load balancer has to evaluate the execution conditions based on the gathered information and to make decisions [9].

As the on-line visualisation tool depicts (see Figure 2, *Prove* window) at the beginning of the 4th minute the load balancer initiated the migration of processes to new nodes: *n19*, *n13*, *n21*, and *n0* (see Figure 2, *Prove* window in right). One message was sent before the migration from the node *n2* (process *sim_0*) and delivered just after the migration to the node *n19* (process *sim_1*); the co-ordinated checkpointer in P-GRADE can handle such situations (on-the-fly messages) without any problems. We could focus on the interesting parts of the trace (see Figure 2, smaller *PROVE* windows) using its zooming facilities. 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 represents the time while the vertical axis represents the three different states of the processes. According to this aggregated statistics the application was executed almost optimally from the 5th minute.
The migration took about 1 min and 57 sec due to mainly the large memory images of processes (more than 95 MB/process), that must be transferred from the actual nodes, stored at the checkpoint server, and must be retrieved during the recovery phase of migration on the new nodes. Since the current P-GRADE version launches only one checkpoint server to store these checkpoint files, the network connection of the single checkpoint server may be a serious performance bottle neck. In our case the migration caused almost 800 MB network traffic on the Fast Ethernet network interface of the checkpoint server.
However, the cost of migration is still acceptable since the application continued its execution more than 2 times faster during the remaining calculation; one simulation step needed 1.5-1.7 sec contrary to the earlier measured 3.5-5 sec. Our application needed only 14 minutes (with 500 simulation steps) instead of about 25 minutes without the intervention of load balancer tool. Obviously, with more simulation steps we could get more significant speedup.

5. Performance results in the Grid

The simulation has been also tested with 10.000 iterations [13]; the parallel application was able to migrate automatically to another friendly Condor [10] pool when the actual pool had become overloaded, as well as to continue its execution from the stored checkpoint files [2].
The application has been also executed successfully on Globus [16] based Grid. In order to support the transparent execution of applications on local and remote (interactive or Grid) resources, P-GRADE provides a new I/O file abstraction layer (see Figure 1, *I/O Files Dialog* window),

where the physical data files of the application can be assigned to logical names, which can be referenced in the application by file operations. We defined the input and output files and, in this way, all the necessary I/O files can be automatically transferred to and from the remote site, and the executable can be also staged by P-GRADE run-time system.

Having a valid certificate to deploy a Globus resource (instead of the local resources), the user can turn on the Globus mode with MPI support in P-GRADE (see Figure 3, *Application settings*). Based on the GRM/Mercury monitoring infrastructure [11] the on-line monitoring and visualisation is also possible on Globus resources as well, only a recompilation is needed for the utilization of the Globus/MPI/Monitoring facilities.

The specific Globus resource can be selected in the *Manual Mapping* Window (see Figure 3) by the user, where the entire application will be executed (in MPICH-G2 mode, the processes can be mapped individually to different Globus resources). The monitoring infrastructure provides on-line view similarly to the local execution of job (see Figure 3, *PROVE* window). In the presented case, we executed the 10-process pipe version of the application as a Globus job. The initial time before the real execution and the transfer of output files back (i.e. the 'cost' of

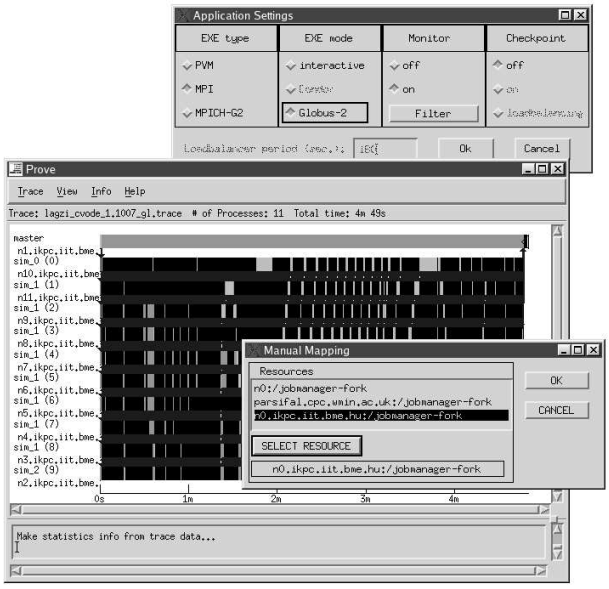

Figure 3. Performance results in Globus mode

Grid based execution from the user's point of view) was within 1 minute because we selected the *fork* job-manager on the Grid site, the cluster was not overloaded, the size of transferred files was relatively small (less then 4MB), and the Hungarian academic network (HBONE) provided high bandwidth between the sites. Generally, this cost is negligible comparing to the typically long (several hours or days) execution time.

6. Related works

P-GRADE has been successfully applied for the parallelisation of different algorithms; e.g. Institute of Chemistry, Chemical Research Centre of the Hungarian Academy of Sciences has recently parallelised a classical trajectory calculation written in FORTRAN [12] in the frame of the joint Chemistry Grid project.
Some other development systems, such as ASSIST [14], or CACTUS [15], 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 [10] or Globus-2 [16] based Grids, PVM or MPI based clusters and supercomputers), migration of parallel jobs across grid sites (or within a cluster) based on automatic checkpointing facilities [2], or application monitoring of parallel jobs [11] on various grid sites, clusters, or supercomputers [11].

7. Summary

P-GRADE 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 [2]. 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 and it provides an easy-to-use solution even for non-specialist parallel and grid application developers, like chemists.

References

[1] Foster, I., Kesselman, C.: Computational Grids, Chapter 2 of The Grid: Blueprint for a New Computing Infrastructure. Morgan-Kaufman, (1999)

[2] Kacsuk, P., Dózsa, G., Kovács, J., Lovas, R., Podhorszki, N., Balaton, Z., Gombás, G.: P-GRADE: a Grid Programming Environment. Journal of Grid Computing Volume 1, Issue 2, 2003, Pages 171 - 197

[3] Zaikin, A. N., Zhabotinsky, A. M.: Concentration wave propagation in two-dimensional liquid-phase self-oscillating system. Nature 225 (1970) 535–537

[4] Luther, R.: Raumliche fortpflanzung chemischer reaktionen. Zeitschrift für Elektrochemie 12 (1906) 596–600

[5] Turing, A. M.: The chemical basis of morphogenesis. Philosophical Transactions of the Royal Society of London series B 327 (1952) 37–72

[6] Liesegang, R. E.: Üeber einige eigenschaften von gallerten. Naturwissenschaflichee Wochenschrift 11 (1896) 353–362

[7] 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

[8] 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

[9] Toth, M., Podhorszki, N., Kacsuk, P.: Load Balancing for P-GRADE Parallel Applications Proceedings of the 4th Austrian-Hungarian Workshop on Distributed and Parallel Systems (DAPSYS 2002), Linz, Austria, pp. 12-20.

[10] 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

[11] Balaton, Z., Gombás, G.: Resource and Job Monitoring in the Grid Proceedings of EuroPar'2003 Conference, Klagenfurt, Austria, pp. 404–411, 2003

[12] Bencsúra, Á., Lendvay, Gy.: Parallelization of reaction dynamics codes using P-GRADE: a case study. ICCSA/MPS 2004, Assisi, Italy (accepted for publication)

[13] Lovas, R., Kacsuk, P., Lagzi, I., Turányi, T.: Unified development solution for cluster and grid computing and its application in chemistry ICCSA/ 2004, Assisi, Italy (accepted for publication)

[14] Vanneschi, M.: The programming model of ASSIST, an environment for parallel and distributed portable applications. Parallel Computing 28 (2002) 1709–1732

[15] Goodale, T., et al.: The Cactus Framework and Toolkit: Design and Applications. 5th International Conference on Vector and Parallel Processin, 2002, pp. 197–227

[16] Foster, I., Kesselman, C.: The Globus Project: A Status Report, Proc. IPPS/SPDP '98 Heterogeneous Computing Workshop, pp. 4-18, 1998.