Development of a grid enabled chemistry application

István Lagzi and Tamás Turányi

Institute of Chemistry,
Eötvös University,
P.O. Box 32, Budapest H-1518, Hungary
E-mail: lagzi@vuk.chem.elte.hu
E-mail: turanyi@elte.hu

Róbert Lovas*

Computer and Automation Research Institute,
Hungarian Academy of Sciences (MTA SZTAKI),
P.O. Box 63, Budapest H-1518, Hungary
E-mail: rlovas@sztaki.hu
*Corresponding author

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 an overview on the parallelisation of two simulator algorithms; for chemical reaction-diffusion systems and for accidental release of chemical (or radioactive) substances. Applying the same user environment we present our experiences regarding the execution of these chemistry applications on dedicated and non-dedicated clusters, and in different grid environments.

Keywords: programming environment; grid; cluster; computational chemistry.

Reference to this paper should be made as follows: Lagzi, I., Turányi, T. and Lovas, R. (2009) 'Development of a grid enabled chemistry application', *Int. J. Computational Science and Engineering*, Vol. 4, No. 3, pp.195–203.

Biographical notes: István Lagzi completed his PhD studies in Physical Chemistry at Eötvös University (2004). He is currently a Postdoc Research Fellow at the Institute of Chemistry. His scientific research fields are numerical simulation in air pollution modelling, adaptive gridding, simulation of reaction-diffusion systems and chemical pattern formation.

Tamás Turányi is a Professor of Chemistry at the Institute of Chemistry of the Eötvös University (ELTE). His degrees are MSc in Chemistry (1983), MSc in Applied Mathematics (1988), PhD in Physical Chemistry (1988), DSc (2004), Dr. habil. (2005). University lecturing: physical chemistry, reaction kinetics, mathematics, and combustion. Fields of research: gas kinetics, air pollution, combustion, and simulation of complex reaction kinetic systems. Participant of several international research projects with partners in the UK, Germany, Italy, Spain, and Mexico.

Róbert Lovas is the Deputy Head of the Laboratory of Parallel and Distributed Systems, Computer and Automation Research Institute of the Hungarian Academy of Sciences, and the Member of the Technical Committee of the Hungarian Grid Competence Center. He received his MSc (1998) and PhD (2006) Degrees at the Budapest University of Technology and Economics. From 1997 he has been involved several national and European research projects as a key developer of P-GRADE programming environment. In the frame of Harness project he worked as a research associate at the Department of Math and Computer Science, Emory University, Atlanta. Fields of research: parallel software engineering and Grid computing tools particularly from design, debugging and application aspects.

1 Introduction

Besides the widely applied PC clusters and supercomputers, different computational grid systems (Foster and Kesselman, 1999) 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 heterogeneous 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 Department of Physical Chemistry (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 fundamental problems 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 dedicated clusters and Condor pools (see Section 4) as well as using non-dedicated clusters (see Section 5) taking the advantages of the built-in dynamic load balancer of P-GRADE run-time environment. Finally, its successful execution on Globus based Grids is also presented (see Section 6) and the ongoing work concerning the simulation of accidental release of chemical (or radioactive) substances (Section 7).

2 Reaction-diffusion equations

Chemical pattern formation arises due to the coupling of diffusion with chemistry, such as chemical waves (Zaikin and Zhabotinsky, 1970), autocatalytic fronts (Luther, 1906), Turing structures (Turing, 1952) and precipitation patterns (Liesegang phenomenon) (Liesegang, 1896). Evolution of pattern formation can be described by second-order partial differential equations:

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

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.

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

$$\frac{\mathrm{d}c_i}{\mathrm{d}t} = R_i(c_1, c_2, \ldots, c_n), \quad i = 1, 2, \ldots, n. \quad (2)$$

The time integration of system (2) is performed with the BDF method using the CVODE package (Brown et al., 1989), 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; 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 (Kacsuk et al., 2001):

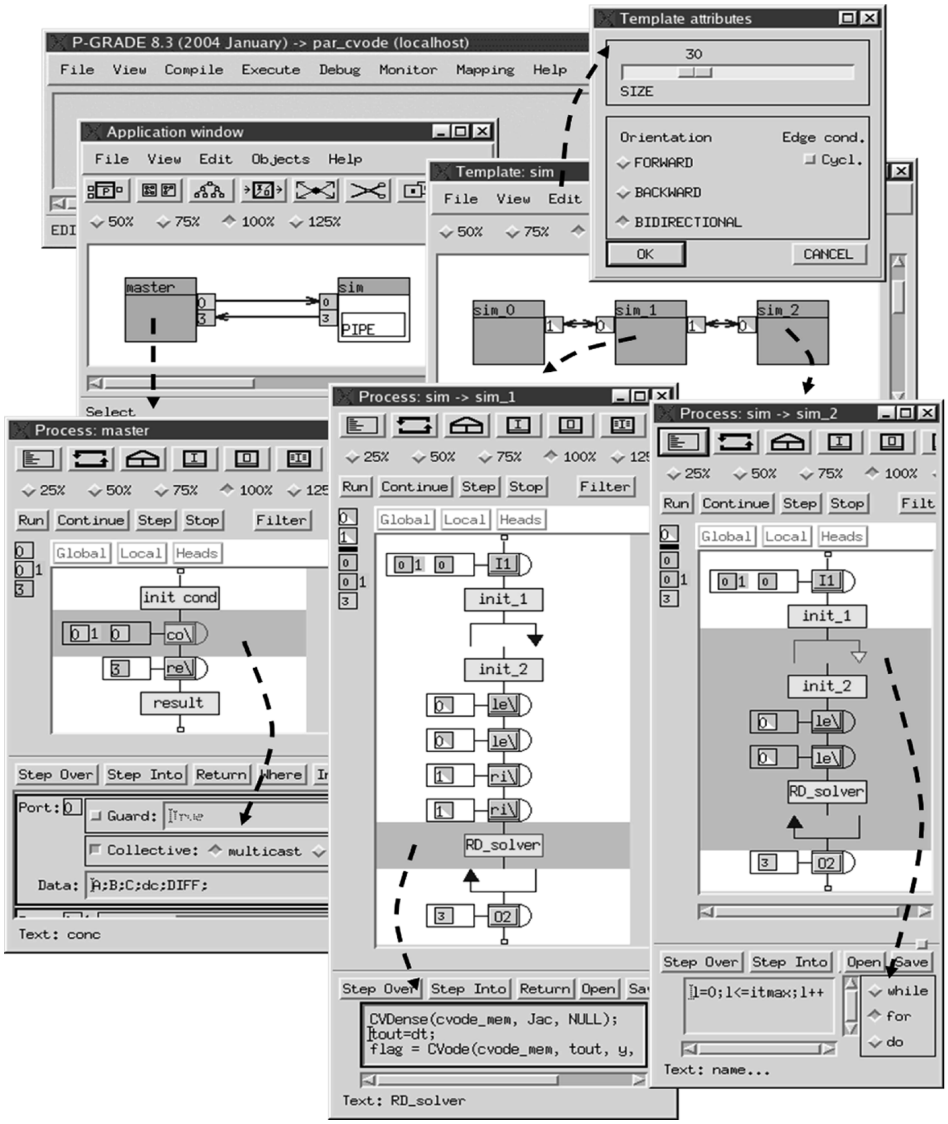
- *Application layer* is a graphical level, which is used to define the component processes, their communication ports as well as their connecting communication channels. Shortly, the Application layer serves for describing the interconnection topology of the component processes or process groups (see Figure 1, *Application window*).
- *Process layer* is also a graphical level where different types of graphical blocks are applied: loop construct (see Figure 1, in window labelled *Process: sim → 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*).
- *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 → 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, *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 has to 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:

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



- *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 otherwise it is acyclic (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 those 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 (Brown et al., 1989) 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 (Kovács and Kacsuk, 2001) 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 on dedicated cluster and condor pools

The parallel version of reaction-diffusion simulation with 1000 simulation steps has been tested on two clusters using Condor (Thain et al., 2003) job-mode of P-GRADE (Kacsuk et al., 2003): on a self-made Linux cluster of MTA SZTAKI containing 29 dual-processor nodes (Pentium III/500 MHz) connected via Fast Ethernet, on a dual mode cluster with 40 nodes (AMD Athlon/2 GHz) located at ELTE. 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 (Kacsuk et al., 2003).

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 s (see Figure 2, PROVE window at the upper-right corner) with 10 worker processes, and it took 1 min 20 s to calculate it with 40 processes (The sequential execution time is approximately a half an hour).

In details, PROVE performance analyser (based on the GRM/Mercury monitoring infrastructure (Balaton and Gombás, 2003)) 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 focused 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 represents the time while the vertical axis represents the three different states of the processes. For each state, the height of the coloured 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 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 40 (see Figure 3).

Figure 2 Performance visualisation with PROVE on MTA SZTAKI cluster

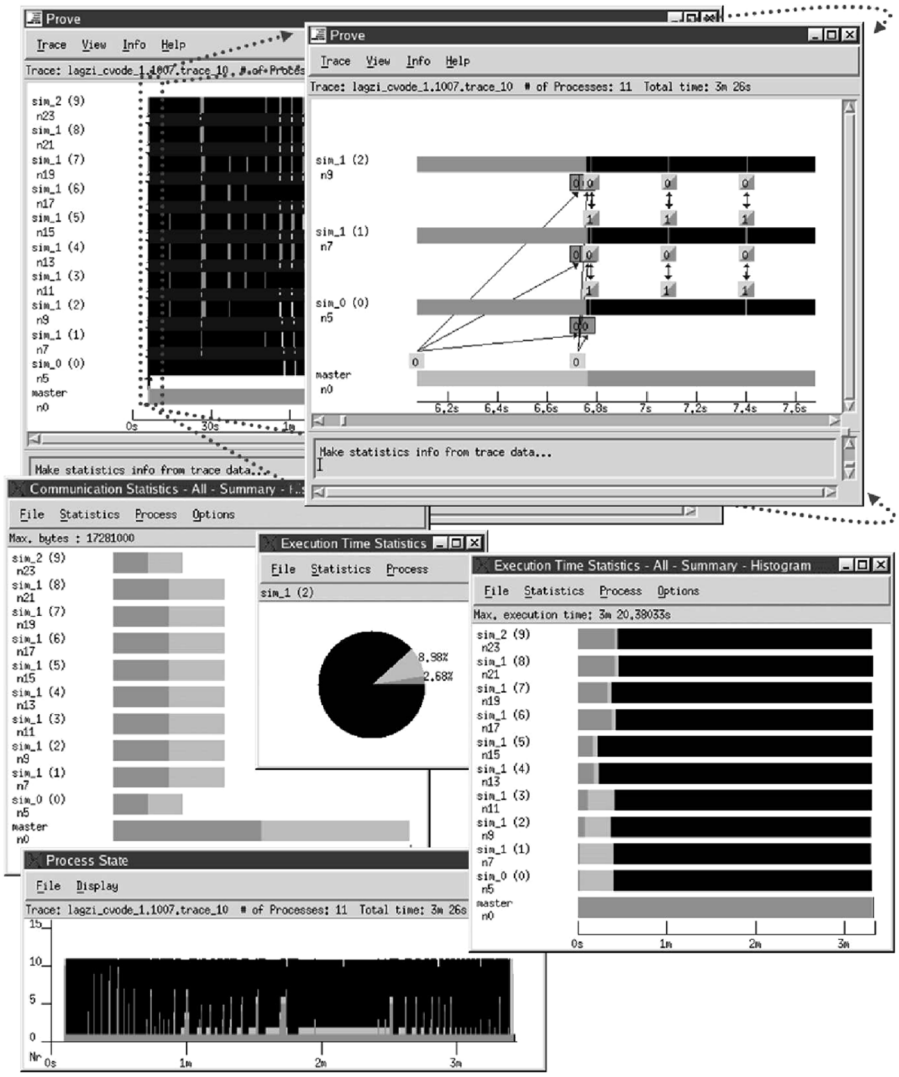
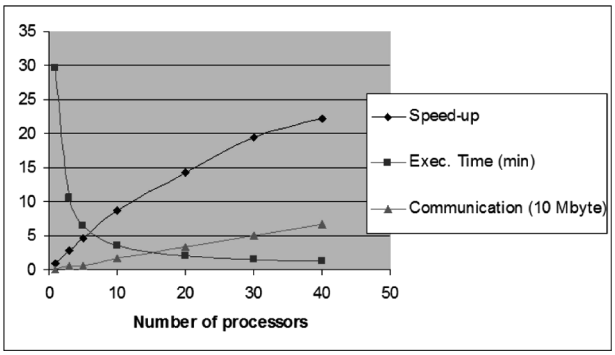


Figure 3 Performance results on MTA SZTAKI's Condor pool



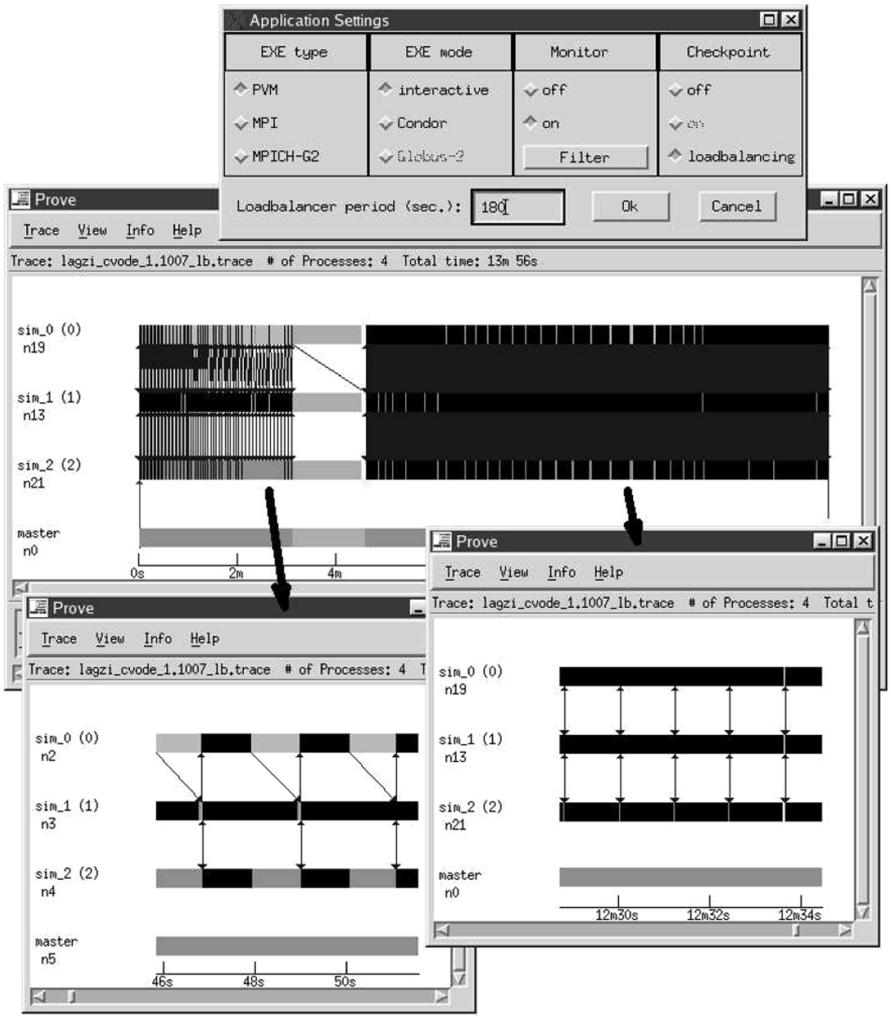
5 Performance results on non-dedicated cluster

Generally, the exclusive access and use of a cluster (e.g., at universities) cannot 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 (Tóth et al., 2002) of P-GRADE environment can be applied.

In case of the reaction-diffusion simulator the parallel application showed balanced CPU loads 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 synchronise the boundary conditions at each simulation steps, and they have to wait for the slowest running process. Such situation can be inspected in Figure 4, *Prove* visualisation window when the application was executed on the *n2*, *n3*, *n4*, and *n5* nodes in the first 3 min (see the details in Figure 4, smaller *Prove* window in left).

Thus, we turned on the load balancing support in P-GRADE and re-compiled the application under PVM (see Figure 4, *Application settings* dialog window). In our case, the actual period was set to 180 s when the load balancer has to evaluate the execution conditions based on the gathered information and to make decisions (Tóth et al., 2002).

Figure 4 Performance visualisation on non-dedicated cluster



As the online visualisation tool depicts (see Figure 4, *Prove* window) at the beginning of the 4th min the load balancer initiated the migration of processes to new nodes: *n19*, *n13*, *n21*, and *n0* (see Figure 4, *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.

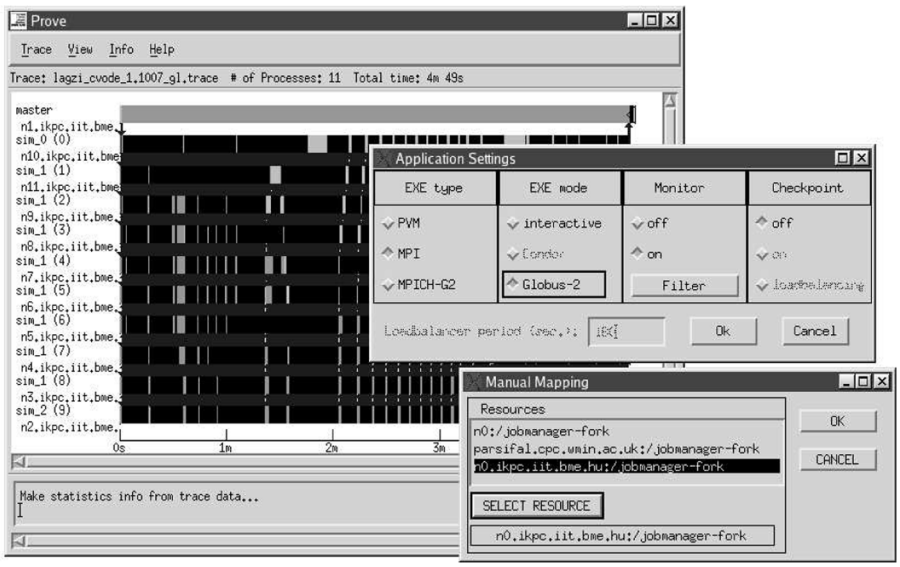
We could focus on the interesting parts of the trace (see Figure 4, smaller *PROVE* windows) using its zooming facilities. According to statistics the application was executed almost optimally from the 5th min. The migration took about 1 min and 57 s due mainly to the large memory images of processes (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 two times faster during the remaining calculation; one simulation step needed 1.5–1.7 s contrary to the earlier measured 3.5–5 s. Our application needed only 14 min (with 500 simulation steps) instead of 25 min without the intervention of load balancer tool. Obviously, with more simulation steps we could get more significant speedup.

6 Performance results in the grid

The application has been also executed successfully on globus (Foster and Kesselman, 1998) 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, 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.

Figure 5 Performance results in Globus mode



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 5, *Application settings*). Based on the GRM/Mercury monitoring infrastructure (Balaton and Gombás, 2003) the online monitoring and visualisation are also possible on Globus resources as well, only a re-compilation is needed for the utilisation of the Globus/MPI/Monitoring facilities.

The specific Globus resource can be selected in the *Manual Mapping* Window (see Figure 5) by the user, where the entire application will be executed (in MPICH-G2 mode, the processes can be mapped individually to different Globus resources but this application showed poor performance in this scenario due to the frequent message exchanges between simulation steps). The monitoring infrastructure provides online view similarly to the local execution of the 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 Grid based execution from the user's point of view) was within 1 min 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 4 MB), 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.

7 Atmospheric dispersion model

Modelling the accidental release of chemical (or radioactive) substances from a single source requires that the numerical simulations must be achieved obviously faster than in real case in order to use them in decision support. A feasible way is the parallelisation of source code. Evolution of chemical species can be described by second-order partial differential equations in 2D layer:

$$\frac{\partial c_i}{\partial t} = K_{x,i}\frac{\partial^2 c_i}{\partial x^2} + K_{y,i}\frac{\partial^2 c_i}{\partial y^2} - u\frac{\partial c_i}{\partial x} - v\frac{\partial c_i}{\partial y} + R_i(c_1, c_2, \ldots, c_n), \quad i = 1, 2, \ldots, n, \tag{3}$$

where c_i is the concentration, $K_{x,i}$, $K_{y,i}$ are the turbulent diffusion coefficients, u, v are the components of the horizontal wind velocity and R_i is the chemical reaction term, respectively, of the ith chemical species. t is time, and x and y are the spatial variables. 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.

In these calculations, the grid spacing is uniform in both spatial directions. The 'method of lines' has been used to reduce 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 the transport terms with chemistry.

The phytotoxic nature of ozone was recognised decades ago. Due to high emissions of ozone precursor substances, elevated ozone concentrations may cover large areas of Europe for shorter (episodic) or longer periods under certain meteorological conditions. These elevated concentrations can be potentially damaging to agricultural and natural vegetation. Occasional extreme concentrations may cause visible injury to vegetation, while long-term exposure, averaged over the growing season, can result in decreased productivity and crop yield. For the computational study of this phenomenon in Hungary, a coupled Eulerian photochemical reaction-transport model and

a detailed ozone dry-deposition model were developed. The Eulerian air pollution model was developed through a co-operation between the Eötvös University, Budapest, The University of Leeds and the Hungarian Meteorological Service. This model fully utilised the experience collected previously at the Leeds University on the use of adaptive gridding methods for modelling chemical transport from multi-scale sources. The model has been elaborated within a flexible framework where both area and point pollution sources can be taken into account, and the chemical transformations can be described by a mechanism of arbitrary complexity. The reaction-diffusion-advection equations relating to air pollution formation, transport and deposition are solved on an unstructured triangular grid. The model domain covers Central Europe including Hungary, which is located at the centre of the domain and is covered by a high-resolution nested grid. The sophisticated dry-deposition model estimates the dry-deposition velocity of ozone by calculating the aerodynamics, the quasi-laminar boundary layer and the canopy resistance. The meteorological data utilised in the model were generated by the ALADIN meso-scale limited-area numerical weather prediction model, which is used by the Hungarian Meteorological Service. For Budapest, the emission inventories for CO, NOx and VOCs were provided by the local authorities with a spatial resolution of 1×1 km and also include the most significant 63 emission point sources. For Hungary, the National Emission Inventory of spatial resolution 20×20 km was applied which included both area and point sources. Outside Hungary, the emission inventory of EMEP for CO, NOx and VOCs was used, having a spatial resolution of 50×50 km.

The work demonstrates that the spatial distribution of ozone concentrations is a less accurate measure of the effective ozone load than the spatial distribution of ozone fluxes. The fluxes obtained show characteristic spatial patterns, which depend on soil moisture, meteorological conditions, ozone concentrations and the underlying land use. The simplified simulation of photochemical air pollution is based on the presented approach (Section 3) as well as the experiences concerning the earlier developed P-GRADE version of ultra-short range weather prediction system. In the final version of the simulator the jobs have not been parallelised due to the unavailable source code of some third-party modules in the model. Thus, we were not able to take advantages of multi-level parallelism, but the workflow level parallelisation was feasbile as it is described in details in Lovas et al. (2005). The P-GRADE portal server is in the centre of air pollution simulation and dedicated to HUNGRID infrastructure. Currently, it provides access to three clusters located at different academic institutes; MTA SZTAKI, KFKI-RMKI, and CRC-HAS. The portal server has access to the meteorological data as well, which are calculated numerically by the Hungarian Meteorological Service based on the available radar and satellite images, the observations, and results of other models. The portal server can be accessed remotely by submitting the simulations, i.e., the P-GRADE workflows, and by downloading the visualisation of execution traces and simulation results on the local machine.

8 Related works

P-GRADE has been successfully applied for the parallelisation of different algorithms; e.g., Institute of Chemistry, Chemical Research Center of the Hungarian Academy of Sciences has recently parallelised a classical trajectory calculation written in FORTRAN (Bencsúra and Lendvay, 2004) in the frame of Chemistry Grid project.

Some other development systems, such as ASSIST (Vanneschi, 2002), or CACTUS (Goodale et al., 2002), 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 Thain et al., 2003 or Globus-2 Foster and Kesselman, 1998 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 (Kacsuk et al., 2003), or application monitoring of parallel jobs (Balaton and Gombás, 2003) on various grid sites, clusters, or supercomputers.

9 Summary

P-GRADE is able to support the entire life-cycle of parallel program development and the execution of parallel applications for both parallel systems and the Grid (Kacsuk et al., 2003). One of the main advantages of P-GRADE is the transparency; P-GRADE users do not need 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 providing an easy-to-use solution even for non-specialist parallel and grid application developers, like chemists (Bencsúra and Lendvay, 2004) or engineers (Gourgoulis et al., 2004).

Acknowledgement

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 T68256 grants, OTKA Instrument Grant M042110, OTKA Postdoctoral Fellowship (D048673), Hungarian IKTA OMFB-00580/2003, and EU-GridLab IST-2001-32133.

References

Balaton, Z. and Gombás, G. (2003) 'Resource and job monitoring in the grid', *Proceedings of EuroPar' Conference*, Klagenfurt, Austria, pp.404–411.

Bencsúra, Á. and Lendvay, Gy. (2004) 'Parallelization of reaction dynamics codes using P-GRADE: a case study', *Lecture Notes in Computer Science*, Vol. 3044, pp.290–299.

Brown, P.N., Byrne, G.D. and Hindmarsh, A.C. (1989) 'Vode: a variable coefficient ode solver', *SIAM Journal of Scientific and Statistical Computing*, Vol. 10, pp.1038–1051.

Foster, I. and Kesselman, C. (1998) 'The globus project: a status report', *Proceedings of the IPPS/SPDP '98 Heterogeneous Computing Workshop*, pp.4–18.

Foster, I. and Kesselman, C. (1999) *Computational Grids, Chapter 2 of The Grid: Blueprint for a New Computing Infrastructure*, Morgan-Kaufman, San Francisco, CA, USA, pp.15–54.

Goodale, T., Allen, G., Lanfermann, G., Massó, J., Radke, T., Seidel, E. and Shalf, J. (2002) 'The cactus framework and toolkit: design and applications', *Proceedings of the 5th International Conference on Vector and Parallel Processing*, Porto, Portugal, pp.197–227.

Gourgoulis, A. *et al.* (2004) 'Creating scalable traffic simulation on clusters', *12th Euromicro Workshop on Parallel, Distributed and Network-Based Processing*, Los Alamitos, USA, pp.60–65.

Kacsuk, P., Dózsa, G. and Lovas, R. (2001) 'The GRADE graphical parallel programming environment', in Kacsuk, P., Cunha, J.C. and Winter, S.C. (Eds.): *Parallel Program Development for Cluster Computing: Methodology, Tools and Integrated Environments*, Chapter 10, Nova Science Publishers, pp.231–247.

Kacsuk, P., Dózsa, G., Kovács, J., Lovas, R., Podhorszki, N., Balaton, Z. and Gombás, G. (2003) 'P-GRADE: a grid programming environment', *Journal of Grid Computing*, Vol. 1, No. 2, pp.171–197.

Kovács, J. and Kacsuk, P. (2001) 'The DIWIDE distributed debugger', *Scalable Computing: Practice and Experience*, Vol. 4, No. 4, pp.5–24.

Liesegang, R.E. (1896) 'Uber einige eigenschaften von gallerten', *Naturwissenschaflichee Wochenschrift*, Vol. 11, pp.353–362.

Lovas, R., Patvarczki, J., Kacsuk, P., Lagzi, I., Turányi, T., Kullmann, L., Haszpra, L., Mészáros, R., Horányi, A., Bencsura, Á. and Lendvay, Gy. (2005) 'Air pollution forecast on the HUNGRID infrastructure', in Joubert, G.R., Nagel, W.E., Peters, F.J., Plata, O., Tirado, P. and Zapata, E. (Eds.): *ParCo 2005. Parallel Computing: Current and Future Issues of High-end Computing*, Malaga, Spain, pp.121–128.

Luther, R. (1906) 'Raumliche fortpflanzung chemischer reaktionen', *Zeitschrift fur Elektrochemie*, Vol. 12, pp.596–600.

Thain, D., Tannenbaum, T. and Livny, M. (2003) 'Condor and the grid', in Berman, F., Hey, A.J.G. and Fox, G. (Eds.): *Grid Computing: Making The Global Infrastructure a Reality*, John Wiley, NJ, USA.

Tóth, M., Podhorszki, N. and Kacsuk, P. (2002) 'Load balancing for P-GRADE parallel applications', *Proceedings of DAPSYS 2002*, Linz, Austria, pp.12–20.

Turing, A.M. (1952) 'The chemical basis of morphogenesis', *Philosophical Transactions of the Royal Society of London Series B*, Vol. 327, pp.37–72.

Vanneschi, M. (2002) 'The programming model of ASSIST, an environment for parallel and distributed portable applications', *Parallel Computing*, Vol. 28, pp.1709–1732.

Zaikin, A.N. and Zhabotinsky, A.M. (1970) 'Concentration wave propagation in two-dimensional liquid-phase self-oscillating system', *Nature*, Vol. 225, pp.535–537.