

VIPER - a tool concept to control parallel CFD applications

M. Lenke

*Lehrstuhl für Rechnertechnik und Rechnerorganisation,
Technische Universität München, D-80290 Munich, Germany*

Abstract

Typical applications in the fields of the so-called *Grand Challenges* need massive parallel computer system architectures. Tools like parallel debuggers, performance analysers and visualizers help the code designer to develop efficient parallel algorithms. Such tools merely support the development cycle. But the engineer who makes use of parallel high-performance computing applications, e.g. numerical simulation algorithms in Computational Fluid Dynamics (CFD), must be supported in his engineering work by another kind of tool. A user tool for the production cycle is required because conventional suggestions rely on serial batch-oriented procedures which have significant drawbacks, and therefore, they are not suitable to make use in nowadays High-Performance Computing (HPC).

This paper presents the principal idea of the VIPER tool concept which enables the engineer who is sitting at his graphic workstation to interact with the parallel simulation algorithm which is running on a remote parallel computer system. Parameters of theoretical model and of the numerical method are considered as objects and available for *online* observation and *online* modification via the graphical user tool interface. First performance results are shown with the VIPER tool prototype implementation applied on the parallel Navier-Stokes solver PARNSFLEX which was running on an Intel Paragon XP/S multiprocessor system.

1 Introduction

Natural and technical processes in physics, chemistry, biology and engi-

180 High-Performance Computing in Engineering

neering can be simulated by calculation of theoretical models which are formulated in mathematical equations. Numerical algorithms of that problem class belong to the Grand Challenges and are characterized by a great complexity of data structures and computations. The successful treatment of the Grand Challenges implies the open of the paradigm *parallelism*. Current research projects are concentrating on the development of parallel processor and system architectures, fast interconnecting network technologies, standardized communication and process models, full supporting operating systems, parallel programming languages, tools for parallel systems and powerfull parallel simulation algorithms.

As a result of those efforts, the parallel code designer is supported by development tools concerning the development cycle, as there are parallel debuggers, performance analysers, visualizers and others. But the engineer must be supported in his engineering work concerning the *production cycle*. Here, the objective is to get conclusions about the theoretical model to acquire know-how about the real-world natural or technical process. Conventional suggestions for the arrangement of the production cycle rely on serial batch-oriented procedures: Here, the production cycle begins with the start of the simulation algorithm on a supercomputer system, usually a vector computer system. Parameters of the theoretical model and of the numerical method are recorded in data files to the end of the simulation run. Both kinds of parameters are necessary to judge the quality of the results. Then, the data files are transfered from the supercomputer to the workstation filesystem where they are processed and displayed via nowadays comfortable and powerful visualization software. After analysis and evaluation of the displayed information the next simulation run can be started with modified parameters.

Such an arrangement of the production cycle is strict batch-oriented. Furthermore, it lacks possibilities to interact with the remote simulation algorithm and to accelerate the production cycle. That is due to the heritage of the traditional work with vector computers. This formative influence is felt in nowadays suggestions for the arrangement of the production cycle in HPC. Unfortunately, only slow progress in acquiring know-how is possible. Additional, some difficulties due to parallel processing have to be overcome: No satisfactory parallel I/O features are available and there is a large amount of data to be handled in distributed data files on distributed filesystems or disksystems.

The quintessence is that there is a necessity to work out new tool concepts for a more suitable arrangement regarding the production cycle of parallel simulation algorithms.

2 The VIPER tool concept

VIPER stands for **V**isualization of **P**arallel simulation algorithms for **E**xten-

ded **R**esearch and is rich in possibilities to interact with the remote simulation algorithm. Figure 1 shows the principal idea of the VIPER tool concept. Parameters of the theoretical model and of the numerical method are formed as objects to spawn the *Object Space*. The tool provides services which enables observations and modifications regarding these objects. The tool process which is running on a graphic workstation is interconnected via LAN (Local Area Network) or WAN (Wide Area Network) with the remote parallel simulation processes which is running on a parallel computer system.

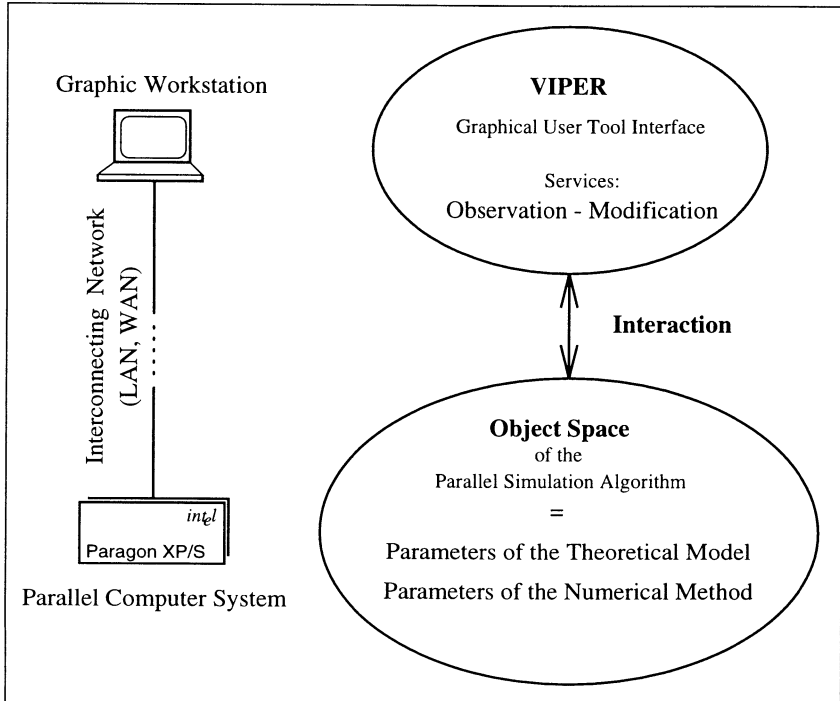


Figure 1: Principal idea of the VIPER tool concept.

The VIPER tool concept is characterized by the following features:

- The engineer is enabled to online observe and modify parameters of the theoretical model and of the numerical method as objects of the *Object Space* via the graphical user tool interface. Thereby, he is enabled to control the running simulation algorithm at any time.
- The engineer has influence on the amount of data to be transferred, by simply activating or deactivating objects.
- The parameters, which are formed as objects, can be displayed in various graphs to be online analysed and evaluated.

182 High-Performance Computing in Engineering

- It is possible to switch between various mathematical methods or computational models.
- The tool VIPER can attach and detach the running parallel simulation at any time.
- The annoying effect of object observation and object modification is reduced to the merely moments of interaction. This ensures a minimum of run time delays for optimized parallel simulation algorithms.
- The simulation algorithm can be based on message-passing or/and shared-variables models.
- The tool concept becomes applicable with all applications by an easy source code instrumentation via library procedures which support the VIPER programming model.
- A strict separation is intended between the code designer who knows the internals of the parallel simulation algorithm and the engineer who knows the engineering aspects of the theoretical model.

The implementation of a prototype in the sense of this tool concept have to be provided by special methods and techniques.

3 The prototype Implementation of VIPER

The realization of the VIPER tool concept is based on a *client-server-client* architecture. Figure 2 shows the server demon *viperd* and both clients the user tool interface of VIPER and the parallel simulation algorithm PARNS-FLEX [BLWM93]. Because the server *viperd* satisfies different requests from two kinds of clients it is named DUAL SERVER.

The VIPER programming model enables the code designer to form parameters of the theoretical model and the numerical methods as objects and define interaction points (IPs) in the source code of the parallel simulation algorithm. An IP ensures consistent object access. *viperd* is instructed to manage all defined objects and their associated interaction points. Objects of this set are switched to an active or inactive state by the engineer via the user tool interface. In the case of interaction demands of the engineer (observation or modification requests) the DUAL SERVER reads/writes directly to the address spaces of the parallel simulation processes.

It is common practice for operating system strategies to prohibit address access into an other process space. But, within the Mach microkernel project this feature is realized. Mach [Loe92] follows a message-oriented operating system concept. Here, all activities are initiated by sending messages to actors via their so-called ports. By means of this mechanism it becomes possible for a process to enable read/write access permission into its own

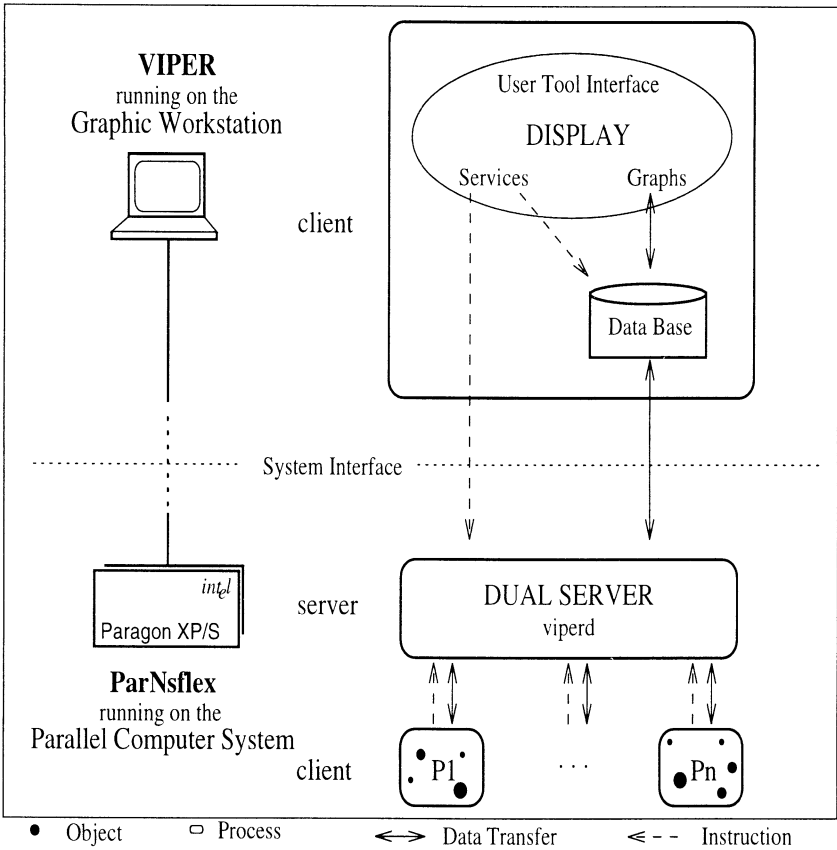


Figure 2: The DUAL SERVER concept *viperd* realized by a client-server-client architecture.

address space to other processes. Such rights are given when the parameters has been declared as objects by using special library procedures. These objects are handled by the DUAL SERVER *viperd*. Therefore, the DUAL SERVER is able to read and write objects of the parallel simulation algorithm directly without any worthmentioning delays (see Table 2).

Data transfers, which are initiated by processing IPs, are realized by the OSI presentation layer (level 6). The XDR protocol [Inc87] which based on Sun RPCs (Remote Procedure Calls, level 5) and TCP/UDP (level 4) makes sure that the data transfers of floats, integers an character, of which representations are the processor architecture dependent, goes off smoothly and as fast as possible. This is important if the processor architecture which are used in the parallel computer systems and the workstation, has an incompatible format.

At the VIPER tool side, incoming and outgoing data are stored and managed in a data base. The *Data Base* allows to adapt existing comfort-

184 High-Performance Computing in Engineering

able and powerful visualization software. In the case of the VIPER prototype a proprietary solution has been chosen which offers simple graphs to visualize objects of the parallel simulation algorithm in various ways. This proprietary solution enables later investigations, e.g. the parallelization of the visualization algorithms.

4 Performance results

Each theoretical tool concept must prove its suitability in practice. The efficiency of the VIPER prototype is depending on three data movement factors which are depending on the used hardware technologies in the main:

- Export of the data (VIPER objects) from the parallel simulation processes.
- Transfer of the data from the filesystem of the remote multiprocessor system to the filesystem of the graphic workstation.
- Visualization of the data on the display.

The test algorithm was the parallelized Navier-Stokes solver PARNS-FLEX [BLWM93]. The test problem was given by the Cast7 profile grid which was partitioned in 32 blocks. A viscous flow type was described with a mach number of 0.7, an angle of attack of 2.0° and a Reynolds number of $6 \cdot 10^6$. The Sun workstation was connected with a Paragon XP/S multiprocessor system via six gateways within a WAN. Numerical convergency can be achieved at about 4500 time step iterations. The source code of PARNS-FLEX was instrumented with one IP. Six objects were associated with that IP which were representing parameters of the physical problem like stress, density or the velocity vectors.

Table 1 shows the amount of data and the elapsed time to export data at the IP from one of 32 the simulation process via *viperd* and via conventional file I/O procedures. To export the amount of data via *viperd* means to be about one thousand times faster.

Table 1: Amount of data and elapsed time

	amount of data	read by <i>viperd</i>	write into ASCII files
per time step	53.696 KB	1.498 ms	1.491 s
500 time steps	26.848 MB	0.758 s	1093.05 s

Table 2 shows the performance influence of *viperd* regarding one of 32 simulation processes. It could be seen that only 1.63 % of the run time was used by *viperd*. During that time 26.848 MB were exported with a transfer rate of 35.41 MB/s.

Table 2: Performance influence of *viperd* regarding one of 32 computation node of the simulation run

	simulation run time (numerical kernel)	<i>viperd</i> (per node)	quota
without <i>viperd</i>	103.64 s	—	—
with <i>viperd</i>	108.080 s	1.770 s	1.63 %
	amount of data	read time	transfer rate
with <i>viperd</i>	26.848 MB	0.758 s	35.41 MB/s

The performance results of the data transfer from the remote multiprocessor system to the workstation regarding one time step iterations of all 32 nodes processes are given in Table 3. The conventional procedure using the Unix service *ftp* and the server/client connection of VIPER within a WAN and LAN are compared.

The size of the data files with 6.662656 MB are much greater than the amount of object data because they are written out in ASCII format. The XDR protocol which is used from the server/client connection have to transform 2.00944 MB (1.718272 MB object data plus 291168 KB data overhead) to a net format which is independent of any processor architecture format. Some more performance improvements could be achieved if no transformation will be necessary.

Table 3: Data transfer results regarding one time step iterations of all 32 node processes (best values)

method	amount of data	elapsed time	transfer rate
conventional procedure (by <i>ftp</i>)	6.662656 MB	201.1	33.1 KB/s
server to client inside a WAN			
	1.718272 MB (+ 291168 KB data overhead)		
with UDP protocol		40.32 s	49.83 KB/s
with TCP protocol		44.91 s	44.74 KB/s
server to client inside a LAN			
with UDP protocol		3.86 s	648.2 KB/s
with TCP protocol		4.34 s	528.8 KB/s

5 Conclusion

If parallelized CFD algorithms should be used in common practice there ought to be develop new tools which support the production cycle with new functionalities. Old strategies are unusable in the long term.

VIPER is a tool concept which encreases the productivity of the production cycle by offering a control mechanism which enables the engineer to observe and modify significant objects of the parallel simulator online.

186 High-Performance Computing in Engineering

These possibilities to interact with the remote parallel simulator accelerate the process of know-how acquisition about the real-world natural or technical process which is simulated by a theoretical model and the numerical algorithm. The VIPER tool concept becomes applicable with all applications by an easy code instrumentation.

This paper shows that great amount of data can be export with high transfer rates. The superiority of the VIPER approach in comparison to traditional file I/O becomes visible. In dependency on the usable network technologies and protocols improvements could be achieved again in comparison to conventional procedure. But, it has to be stressed that the network data transfer is the actual bottleneck. Only improved network technology with much higher transfer rates could take remedial action. The data visualization will not be the time critical procedure with respect to powerful graphic workstations (e.g. Silicon Graphic) with multiple processor units.

Provided that a local network connection to the parallel computer system is had at disposal it could be possible to work with the VIPER prototype on a reasonable performance level in the near future. Furthermore, future prospects for the VIPER tool concept are even much more optimistic with, e.g., ATM (asynchronous transfer mode [Maz94]) network technology.

References

- [BLWM93] A. Bode, M. Lenke, S. Wagner, and T. Michl. Implicit Euler Solver on Alliant FX/2800 and Intel iPSC/860 Multiprocessors. In Ernst Heinrich Hirschel, editor, *Flow Simulation with High-Performance Computers I*, volume 38 of *Notes on Numerical Fluid Mechanics (NNFM)*, pages 41–55. Vieweg, Braunschweig, 1993.
- [Inc87] Sun Microsystem Inc. *XDR: External Data Representation Standard*. RFC 1014, June 1987.
- [Loe92] Keith Loepere. Mach 3 kernel principles. Technical report, Open Software Foundation und Carnegie Mellon University, July 1992. OSF Mach3 series.
- [Maz94] John F. Mazzaferro. *FDDI vs. ATM: High-Speed Networks*. Computer Technology Research Corp., Charleston, South Carolina U.S.A., 1994.