

# An Environment for Generating Applications Involving Remote Manipulation of Parallel Machines

Luciano G. Fagundes, Rodrigo F. Mello, Célio E. Morón

Federal University of São Carlos – Department of Computer Science  
São Carlos - SP - 13565-905 - BRAZIL  
E-mail: {luciano, mello, celio}@dc.ufscar.br

**Abstract.** *This paper summarises the Visual Environment for the Development of Parallel Real-Time Systems and focuses on one of its tools, namely the Graphical User Interface Generator (GUIG). The aim of this tool is to facilitate the generation of a Graphical User Interface for applications developed using the Parallel Kernel Virtuoso (Virtuoso is a trademark of Eonic Systems - <http://www.eonic.com>). Roughly, the GUIG should be able to provide integration among the application being developed, the Internet and the graphical interface of the application. A first version of the Environment was released as Teaching Environment for Virtuoso (TEV), available for downloading from <http://www.dc.ufscar.br/~tev/tev.html>.*

## 1. Introduction

The Visual Environment for the Development of Parallel Real-Time Systems provides an integrated development environment for the construction of parallel real-time systems. This tool has been designed to facilitate the change from the sequential to the parallel paradigm. Parallel programming is being encouraged by the low cost of processors which makes possible the construction of powerful parallel systems, which are able to support high-performance applications. The new challenge is the software complexity and low control of computation. The sequential paradigm has forced inherent parallel problems to be mapped into sequential languages. Tools for sequential programming encapsulate the machine architecture, a feature that is not adequate for parallel systems because the programmer needs to know at least how many processors are available to execute his program. Some works in this direction can be found in [1,2,3,4,5].

The Visual Environment being described here is introduced as an alternative for approaching the problems mentioned above. This environment will be composed by a set of tools to support the final steps involved in the development of parallel real-time programs, one of the tools is the Graphical User Interface Generator.

When designing the Graphical User Interface Generator, the main concern was the communication speed on the Internet. On-line interaction may be spoiled due to the low speed of data transfer across the Internet. This point is being dealt with in two ways: first, we are using specific algorithms to maximise the network usage, and second, studying initiatives like the “Internet2” and the “Next Generation Internet”, which research new technologies to build a new Internet, with speeds ranging from 100 to 1000

times higher than is available today. This tool will initially help in the construction of applications that use local high speed networks and, in a near future, with the increase of network speed it will be possible to create more complex Internet-based systems.

The tools composing the Visual Environment will be integrated through a common graphic interface, and therefore offer a high-level environment for the construction of parallel applications. This paper focuses in the Graphical User Interface Generator. The remaining of this paper is organised as follows: Section 2 describes the tools of the Visual Environment for the Development of Parallel Real-Time Systems; Section 3 discusses the implications of the migration from text-based I/O to Windows-based I/O, as well as the issues involved in the creation of Remote GUIs. Finally, a brief conclusion, that shows the most important issues of this article, is provided.

## **2. A Visual Environment for the Development of Parallel Real-Time Programs**

The main feature of the Visual Environment [6, 7, 8] is to increase the productivity of the development of real-time systems. Visual programming represents an appropriate alternative since the graphic editor provides the programmer with an easy way to generate the main features of his project with the possibility of looking at low level details when necessary. The basic units of visual programming are the objects identified in the Kernel Virtuoso (Virtual Single Processor Programming System) [9], plus the visual components defined by the user. These abstractions help to minimise the maintenance costs while at the same time maximise the reuse of source code.

### **3. From Text-based I/O to Windows-based I/O and Remote Issues**

Parallel machines usually do not have an operational system, they are rather linked to a host computer. The host computer has utilities that enable it to load applications in parallel machines. The interface between parallel computers and the host operating system is composed by those utilities. That interface carries out all input and output operations. The Kernel Virtuoso offers text-based I/O facilities; its utilities are called Host Server.

The Graphic User Interface Generator tool is aimed at supporting the creation of Graphical User Interfaces (GUIs) that will extend the actual architecture of user interaction with the application. It will migrate from text-based to WINDOWS-based systems. The environment described here has been developed for PCs; however, during system development all features emulate a parallel machine. When the user wants to execute in a real parallel machine, the system will provide a cross-compiler that will generate the specific binary code for the specific machine. Currently users need to have a parallel machine available at their facilities, or to use a text-based shell. The aim of the Visual Environment is to generate a system naturally distributed, so that users do not need to have their own parallel machines, but rather execute their programs remotely. It should be highlighted that by using the GUIs Generator, the user is

able to develop all his work in a PC compatible, and he will run his applications in a real parallel machine only when he wants to do so.

### **3.1. Remote GUI Approach**

This tool is aimed at supporting the generation of graphical user interfaces that can interact with remote parallel systems. The main idea behind the tool is to break the system into two distinct modules, one program that is cross compiled for a parallel machine (the server), and the other one that is generated using a windows-based language (the client). Both modules communicate through a network. The Graphical User Interface Generator is quite similar to the Parallel Program Generator [8]; that is, generates the GUIs using visual objects and connections.

The Parallel Programs Generators (PPG) [1,2,3,4,5] will be added with a new object: the GUI. This new object will have a graphical representation and a list of primitives.

At the low level, the GUI Server module is added to the parallel program generated by the PPG; this module is accessed to through an interface composed by C language primitives, in a way analogous to the kernel Virtuoso objects.

The Client module is composed by an external program. The chosen language has been the Java Language, but it could be any other language, provided that it supports network programming and has visual capabilities. The main objective of this is to manage user interaction and to show results from the parallel system.

The network programming level is transparent to the programmer; all connection management and network verification will be encapsulated into GUI primitives. In the GUI Generator it is possible to manipulate user events; however, this can only be done in the host machine, and not in the parallel machine. The interface between the GUI and the parallel machine will be carried out through local variables, which represent the state of the GUI Components (Buttons, Lists, Text Areas, etc). The parallel machine will need to verify the values of those variables and to take decisions about what to do with them.

The Remote GUI approach was chosen mainly because the environment was designed to support real-time systems, in particular those which do not need too much user interaction; that is, the idea is to have a GUI that can interact with the parallel system at run time. The generated real-time parallel system will not be tied up to the GUI, unless the programmer wants to do so. With this flexibility, the GUI will be executed only when an intervention becomes necessary in the parallel machine or when the system needs to obtain data. It should be noted that a parallel system which does not need data for its processing can execute in a stand-alone manner; however, if the programmer wants to generate a GUI, he will have available a user-friendly manner to manage his parallel real-time requirements. All this is possible using the GUI Generator.

The GUI generator is shown through the Parallel Program Generator. Figure 1 shows the GUI Generator. It is composed by a workspace where the GUI assembly happens. The component palette is at the left side; where components like Frames, Buttons, Check Boxes/Check Boxes Groups, Labels, Text Fields/Text Areas, Lists, Choices, Canvases, Scripts, etc are available. In addition, the GUI generator has another important window, the Object Inspector that manages the properties of the visual component.



**Fig 1.** GUI Generator

The GUI assembly is carried out through dragging and dropping visual components, whereas the logic is specified through visual connections, or scripts in Java language.

This approach makes possible to explore two features that may be undermined by programmers of parallel machines, the processing power on the host machine, and the creation of user-friendly interfaces. As mentioned before, the host machine has been used mainly to manage I/O and to load the binary code into the parallel machine. With the creation of local scripts, the programmer gains in processing power which can be used for usual operations that do not need to compete with the tasks in the parallel machine. The parallel machine will be a real extension of the host, and not a system itself.

The GUI state will be verified by the parallel system through local variables. Each component has at least one variable that carries its state. The user can choose whether that information is sufficient or whether he needs a real image of the component.

The component image is interesting for data that is frequently used by the parallel system but one that needs more network usage to update all the data. The user should choose which components need a complete image and which not.

#### **4. Conclusion**

A significant limitation for the development of parallel real-time systems is the lack of adequate programming tools, mainly for supporting the final stages of the development life cycle. CASE tools represent an alternative in this direction, but require development to follow a single methodology from beginning to end. This work presented an environment to help with these problems, and a tool, Graphical User Interface Generator, that allows easy, and

rapid, development of applications that use windows-based I/O and are naturally distributed.

The extensions presented here for the Visual Environment enable the user to create a GUI that allows remote interaction with parallel machines. Using the GUI generator it is possible to build systems that share remote machines using a user-friendly interface instead of the text-based interface currently being used. Finally, it is worth highlighting that the programmer only will need to supply the machine's address, all network programming will be generated by the Visual Environment.

## References

- [1] W. Cai, T.L. Pian, S. J. Turner. "A Framework for Visual Parallel Programming", In Proceedings of Aizu International Symposium on Parallel Algorithms/Architecture Synthesis, IEEE Computer Society Press, Japan, March 1995
- [2] L. Schaefer, C. Scheidler, O. Kraemer-Fuhrmann, "TRAPPER - A Graphical Programming Environment for Industrial High-Performance Applications", Parle (Parallel and Languages Europe), Muenchen, June 1993, pp. 11.
- [3] G. Dózsa, P. Kacsuk, T. Fadgyas, "Development of Graphical Parallel Programs in PVM Environments", In Proc. of 1st Austrian-Hungarian Workshop on Distributed and Parallel Systems, Miskolc, Hungria, October 1996, pp. 33-40.
- [4] M. Aspñäs, R.J.R. Back, T. Långbacka, "Millipede - A Programming Environment Providing Visual Support for Parallel Programming", Reports on Computer Science & Mathematics, Åbo Akademi, Ser. A, N<sup>o</sup> 129, 1991.
- [5] G. R. R. Justo, "PVMGraph: A Graphical Editor for the Design of PVM Programs", Technical Report, University of Westminster, May 1996.
- [6] Célio E. Morón, José R. P. Ribeiro, Nilton C. da Silva, Towards a Rapid Prototyping by Linking Design, Implementation and Debugging in Real Time Parallel Programs *Systems*, Proceedings of the 9th IEEE International Workshop on Rapid System Prototyping, Leuven, Belgium, June 1998.
- [7] Nilton C. da Silva, José R. P. Ribeiro, Célio E. Morón, Sérgio D. Zorzo, Unifying Implementation, Visualisation, Debugging and Validation of Temporal Requirements in Real-Time Parallel Programs, I Workshop on Real-Time System, in the 16<sup>o</sup> Brazilian Symposium of Computer Network, May 1998, (in Portuguese).
- [8] José R. P. Ribeiro, Nilton C. da Silva, Célio E. Morón. *A Visual Environment for the Development of Parallel Real-Time Programs*, published on WPDRS'98 (Workshop on Parallel and Distributed Real-Time Systems), in LNCS # 1388 (Lectures Notes in Computer Science) from Springer-Verlag, Proceedings of the IPPS98 Workshop – IEEE, Orlando, Florida - USA, March, 30 to April, 3 / 1998.
- [9] "Virtuoso - The Virtual Single Processor Programming System", User Manual, Version 3.11, EONIC SYSTEMS.