

# PORTING KHOROS APPLICATIONS TO A PARALLEL VISION MACHINE

**Miguel Pessoa Monteiro**  
Voice: +351.53.604479  
Email: mesmpm@di.uminho.pt

**José Afonso Bulas-Cruz**  
Voice: +351.59.320356  
Email: jcruz@utad.pt

**Alberto José Proença**  
Voice: +351.53.604140  
Email: aproenca@ci.uminho.pt

## Abstract

Khoros applications can be seen as data-flow graphs, which can be specified and executed from Khoros' visual programming environment, Cantata. Many such applications consist of a chain of operators and would clearly benefit from parallel execution, namely pipelines.

This paper presents a methodology to port conventional Khoros applications to a Distributed Memory Multicomputer (DMM) platform, in a transparent way. Pipelined versions of sequences of operators can be automatically generated, loaded and executed on Cantata sessions running on the DMM's host system.

The first target system is a Parsytec MC-3, built around a high speed bus for image data, the TIP-BUS. Each computing node has access to the TIP-BUS and contains two processors: a PowerPC MPC601 for computing tasks and a Transputer T805 for internode communication tasks. The ported applications also support direct input from a frame grabber and direct output to a monitor display controller, useful for video processing.

## 1. Introduction

Khoros is a software environment for data processing, data visualization and software development, with a strong emphasis on Image Processing and Computer Vision (IPCV [1, 2]) applications. Khoros has several features that make it a suitable tool for development of parallel IPCV applications. These include tools for the creation and development of new applications, support for a variety of representation formats, a large repository of already developed IPCV operators, and a visual language, the Cantata front end.

Cantata provides a user-friendly environment where new applications can be created in an intuitive way, with a very short build/test cycle.

Khoros has a world-wide base of users, which is enriching the available repository of Khoros tools and applications. Many of these applications should benefit from the performance enhancements provided by specialized hardware sub-systems. However, since these tasks are still customized and time consuming, many developers choose not to do it. It is desirable to develop a common methodology to port Khoros applications to a large variety of systems, including

reconfigurable hardware devices and add-on systems which support computing parallelism [3].

This communication presents an effort to develop a porting methodology, which was successfully tested on a (Distributed Memory Multicomputer) DMM system. This methodology will later receive further enhancements to support ports to any specific hardware platform that will act as an accelerator - including hardware reconfigurable boards - to speed-up the execution of a Khoros IPCV application.

A new platform for the development and test of real time prototypes of IPCV applications is described. The platform was obtained through the integration of Khoros 2.02 and a DMM system equipped with specific hardware for image acquisition and visualization.

The work presented here aims to provide an automatic tool to speed-up such ports: the individual IP (Image Processing) components of a Khoros IPCV application are automatically assigned to individual computing nodes of a parallel machine. These components are handled as off-the-shelf components, the same way as the original versions from the UNIX/X Window platform. The solutions presented do not pursue the maximum performance enhancements for any specific application. They seek instead to place a broader range of IPCV applications within the reach of parallel systems, by designing a porting methodology that will enable such applications to be transparently reinstalled on a parallel environment. The porting evaluation focuses on a particular variety of parallelism: pipeline of a chain of operators.

## **2. The parallel vision system as a case study**

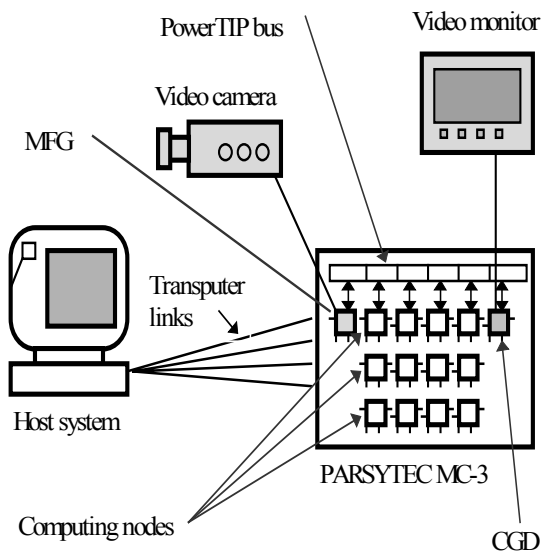
The transputer family of microprocessors became popular as a base for building DMM systems, frequently using the T800 range, although aiming the T9000. However, as the

performance of these chips lagged behind that of more general purpose rivals - such as those from Intel, Digital and Motorola - the design of newer computing nodes required the merge of the communication facilities of the transputer with the greater performance of rival microprocessors.

To overcome these limitations new proposals were implemented based on heterogeneous computing nodes, which include a transputer for the internode communication across the network, and a powerful processor to handle the bulk of the computing workload. The prototype target platform for the ported version of Khoros uses this approach - a Parsytec MC-3 [4]. It is a distributed memory network of computing nodes, where each node contains one transputer and one PowerPC, sharing a central memory.

This system is also equipped with the TIP bus (Transputer Image Processing [5]), a high bandwidth bus for transferring image data. It connects a Monochrome Frame Grabber (MFG [6]) board - for the acquisition of digital images - a Color Graphics Display (CGD [7]) controller - for image display - and four general purpose computing nodes (TPM-MPC [8]). The natural source and destination of data to be processed by the system are the MFG and CGD nodes. Fig. 1 presents the system architecture.

One of the goals of this project was to run Khoros IPCV applications on parallel systems, while using Cantata as the control interface. The applications originate from Khoros' existing repository of IPCV programs, which are to be reinstalled on the target parallel system, in such a way that ported applications would still be configured and started from the Cantata environment, which should be running on the host machine.



**Figure 1**

### 3. Porting methodology

The operator is the basic unit of all applications developed within Khoros. Operators in Khoros are autonomous programs which perform individual IP operations, such as median filtering, gradient calculation, or histogram equalization. Every operator in Khoros is represented within Cantata's visual environment by a graphical icon, called a *glyph*. *Glyphs* can be connected in graphs or chains, to build an application.

The modular nature of Khoros operators make them attractive to build parallel applications. Optimization and parallelization tasks can be appended to a particular operator without risking secondary effects on other operators, provided the operator's standard interface is maintained.

The source code of Khoros operators is supported by a separate software layer, implemented through an extensive set of libraries that duplicate every library function from the usual C programming environment on UNIX platforms. This enables Khoros to isolate its applications from the specific

features of a particular platform, and ensures compatibility across all platforms.

An analysis of Khoros programming libraries revealed that most of its source code can be recompiled for the parallel system without significant modifications. The exception lies on the data services libraries that implement the three data models - polymorphic, geometric and color model - that give Khoros its added functionality, power and flexibility.

One major technical hurdle of current Khoros implementation - on UNIX environments with X Windows - is its reliance on UNIX rich set of services, that the parallel machine often does not support. Khoros 2.02 also lacked tools to provide access to outside systems such as the DMM here described, and its internal representation format proved to be impractical outside of the original environment. These limitations had great influence on the shape of the porting methodology.

The porting tasks consisted mainly in reinstalling or rewriting a subset of Khoros software layer - the programming libraries required by the target operator's source code - into the parallel environment. The reinstallation was done by recompiling the source code of the libraries, with the exception of the data services libraries. The subset of the data services libraries [9] most used by a set of IP operators was re-written to fit into the new environment. Several operators were selected to test this methodology, which made up the first porting case study.

The new version of data services was written on the basis of a new version of the Khoros *kobject* data type, custom-made for IPCV applications on the target system. The new library includes its own versions of every function from data services used in the Khoros operators source code.

An incremental approach was adopted, which addresses the port of each operator one by

one. The port of a Khoros operator can be performed by applying the following steps:

- to recompile the source code of every library function that is not part of data services;
- to create dummies for every data services function whose functionality is not needed for the IPCV applications in the target system;
- to rewrite all data services functions whose functionality is needed for the subset covered by the new *kobject* data type, to run on the target system;

Once these steps are completed, the source code from the target operator can be recompiled and the resulting executables yield the same results in the target system as the original Khoros operators on a standard UNIX platform.

#### **4. Interface between the host and the target parallel system**

The Parsytec MC-3 system is a network of computing nodes interconnected through transputer links. The host system - a Sun Workstation - runs the Cantata interface module and the required cross-compilers, and manages the execution environment, including downloading the application code.

The parallel system has no dedicated I/O system, and shares the I/O system of its host through a server program, the *iserver* 1.5. The *iserver* runs on the host and provides all I/O services to the parallel system, using a transputer link, and loads the code to the network's computing nodes at the start of a session.

The target processes are limited to the services the *iserver* provides. However, the original version of the *iserver* only supports stream (*stdin*, *stdout* and *stderr*) and accesses files in the host's disk: these are unsuitable for transferring large amounts of data - such as image data - between the parallel machine and the host. Extensions were added to the *iserver* to share memory segments [10]

between the processes in the host and the target system. This version of the *iserver* 1.5 - extended with a more adequate mechanism for data transfer - works as an interface between host processes and target processes.

The extended *iserver* is controlled by the target processes, which the *iserver* loads into the target computing nodes. One of these processes requests to the *iserver* the creation of two shared memory segments, one for passing instructions, the other to transfer large data blocks, such as images. As soon as the *iserver* creates these segments, any host process can use them (one process at a time) to exchange messages with the target system, provided its processes are prepared to handle them.

Through this mechanism, images stored as files on the host disk can be transferred to the PowerTIP, processed, and visualized on the video monitor connected by the CGD. Similarly, images grabbed by the MFG can be transferred to the Cantata environment, and visualized there. This mechanism was also used to allow host processes to issue commands to server applications running on the parallel machine.

#### **5. Pipeline control**

One of the tasks of Cantata is to control the execution of Khoros operators, from the specification of an IPCV application. To provide the same control functionality in the ported Khoros operators, new processes were created in the target system. These controller processes use the PowerTIP system and are responsible for the part of Cantata that gets specifications for sequences of operators and starts them afterwards. The specifications includes the values of the operator's arguments, after which the specified sequence runs in a loop.

The controller system was designed to support chains of operators and to assign them to the computing nodes, connected in a pipeline. Each computing node handles a

segment of the chain and represents one stage of the pipeline. A typical pipeline architecture gets its data from the MFG node and the end result is sent to the CGD. The controller system automatically forwards the processed data across the pipeline.

The user can send the specifications for the operators that make up the chain through a new type of Khoros operator, created for this purpose. Its corresponding *glyph* (named a "specification *glyph*") duplicates the interface from the original operator's *glyph*, and adds more arguments. The additional arguments were specifically designed for the configuration of applications on the parallel system. The most important of these arguments is the tag number of the computing node to which the operator is to be assigned. A specification *glyph* must be created for every ported operator.

Cantata stores the information on the chains of operators that make its visual programs in workspace text files. Therefore a workspace processor was created to transparently port applications initially created for Khoros conventional platforms. The workspace processor gets a workspace file and returns a second workspace file. In this second workspace file all references to operators which already have a ported version are replaced by references to their corresponding specification *glyphs*.

When loaded by Cantata, the second workspace is ready to run on the parallel system. When executing, the second workspace causes a series of operator specifications to be sent to the pipeline controller, which activates the corresponding sequence of ported operators in the parallel system.

The workspace processor automatically maps the operators onto the computing nodes, by providing the tag of the computing node as one of the operator's arguments where the operator is to run. Through this mechanism applications originally created for

conventional platforms are automatically converted into pipeline applications for a parallel environment.

## 6. Porting evaluation

A set of operators were selected to evaluate the porting mechanism: *Median*, *Equalize*, *Gradient* and *Thresh Above*. The following files were used:

*Median* operator:

*image/objects/library/kimage\_proc/src/lmedian.c;*

*Equalize* operator:

*datamanip/objects/library/kdatamanip/src/lkhistops.c;*

*Gradient* operator:

*image/objects/library/kimage\_proc/src/ligradient.c;*

*Thresh Above* operator:

*datamanip/objects/library/kdatamanip/src/lkclip.c.*

These files were recompiled without any modifications in the source code, with the exception of *lkhistops.c* and *ligradient.c*, where missing calls to the *kfree()* function were added.

These operators were connected in a chain which resulted on a video processing application for enhanced edge detection. The application was specified from Cantata, running on the host system. The applications run in a loop, using images grabbed by the MFG as inputs, which after processing are displayed on a video monitor controlled by the CGD.

An additional control operator was created for Cantata, with which processed images are displayed on a window, enabling the remote monitoring of applications.

## 7. Summary and conclusions

This communication presents a methodology for porting IPCV operators from Khoros to a parallel machine. It also describes a platform for exploring a repository of ported operators using the same visual environment Cantata as in the conventional Khoros platforms. The proposed methodology is able to automatically extract pipeline parallelism from conventional applications which are made up of a chain of IP operators. An

implementation of this platform in a Parsytec MC-3 machine was developed and successfully tested.

Recent versions of Khoros (namely Khoros 2.1) present a more tractable internal representation format (KDF) than that used by K2 Viff from Khoros 2.02. This improvement should lead to a review of the software infrastructure that enabled the port.

Future work will focus on alternative forms of non-pipeline parallelism, namely algorithmic and geometric parallelism. This work will also take advantage of recently available contributions that give access to standard tools for parallel programming, such as the MPI toolbox.

## References

1. R.C Gonzalez, P.Wintz, *Digital Image Processing*, Addison-Wesley, 1987.
2. D.H.Ballard, C.M.Brown, *Computer Vision*, Prentice Hall, 1982.
3. G.S.Almasi, A. Gottlieb, *Highly Parallel Computing*, Benjamin/Cummings 1994.
4. J.Löwenhag, R.Tekotte, J.Luerken. *MultiCluster-3 Technical Documentation*, Parsytec GmbH, December 1992.
5. PowerStone Parallelrechner *PowerTIP Technical Documentation for Release 1.0*, PowerStone Parallelrechner GmbH, March 1995.
6. N.Schemm. *TIP-MFG Technical Documentation, version 0.9*, Parsytec Computer GmbH, March 1992.
7. K.Bavendiek, N.Schemm. *TIP-CGD - Colour Graphics Display Technical Documentation version 0.9*, Parsytec Computer GmbH, March 1992.
8. Parsytec GmbH. *Parsytec PowerPC Series TIP-MPC Hardware Documentation*, Parsytec Industriesysteme GmbH, November 1994.
9. Khoral Research, Inc. *Programming Services Volume II - polymorphic data services*, 1994.
10. S. Stevens, W. Richard. *Unix Network Programming*, Prentice Hall software series, 1990.