Visual Environment for High Performance Real-Time 3D Reconstruction

Célio E. Morón[1], Lilian N. Faria[1], Nelson D. A. Mascarenhas[1],
José H. Saito[1], Reinaldo R. Rosa[2], Hanumant S. Sawant[2]

[1] Federal University of São Carlos - UFSCar
Computer Department
São Carlos - SP - 13565-905 - Brazil
{celio, lilian, nelson, saito}@dc.ufscar.br
[2] National Institute for Space Research – INPE – LAC/DAS
São José dos Campos - SP - 12201-970 - Brazil
{reinaldo@lac.inpe.br, sawant@das.inpe.br}

Abstract

We are carrying out a development of a whole system for processing solar images in a high performance parallel system. The main objective is to create the initial conditions for studying and forecasting the solar explosions in real time. Due to the high computational costs involved in the processing, visualization and analysis of a great amount of solar images, a high performance computer system becomes necessary to carry out the forecast of solar explosions. As a joint effort between the Department of Computer Science at Federal University of São Carlos (UFSCar), the Astrophysics Division (DAS) and Associated Laboratory for Computing and Applied Mathematics (LAC) at National Institute for Space Research – INPE, a high performance parallel system was developed with capacity to support realistic applications, involving a reasonable amount of parallel processing. The forecast of solar explosions is important as they may cause serious perturbations in terrestrial communication systems. 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. This work presents a development environment, called Visual Environment for the Development of Parallel Real-Time Programs, that supports the design and implementation of parallel real-time applications executed with the support of a parallel kernel. This paper shows how this Environment was used to carry out the 3D Reconstruction of Solar Images.

1. Introduction

With the advance of space technology, solar images with high spatial and temporal resolution, captured by satellites equipped with X-ray telescopes, revealed structures in the solar atmosphere, known as *coronal loops* [5, 6]. Coronal loops (Figure 1) consist of the magnetic field lines that connect magnetic regions on the solar surface, holding a hot moving plasma inside a tube having an arc shape.

The coronal loop can become unstable, and suspicious to be able to cause solar explosion that is nothing more than a sudden release of great amounts of stored magnetic energy in the coronal loop. These explosions emit energetic particles that may cause serious perturbations in terrestrial communication systems, to satellites and energy transportation system. Thus it is important to investigate the dynamics of coronal loops in order to improve the capability of predicting solar explosions.

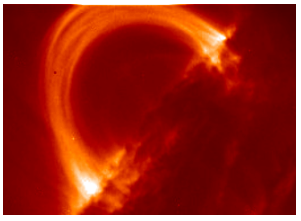

Figure 1 – Coronal loop.

Motivated by the need to forecast these explosions, efforts have been made in order to allow the tridimensional reconstruction of the coronal loop from 2D images obtained through satellites.

The DC/UFSCar acquired an Atlas systems, a parallel machine based on DSPs (Digital Signal Processors),

composed by one host PC Pentium and four DSPs using the Virtuoso real-time operating system.

In order to offer support to the development of parallel programs in the Atlas system using the kernel Virtuoso, a *Visual Environment for the Development of Parallel Real-Time Programs* [2, 4] was developed at the Department of Computer Science at UFSCar. Using this tool, an application for 3D reconstruction of X-ray tomographic images of the solar atmosphere was implemented to execute in a parallel machine using DSPs.

The Visual environment is aimed at supporting the generation of source code for the programs executed in a parallel machine. The applications are built through the construction of a graphical model, which is used to integrate the other tools from the visual environment. This model is represented by a graph, where nodes denote the data structures that compose a parallel program, and arrows denote the communication operations and synchronization between the structures. The information in the graphical model can be complemented with textual descriptions, that is, segments of code written by the user. The application's graphical and textual information is stored in the project's file. Based on this information, the tool automatically generates the source code of the application and the makefiles, which are used for the compilation and linkage of the source code produced.

This paper is organized as follows. In section 2, the 3D reconstruction of coronal loops is presented. Section 3 describes the parallel Atlas system and the Visual Environment for the Development of Real-Time Parallel Programs used for the development of the parallel application. Section 4 describes a parallel 3D reconstruction method of coronal loops. Section 5 presents the results. Finally, Section 6 presents the conclusions.

2. 3D reconstruction of coronal loops

X-ray tomographic images captured by telescopes on board of the Yohkoh satellite show the energy emission of a magnetic loop in two different depths of the solar atmosphere. Figure 2 shows the X-ray images with a resolution of 128x128 pixels, with information of the top and bottom of the loop, respectively.

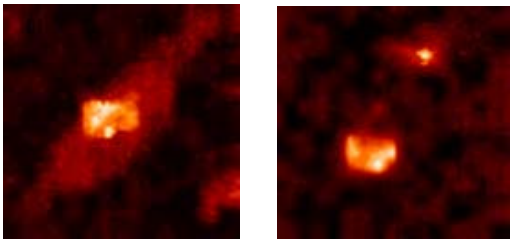

Figure 2. Top and bottom images of the loop observed by the Japanese satellite Yohkoh.

In order to reconstruct the 3D structure of the coronal loop from these X-ray tomographic images, a reconstruction method is used to generate the intermediate images of the loop. However, as there is no information about the tridimensional structure of the loop between the top and bottom images, conventional methods of image interpolation do not reconstruct the magnetic loops, because the shape of the transversal sections between the top and bottom images must gradually be modified to generate the loop.

In order to solve this problem, a reconstruction method was developed to reconstruct the coronal loop using image metamorphosis [8, 9] with a transition function controlled by a Bezier curve in arc shape to generate the intermediate images.

Image metamorphosis (or image morphing) is the gradual transfomation of shape and color of an image into another one [9]. In order to obtain the 3D reconstruction of the loop, the deformation of the top and bottom images is controlled by a Bezier curve that approximates the shape of the loop (Figure 3).

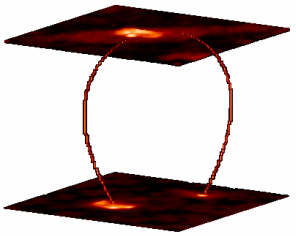

Figure 3. Original images and Bezier curve.

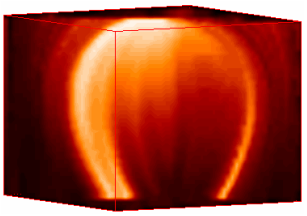

Figure 4. 3D reconstruction of a loop using image morphing with a Bezier curve.

Due to the need of obtaining the 3D reconstruction of the coronal loops within a reasonable time (to estimate and analyze the physical parameters involved in the forecasting of solar explosions), the reconstruction method was implemented to execute in the high performance parallel system described in the next section. Figure 4 shows the final result of the 3D reconstruction of a loop using image morphing with a Bezier curve.

3. Parallel system based on DSPs

A parallel system [3] is being developed by the Department of Computer Science at UFSCar in cooperation with the Astrophysics Division at INPE, aimed at the processing, visualization and analysis, in real-time, of 3D tomographic images of the solar atmosphere.

The ATLAS™ (Figure 5) system [1], includes hardware and software to implement and execute applications that need high performance and digital signal processing. This system is composed by one host PC Pentium with Windows NT, and four processors ADSP-21160 (Hammerhead SHARC™) from Analog Devices. These high performance signal processors are used for communications, graphics, and imaging applications, which combine floating-point operations with multi-processing support.

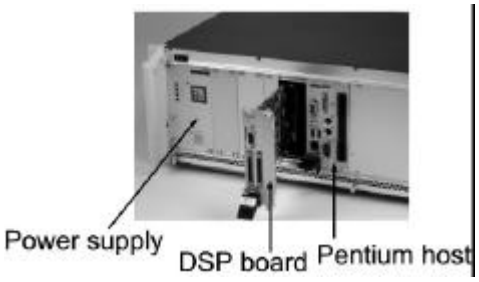

Figure 5. Atlas parallel system.

The ATLAS™ system is shipped together with the fully installed Virtuoso™ real-time operating system (RTOS) from Wind River Systems, Inc.

3.1. Virtuoso™ RTOS

The Virtuoso kernel (Virtual Single Processor Programming System) [7] is a programming tool that offers support for the development of real-time applications.

The applications developed using Virtuoso are divided into tasks; that is, independent program modules that can interact with other tasks through communication and synchronization services. In single-processor systems, Virtuoso uses the multitask concepts in order to simulate simultaneous execution of processes. In multi-processor systems, the tasks can be easily distributed among the different processors, until the real-time requirements are met.

The Virtuoso kernel is an operational system that concentrates only the objects and services necessary for the development of real-time applications in multi-processor systems. Each microkernel object – task, semaphore, resource, and so on – has specific attributes and supports a set of services.

The main microkernel objects are: tasks, semaphores, mailboxes, queues, channels, memory partitions, resources and timers.

Task - A task is a program module that exists to perform a defined function or a set of functions. A task is independent of other tasks but may establish relationships with them.

Semaphore - Semaphores are used to synchronize two tasks and/or events. A signalling task will signal a semaphore while there will be another task waiting on that semaphore to be signalled. One can wait on a semaphore with a time-out or return from the wait if no semaphore is signalled. This can be useful to make sure that the task does not get blocked.

Mailbox - Messages are used between a sender and a receiver task. This is done using a mailbox. The mailbox is used as an intermediate agent that accepts messages. Messages work with arbitrary sizes and allow a selective transport between sender and receiver.

Queue - Queues are also used to transfer data from a task to another one but here the data with fixed size is actually transferred in a buffered and time-ordered way. The advantage is that no further synchronization is required between the enqueuing and the dequeuing task, allowing the enqueuer to carry on.

Channel - A channel consists of queued writer(s) and reader(s) and an optional channel buffer. In the unbuffered case, data with arbitrary size will flow directly from writers to readers. When using the option of channel buffers, data will probably first be copied to the buffer before being finally transferred to the reader. Channels should be thought of as software 'pipes' that allow one task to put data in and another one to take it out. In addition, channels allow communication between a task and an external program.

Memory - In any system, memory is a resource for which tasks compete. Memory management is an area where various techniques can be applied. Many techniques are very fast and use elegant models for allocation and deallocation of memory.

Resource - The resource protection calls are needed to assure that the access to resources is done in an atomic way. Unless the processor can provide real physical protection, the locking and unlocking of a resource is in fact a convention that all tasks using a resource must follow.

Timer - This class of calls allows an application task to use a timer as part of its function. From them on, the timer can be started to generate a timed event at a specified moment (one shot) or interval (cyclic). This event can then signal the associated semaphore. Timers are mainly used to regulate the execution of tasks in relation to a required timely behavior.

3.2. Visual Environment for the Development of Real-Time Parallel Programs

The Visual Environment for the Development of Real-Time Parallel Programs is an integrated programming environment for the development of applications executed in a parallel machine.

In the Visual Environment, applications are built through the construction of a graphical model. This model is represented by a graph, where nodes denote the data structures that compose a parallel program (tasks, signals, resources, mailboxes, etc.), and arrows denote the communication and synchronization operations between the structures.

The graphical notation used by the Visual Environment tries to integrate the three basic components (functional, behavioural and structural) of a parallel real-time program into a single graphical representation [2, 4]. As in most visual environments, this representation is based on graphs composed of nodes (tasks) and arcs (message flow). However the graph is extended in order to represent also the data structures that control the communication and synchronization between the processes (mailboxes, channels, semaphores, etc.).

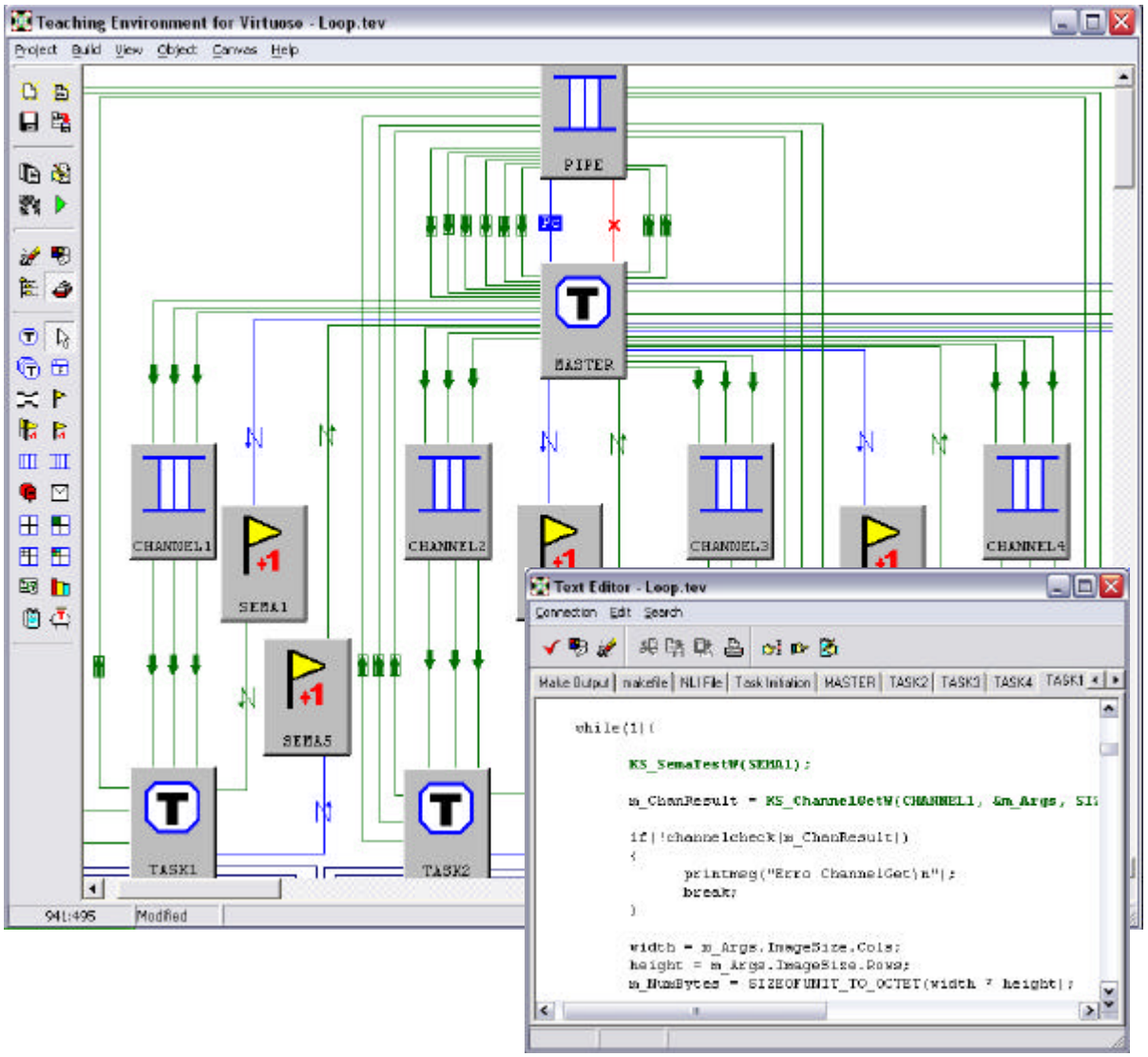

Figure 6. Main interface of the Visual Environment for the Development of Real-Time Parallel Programs

The information in the graphical model can be complemented with code written by the user. Based on this information, the Visual Environment automatically generates the source code of the application.

The programmer defines the microkernel objects – tasks, semaphores, resources, and so on – and distributes these objects among the more convenient processors, until the real-time requirements are met. Figure 6 shows a simplified graphical representation of the parallel program in the Visual Environment

4. Parallel program for 3D reconstruction of coronal loops

The application for 3D reconstruction of coronal loops is composed by a main program running in the host PC of the Atlas parallel system, and a 3D reconstruction parallel program running in the four DSPs. The communication between these programs is carried out through a bi-directional communication pipe.

The 3D reconstruction parallel program, implemented in the programming model based on tasks and channels, is divided into five tasks (MASTER, TASK1, ... , TASK4).

Task MASTER distributes the original images of the loop between the tasks TASK*i*, where each task TASK*i* executes in a different processor, in order to simultaneously generate a subgroup of intermediate images.

The tasks are interconnected by channels (PIPE, CHANNEL1, ... , CHANNEL4). The external channel (PIPE) allows a bi-directional communication between the main program at the host PC and the tasks on the DSPs of the parallel machine. The other channels carry out the inter-processor or intra-processor communication between tasks.

Figure 7 illustrates the data flow of the 3D reconstruction application.

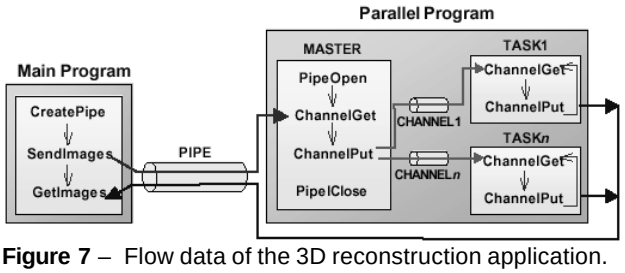

Figure 7 – Flow data of the 3D reconstruction application.

When the main program is executed, a pipe for external communication is created, and the parallel program initiates the execution. Task MASTER opens a connection with the pipe and waits for the main program to send the original images of the loop.

When the user requests a 3D reconstruction, the main program sends the original images for task MASTER, who in turn, distributes these images between the other tasks using the internal channels. Each task TASK*i* waits until receiving the original images to initiate the processing.

During the generation of the intermediate images subgroup, tasks TASK*i* send the intermediate images to the main program through the communication pipe. The main program stores the intermediate images to obtain a volume dataset. After the 3D reconstruction, the tridimensional structure of the coronal loop can be visualized through an interface of volume visualization.

After that, new requests of 3D reconstruction can be executed. Finally, when the main program is finished, the parallel program closes the pipe and ends the execution.

5. Results

The 3D reconstruction parallel program was executed for performance analysis in the parallel machine with four processors. Then we estimate the result for 8, 16, 32, 48 and 64 processors.

Based in these estimates, we noticed that up to 16 processors the efficiency of the program basically keeps constant. However, due to the increasing of overhead and the communication bottleneck, the efficiency tends to decrease and the speedup does not increase linearly as the number of processors is increased (Figure 8 and 9). Nevertheless, the estimated values show that the algorithm is scalable allowing an increase in the number of processors.

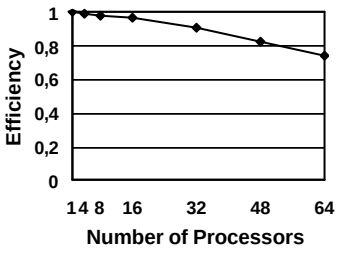

Figure 8 – Efficiency x number of processors.

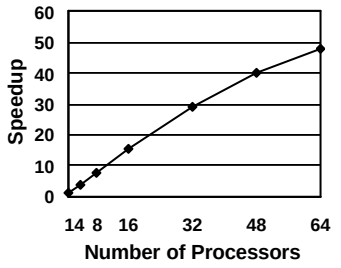

Figure 9 – Speedup x number of processors.

Based on this analysis, it can be concluded that, increasing the number of processors to 32, the same 3D reconstruction can be obtained in about 2.5 seconds. The next step is to begin the work of optimizing our software as we believe that the processing time can be further decreased.

6. Conclusions

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 paper presented a Visual Environment that was used to develop a 3D reconstruction of coronal loops. Due to the need to obtain the 3D reconstruction of the coronal structures in real-time for the forecasting of solar explosions, a parallel program was implemented to execute in the Atlas system with four DSPs. The parallel program was developed using the Visual Environment for the Development of Real-Time Parallel Programs, which offers support for the development of applications using the Virtuoso kernel. The graphical notation used by the Environment integrates the three basic components of a parallel real-time system (behavioural, structural and functional), into a single graphical representation, facilitating therefore the understanding of the system as a whole.

Preliminary results of the scalability analysis show that the number of processors can be increased. The 3D reconstruction using the Atlas system with 4 processors spent 19.28s, with the speedup of 3.93, nearly linear.

The 3D reconstruction parallel program was executed for performance analysis in the parallel machine with four processors. The values for the execution time were measured only for one, two and four processors. For the higher number of processors, the values were calculated by an estimate of the execution time. Our estimation shows that using 32 processors, the same reconstruction can be carried out in 2.5 seconds. We are working now in decreasing this time further.

Acknowledgments

We thank the financial support offered by the three Brazilian government agencies: FAPESP (Foundation for the Support of Research for the State of São Paulo), FINEP (Financing Body for Studies and Projects) and CNPq (National Council for Scientific and Technological Development).

References

[1] *Eonic Atlas System User Guide: Atlas2-HS V1.1*, Eonic Solutions.

[2] C.E. Morón et al., "A visual environment integrating design, implementation and debugging in parallel real-time systems", *12th Brazilian Symp. on Computer Architecture and High Performance Computing*, *SBAC-PAD*, Brazil, Oct. 2000.

[3] C.E. Morón et al., "Parallel Architecture using DSPs", *Proc. 9th Brazilian Symp. on Computer Architecture and High Performance Computing*, *SBAC-PAD'97*, Brazil, 1997, pp. 605-608.

[4] J.R.P. Ribeiro, N.C. da Silva, and C.E. Morón, "A Visual Environment for the Development of Parallel Real-Time Programs", *Lecture Notes in Computer Science*, vol. 1388, 1998, pp. 994-1014.

[5] S. Tsuneta and J.R. Lemen, "Dynamics of the solar coronal observed with the Yohkoh soft X-ray telescope", *Physics of Solar and Stellar Coronae*, 1993, pp. 113-130.

[6] Y. Uchida, "New Aspects of Solar Coronal Physics Revealed by Yohkoh", *Physics of Solar and Stellar Coronae*, 1993, pp. 97-112.

[7] *Virtuoso™ User Guide for Version 4.2*, Wind River Systems, 2001.

[8] G. Wolberg, *Digital Image Warping*, IEEE CS Press, Los Alamitos, CA, 1990.

[9] G. Wolberg, *Image morphing: a survey*. The Visual Computer Journal, 14, pp. 360-372, 1998.