Three-Dimensional Visualization of Dynamic Processes in Active Programs

Bernhard Reitinger
breiting@gup.uni-linz.ac.at

GUP Linz, Joh. Kepler University Linz
Altenbergerstr. 69, A-4040 Linz, Austria

Abstract

Online visualization of active programs is used to increase a user's understanding of the dynamic processes manifesting a program's behavior. The novel idea presented in this paper applies Virtual Reality techniques to display executing parallel applications. For that reason, the memory of a program is mapped onto different representation, which may even be supported by corresponding sounds, in order to allow the user to feel the program. This allows to distinguish between areas of high activity and non-active parts, which is useful for both, error detection and performance tuning.

KEYWORDS: virtual, reality, immersion, visualization, program, auralization

1 Introduction

Understanding a program's behavior is the central aspect of program analysis, which is applied for performance tuning, error detection, or for monitoring & steering. Simplified, the user's goal is to look inside the computer's memory and see what the program is doing.
Thus, it is of major importance to select a suitable program representation for visualizing the program's behavior and it's activities during execution.

In this paper we present a novel idea to the visual program behavior. Our approach was inspired by the fiction novel "Tracking Reality" of Michael Ridpath, who described a virtual reality display to understand the activities of the world's bond markets.
"... I was looking at a representation of the world's bond markets. The hillside was made up of a series of ridges. Each ridge represented a bond market; the higher the ridge, the higher the yield. ..."
The main aspect here is the immersion of the user into a virtual created reality. Howard Rheingold [8] mentioned the importance of immersion in Virtual Reality. The user is inside a computer-generated scene, in our case, inside an arbitrary active program. With stereoscopy, gaze-tracking, and other technologies the user can feel the program. And this is the intention of our project: feeling the program.

The remainder of this paper is organized as follows. The next section describes analysis of large parallel programs, and monitoring & steering mechanisms for (parallel) programs. A brief overview of related work and methods for Virtual Reality will be given. Section three will cover the problem of relating byte-streams to visual representations.
The last section will present future work to do, ideas of supporting visual representation with sounds - *auralization*.

2 Analysis of large parallel programs

Solving big problems (weather forecasting, simulation of astronomical events, ...) often requires parallel systems for high-performance computing. The runtime of such programs often takes a lot of time (several days, or even several weeks). During this time, there is no possibility to look inside of the calculation without a special mechanism. Users often want to assist and guide these long running computations using their domain knowledge.
The goal of our project [4] is to be able of on-line monitoring & steering of any arbitrary active program in memory. The main advantage of our system is, that there is no need to modify the source codes and to recompile the program. Therefore, the user needs not restart the program. Since large amount of data is provided by parallel programs, it seems natural to use graphical visualization.
Due to a big variety of existing visualization tools, we had to decide for one for us suitable tool. We voted for the Data Explorer from IBM to display provisional results of the calculation process, because of several advantages:

- it is an open-source software
- it has a client-server architecture
- it supports a lot of platforms
- it takes full advantage of multi-processor systems

One part of this project is the visualization of programs themselves, and this is the topic of this paper. The user should be able to feel the program, which is lying in memory. Different representation of programs will be described in one of the next sessions.

3 Related Work

A lot of related work on the topic "VR and data representation" already exists. In this chapter we give a brief overview of three well known basic approaches.
In [1], a data-immersive virtual environment - called *Avatar* - has been developed at the University of Illinois, which explores performance data and provides real-time adaptive control application behavior. Using three-dimensional scatter-cubes, they

try to give the user an overview of the performance of the parallel program. One of the advantages of three-dimensional representation is the ability of changing the viewing perspective. Through personal movements, users can explore the objects in a natural way.

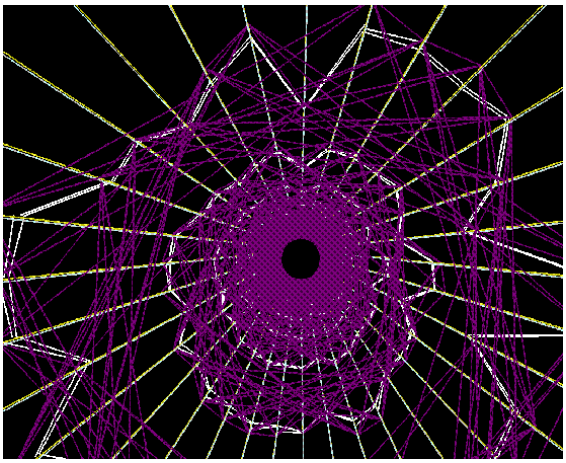

Abbildung 1: Time-Tunnel display

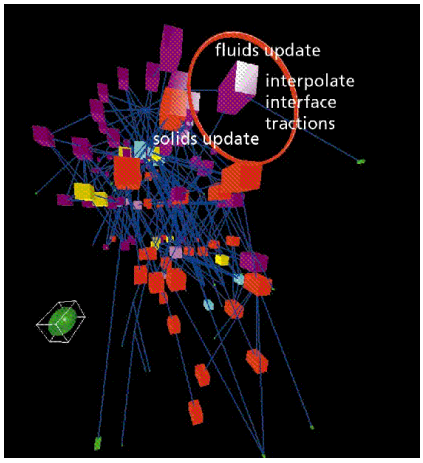

Abbildung 2: Call-graph display

Another prototype has also been developed by the University of Illinois - called *Virtue*. *Virtue* [2] exploits human sensory capabilities to help performance analysts explore large-scale applications. One part of this project is the visualization of communication between parallel processes, displayed by a Time-tunnel (Figure 1). In other words, users are surrounded by tasks, and they can see the processor communication as well as task-computation. Therefore, analysts can see performance problems within communication, and where performance enhancements are useful.
A further part of *Virtue* is the Call-graph display (Figure 2). This data mapping represents procedures, vertex color and size indicate invocation count and time spend by the procedure. The labels show the names of the procedure.

The Computer Science Department at University of Wisconsin-Madison developed a tool called *Devise* [6], which is a generic visualization tool to allow arbitrary number of different but related data-streams to be integrated and explored visually in a flexible manner. In contrast to both other projects, they do not use Virtual Reality to display the information. Streams of data are displayed in line plots, histograms or bar charts.

Our representation is limited to three-dimensionality. We also have a Time-tunnel-like display, as well as a fractal landscape. Detailed information about our approach will be given in the next sessions.

4 Semantics of Byte-Streams

The main issue of visualizing a program is the way, how to display a byte-stream. Code and data in memory are nothing else than a large amount of bytes. Code is not worth displaying, because there is not to really any activity in the code section in memory, except the user does code-patching. When we are speaking of bytes, we just mean raw data.

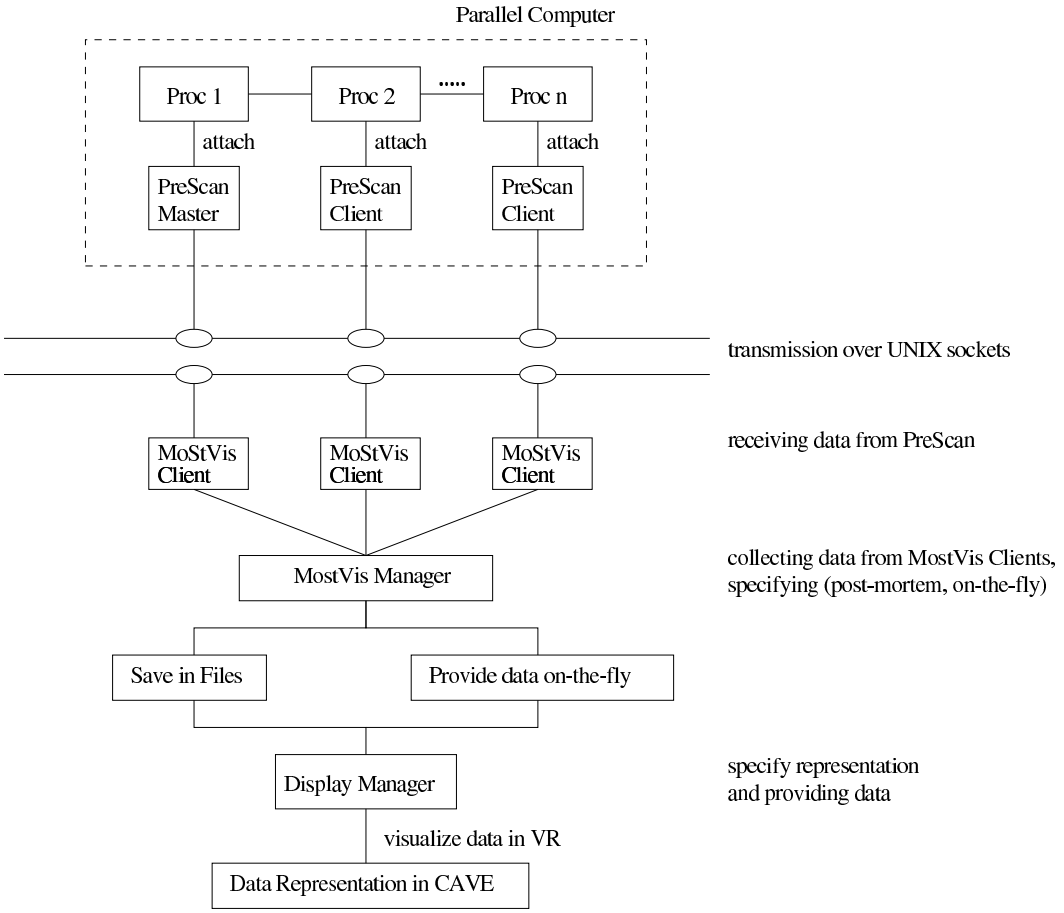

Abbildung 3: Block Diagram of Visualization Process

A core dump out of an active program provides one stream of bytes. Our idea

is, seeing the byte-stream as a two-dimensional array (mapping a one-dimensional stream with the square-root function).

The bytes are transmitted from the steering program by UNIX sockets. As a result, several streams can be transmitted parallel.
Basically, there are two different delivery types:

- post-mortem
- on-the-fly (real-time)

In the post-mortem approach, the whole information is being collected over a period of time. Independently from execution-time, the user can view the activity of the program at any time.
In contrast to post-mortem approach, the on-the-fly approach the data-stream is provided during program execution every δ seconds, or even in real-time.

Figure 3 describes the visualization process in a schematic way. The PreScan Master forks PreScan Clients on each node of the parallel computer. After attaching to the processes, every PreScan Client transmits the raw data-stream over UNIX sockets. MostVis Clients – running as threads – receive the information and store them into a global data array.
The MostVis Manager collects these information and specifies the delivery type (post-mortem, on-the-fly). In dependence on this type, the Display Manager fetches the data and displays it in the CAVE (Cave Automatic Virtual Environment).

5 Program Representation in VR

To find a representation of a byte-array, or of an arbitrary byte-stream is not a trivial problem. We selected two suitable display types for our program visualization.

Firstly, we take the byte-array, and map it into a fractal landscape (Figure 4). Mountains in this landscape mean a lot of activity in this part of the memory, and valleys mean no activity. As a consequence, the user can see which part of the program needs to be discovered in detail. For example, the user can set bounds for an array to visualize this part of the program in another representation (color-maps, rubber-sheets, ...).

The second representation is a tunnel (schematic sketch in Figure 5), where every single byte takes one small slice of the tunnel. The color in this tunnel shows the amount of activity in the program. Red colors mean much activity, whereas blue color indicates less activity.
The problem of this approach is, that the tunnel can get very long, because of the large amount of data. Image, the program has 1 MB in memory, the byte-stream would be 1.048.576 bytes. Reserving 1 cm for 1 byte, the actor would have to "walk" 10 km. And this is impossible! Hence, this representation is only effective with small programs.

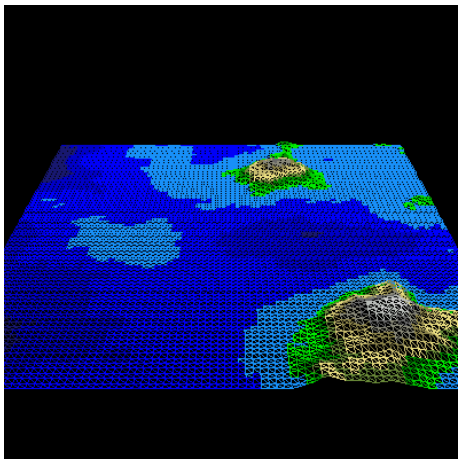

Abbildung 4: A fractal landscape, which represents the activity of byte-streams in memory

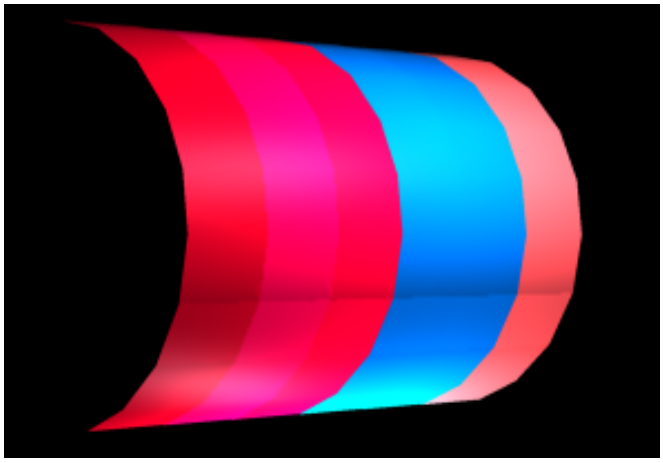

Abbildung 5: Schematic sketch of a tunnel, showing the visualization of byte-activity

Both, the fractal landscape and the tunnel are displayed in the CAVE. As for parallel processes, the user can choose out one of the active processes, and either fly over the landscape or navigate through the tunnel. As mentioned before, the CAVE provides an immersive feeling so that the user feels as if being in the program, which is supplemented by a full three-dimensional effect because of stereoscopy.
The benefits are obvious. The user has the opportunity to see, what is happening in the program and can see enforced activity regions. For example, regarding a poisson equation, the user can see a big mountain range and one lonesome mountain peak. The range represents the array, in which the color values are calculated, and the peak represents the ϵ - the barrier for the calculation.

6 Future Work

In our project, only two possible representations of a program are picked out, but there are further display types. Due to our extensible framework, the display type can be exchanged very simple.
Beside visualization of program, we are also thinking of *auralization*. Even in the early years of computing (1950), Fernando Jose Corbató described a feature of the Whirlwind computer as follows:

"You even had audio output in the sense that you could hear the program because there was an audio amplifier on one of the bits of one of the registers – so each program had a signature. You could hear the tempo of how your program was running. You could sense when it was running well, or when it was doing something surprising.[3]"

Francioni and Jackson presented approaches [5] for using sounds as a investigation of parallel program behavior. They are using different sounds for different events in a parallel program.
As for the tunnel, sound support can be very effective. Thinking of Didgeridoo sounds, smooth sound effects represent few activity whereas vibrant effects mean a lot of activity.
In general, the ear is more sensitive than the eyes. And combining these two senses, the user gets an impressive imagination of the dynamic processes in programs.

7 Conclusions

Parallel programs use large amount of data. Users are often interested in viewing the dynamic of such programs, and inspecting the data, which are lying in memory. An important role in visualizing these data plays immersion. The CAVE, is the best way to provide immersion for the user. With stereoscopy and gaze-tracking, the user gets the impression of being inside the program, which is one of the goal of our project.
As a consequence, the user has the opportunity to define a range in the array, to visualize this range in more detail in another representation and in another tool (like Data Explorer).

8 Acknowledgements

This work was supported by the rest of the MoSt-Team (Christian Glasner, Roland Hügl), who are working on the Monitoring & Steering Project at the Department for Computer Graphics and Parallel Processing.
Special thanks to Dieter Kranzlmüller and and my supervisor O.Univ.-Prof.Dr. Jens Volkert.

Literatur

[1] W.H. Scullin L.F. Tavera G.L. Elford D.A. Reed, K.A. Shields. Virutal reality and parallel systems performance alalysis. *IEEE*, November 1995.

[2] S.. Whitmore B. Schaeffer E. Shaffer, D.A. Reed. Virtue: Performance visualisation of parallel and distributed applications. *IEEE*, pages 44–51, 1999.

[3] K.A. Frenkel. An interview with fernando jose corbató. *Comm. ACM*, 34(9):83–90, 1991.

[4] D. Kranzlmüller J. Volkert. Monitoring and steering of parallel and distributed programs. Technical report, Dept. for Graphics and Parallel Processing (GUP Linz), 1999.

[5] J.A. Jackson J.M. Francioni. Breaking the silence: Auralization of parallel program behavior. *Academic Press, Inc.*, 1993.

[6] Miron Livny Barton P. Miller Karen L. Karavanic, Jussi Myllymaki. Integrated visualization of parallel program performance data. In *Parallel Computing*, pages 181–198, 1997.

[7] R. Aydt E. Shaffer B. Schaeffer S. Whitmore L. De Rose, M. Pantano and D. Reed. An approach to immersive performance visualization of parallel and wide-area distributed applications. In *IEEE Symposium on High Performance Distributed Computing.*

[8] Howard Rheingold. *Virtual Reality.* Mandarin, 1991.