Exploring UltraSPARC T2 Processor Parallel Processing Capabilities in Video Streams

Javier Iparraguirre[1,2], Claudio Delrieux[1], Lisandro Perez Meyer[1], and Adrian Rostagno[2]

[1] Universidad Nacional del Sur, Laboratorio de Ciencias de las Imagenes,
Avenida Alem 1253, Bahia Blanca, Argentina
{javier.iparraguirre,cad,perezmeyer}@uns.edu.ar
www.imaglabs.org

[2] Universidad Tecnologica Nacional, Facultad Regional Bahia Blanca
11 de Abril 461, Bahia Blanca, Argentina
arostag@frbb.utn.edu.ar
www.frbb.utn.edu.ar

Abstract. UltraSPARC T2 is a system on a chip. It contains 8 cores and it is capable of running 64 simultaneous hardware threads. Although it was designed for server workloads with high throughput requirements, we tested the architecture using a visual computing application. In this work we perform image binarization in video streams. We use a serial and a parallel implementation to test the performance of the parallelization. An evaluation of the speedup for several screen resolutions of the same video stream shows that this architecture can also be a good choice for real-time video streaming and processing applications.

Keywords: Multi-core processing, Parallel processing, Video processing, SMP architectures

1 Introduction

Processing video can be a challenging problem in terms of hardware requirements [3]. Depending on the requirements of the application, the resources demand can scale exponentially. Visual application developers are always looking for new computing alternatives to speed-up the execution times and to be able to include better processing features in the system. In visual computing, the faster the system runs, the more vivid the user experience is, which is the most important asset in these applications.

UltraSPARC T2 is the second generation of a system-on-a-chip (SOC) family developed by SUN Microsystems [7]. It has 8 cores in one single chip. The processor is capable of running 64 hardware threads at the same time. Each of the 8 cores has one floating point unit and two integer units. Typical applications intended for this design are database server, web server, virtualization, and Java application server.

The same architecture was tested in different arenas. Part of the design team showed the capabilities of the architecture [7]. It is possible to find works that show the behavior at the operating system level [8] [5]. There are even experiences using UltraSPARC T2 in Computational Science [6].

In our case, we are interested in exploring the potential of the machine in visual computing applications. In this work we process a video in several frame resolutions. So far our main interest is to assess the viability of applying this architecture in real-time video processing applications, and therefore we tested the system using only simple image processing filters, in this case digital binarization [4]. The speedup factors that we achieved, as compared to a serial reference implementation, allows us to conclude that this architecture is a promising alternative in real-time video processing applications.

In the next section we describe the changes required in the original configuration and setup of the server for this application. Section 3 explains the internals of the testing we performed. The results can be found in Section 4. Finally, we state the conclusions and future work in Section 5.

2 Machine Setup

For this testbed we used a Sun SPARC Enterprise T5120 Server (`www.oracle.com/us/products/servers-storage/servers/sparc-enterprise/index.html`). The machine has 32GB RAM memory and 2 SAS drives 146 GB, 2.5 in. By default, the system had Solaris 10 installed (`www.sun.com/software/solaris`).

Given the intended application and the fact that some required software libraries are not available in the original setup of the machine, we installed Debian GNU/Linux (`www.debian.org`), and therefore the system is running the 2.6.32-3-sparc64-smp Linux kernel. We did not tune any of the packages in the system. All the configurations follow the default parameters defined by the original distribution.

3 Doing Binarization

In the image processing arena, binarization is the process that divides the pixels contained in a image into two sets [4]. Based on a threshold level, all the pixels above it are converted into white and the rest into black. The process needs a gray-scale image as an input. In case of color images, they have to be converted into gray-sale before the binarization is applied. In terms of computational requirements, the binarization means to visit all the pixes of the image and decide if the pixel will be white or black.

We designed an application that opens a video stream, converts each frame to another gray scale frame, and produces a binarization of the gray-scale frame. We have two methods to perform binarization. The first one is the regular serial binarization. The second is a binarization that employs OpenMP[2]. Our application use OpenCV [1] to open the stream but we designed out own processing methods to ensure the correctness in the performance analysis. We used C++ as development language. Figure 1 shows an screen-shot of the application in development stages.

The parallel implementation of our application consist on accelerating just the part of the code that produces the binarization. In practical terms, the parallel code is equal to the serial one except for the OpenMP directives [2]. We calculated the total time consumed by the serial binarization and the total time of the parallel version. As a result, the speedup values is calculated as the serial time over the parallel time.

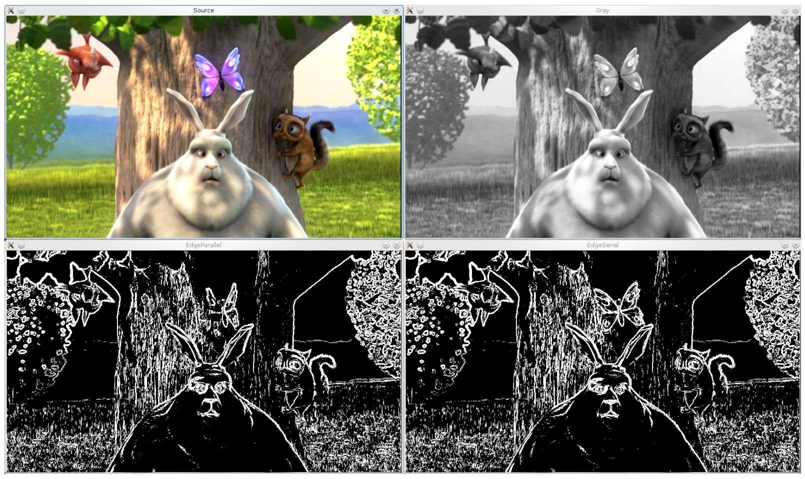

Fig. 1. A screen-shot of our application during development stages. The original movie is in the upper left corner. The same image as gray-scale is in the upper right corner. The serial binarization is in the lower left corner and the parallel binarization is in the lower right corner

Big Buck Bunny `www.bigbuckbunny.org` was used as our test video. It is possible to download free versions of the movie in several formats and resolutions. In our case we downloaded the Ogg format streams and we used three different resolutions.

4 Results

We run the application on the machine and we measure the performance. We compared the execution of the serial binarization against the parallel version on the same operation. The same experiment was repeated for three different movie resolutions. The first resolution was 854x480. The second was 1280x720 and the last one was 1920x1080. We found that the speedup values were 10.49 for the lowest resolution, 12.22 for the medium size and 13.74 in the case of the highest quality video. Detailed information is expressed in Table 1.

Based on the results, it is possible to see that the parallelization scales well among the cores. Another fact is that the speedup increases with the size of the image frame. This is an indicator that as the size of the frame increases, the relative overhead of the parallel threads decreases.

Table 1. Speedup values for each of the available resolutions

Resolution	Speedup
854 x 480	10.49
1280 x 720	12.22
1920 x 1080	13.74

There are some factors that are important in the analysis of the results. The first detail is that binarization only uses integer operations and we are not testing the floating point features of the processor. Another detail is that we are not comparing the machine against other computer architectures. We are evaluating the way an application scales in the same machine.

5 Conclusions and Future Work

We tested the performance of UltraSPARC T2 processing video. We only performed integer operations. The application scaled up to 13.74 times faster running in parallel mode. The results look reasonable since we are using 8 cores to perform the computing work.

Our future research will focus on three different directions. The first path will be testing the machine in other visual applications with more sophisticated requirements and processing complexities. The second track will orient towards UltraSPARC T2 floating point capabilities. Finally, we are interested in comparing this machine against x86 architectures and graphic processing units (GPU's).

References

1. G.R. Bradski and A. Kaehler. *Learning opencv.* O'Reilly, 2008.
2. R. Chandra. *Parallel programming in OpenMP.* Morgan Kaufmann, 2001.
3. R. Datta, D. Joshi, J. Li, and J.Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys (CSUR)*, 40(2):1–60, 2008.
4. R.C. Gonzalez and R.E. Woods. *Digital image process.* Prentice-Hall, Upper Saddle River, New Jersey, 2002.
5. Y. Guo, R. Barik, R. Raman, and V. Sarkar. Work-first and help-first scheduling policies for async-finish task parallelism. 2009.
6. Martin Sandrieser, Sabri Pllana, and Siegfried Benkner. Evaluation of the sun ultrasparc t2+ processor for computational science. In *ICCS '09: Proceedings of the 9th International Conference on Computational Science*, pages 964–973, Berlin, Heidelberg, 2009. Springer-Verlag.
7. M. Shah, J. Barreh, J. Brooks, R. Golla, G. Grohoski, N. Gura, R. Hetherington, P. Jordan, M. Luttrell, C. Olson, et al. UltraSPARC T2: A highly-threaded, power-efficient, SPARC SOC. In *IEEE Asian Solid-State Circuits Conference*, pages 22–25, 2007.
8. V. Čakarević, P. Radojković, J. Verdú, A. Pajuelo, F.J. Cazorla, M. Nemirovsky, and M. Valero. Characterizing the resource-sharing levels in the UltraSPARC T2 processor. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 481–492. ACM, 2009.