Generation of Microlensing Magnification Patterns with High Performance Computing Techniques *

Pablo López[2] Antonio J. Dorta[1] Evencio Mediavilla[2] and Francisco de Sande[1]

[1] Depto. de E.I.O. y Computación, Univ. de La Laguna,
38271–La Laguna, Spain; {falmeida, fsande}@ull.es
[2] Instituto de Astrofísica de Canarias, (IAC)
38205–La Laguna, Spain; {plopez, emg}@iac.es

High Performance computers have became an essential part of modern research, but the software tools used to harness their power have not reached the level of simplicity expected by the average researcher. Researchers are familiar with the logic of sequential coding and the time needed to learn new tools is seen as a distraction from their field of research. Furthermore, most of the issues related to parallel programming (such as concurrency, message passing, etc.) are foreign to them. Last, but not least, codes are usually developed on their workstations which are single processor systems. They can develop, test and run without having to deal with queues, shared resources, etc. using tools which are familiar to them. This generates situations where problems which can be solved with current technology, are put aside because execution times are considered too large to be useful. In many cases, only through the collaboration with researchers in computer science these codes can be transformed into tools which allow for further advances in their fields.

In this work we show a case where a sequential code, which can become an important tool for astrophysics research, presents a parallelization problem which is not trivial to solve by programmers without a computer science background. Astrophysics researchers at the Instituto de Astrofísica de Canarias (IAC) developed a Fortran77 code to generate magnification patterns induced by micro lensing. In order to understand the value of this research and its computational complexity, an introduction to the problem needs to be given.

Deflection of light by gravity is one of the most outstanding predictions of general relativity. It was experimentally confirmed by the famous expedition to measure the displacement of the apparent positions of stars on the sky caused by the gravitational field of the Sun during the eclipse of 1918 in Egypt. In spite of its great importance like a test of general relativity, the deflection of star light by the Sun or any other star is a relatively weak effect. After the discovering of galaxies astronomers realize that the huge gravitational field of these objects may induce strong optical effects like the formation of multiple images (in a way similar to terrestrial atmospheric mirages). But, it was only 60 years after the expedition to the Egypt eclipse, that a double image of a

* This work has been partially supported by the EC (FEDER) and the Spanish MCyT (Plan Nacional de I+D+I, TIN2005-09037-C02-01)

very far away object caused by the gravitational field of an intervening galaxy (the lens galaxy) was discovered. The distant source was a quasar, a galaxy that hosts in a tiny nuclear region a super-massive black hole surrounded by a shining disc of in-spiraling matter (active galactic nucleus). The brightness of the two images of the quasar is different and depends on the source and lens distances and on the degree of alignment among, observer, lens and source. As far as this geometry is not changing, the ratio between the brightness of the two images should be constant. However, the distribution of matter in galaxies is not smooth but strongly discontinuous. It is granulated in stars and, perhaps, other compact objects. Thus, the movement of one star (or more realistically of a distribution of stars) crossing the light beam of one of the quasar images can produce a small scale but measurable gravitational lens effect (quasar micro lensing). As far as the movements of the stars in the regions were the light-beams of the two images cross the lens galaxy are independent, micro lensing will induce uncorrelated variability in the brightness of the two images. This variability will give to us information about two important physical issues very difficult to study by other means: the distribution of stars in the lens galaxy and the unresolved structure of active galactic nuclei.

The study of the experimental records of brightness variability of lensed quasars with time (light curves) face an statistical (formally stochastic) problem. We must find what configurations of stars in the lens galaxy and trajectories (in reality relative displacements) of the source can reproduce the observed variability. To obtain statistically acceptable estimates of the physical magnitudes involved, a large number of light curves based in random distributions of stars and trajectories should be generated. Usually this is attempted by computing magnification patterns that give, as a function of position, the magnification induced by micro lensing in the source plane. The displacements of the source across this pattern will generate the model light curves. Magnification patterns are usually computed using the inverse ray shooting technique that consists in back shooting a regular grid of rays from the image plane to the source making the amplification in a given source-plane pixel proportional to the number of light rays collected by it. This imply a high computational cost. On the one hand, around 2^8 rays per pixel should be shot to obtain magnification patterns with reduced noise. On the other hand, the ray equation include contributions for even thousands of stars that should be evaluated for each shot.

The execution time for the sequential code used for the generation of the magnification patterns is unacceptable even for cases with a small number of stars. Considering the fact that the number of executions needed to obtain a statistically acceptable estimate is approximately one hundred, we can only conclude that the usefulness was, to say the least, limited. In order to make it a viable tool, execution times have to be decreased by at least three orders of magnitude. Clearly this could only be achieved through the use of High Performance Computing techniques.

In the last years OpenMP [1] and MPI [2] have been universally accepted as the standard tools to develop parallel applications. OpenMP is a standard for

shared memory programming. It uses a fork-join model and it is mainly based on compiler directives that are added to the code that indicate the compiler regions of code to be executed in parallel. MPI uses an SPMD model. Processes can read and write only to their respective local memory. Data are copied across local memories using subroutine calls. The MPI standard defines the set of functions and procedures available to the programmer. Each one of these two alternatives have both advantages and disadvantages, and very frequently it is not obvious which one should be selected for a specific code. The pure cases of MPI or OpenMP programming models have been widely studied in plenty of architectures using scientific codes. The case of mixed mode parallel programming has also deserved the attention of some researchers in the last few years ([3], [4]), however the effective application of this paradigm still remains as an open question.

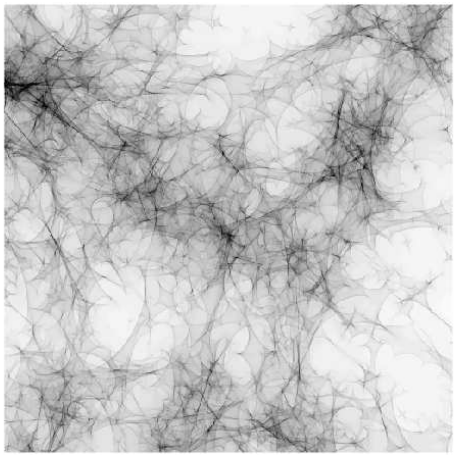

Fig. 1. Resulting matrix. Dark areas represent high density of ray hits

One should expect a high level of speed up in a parallel version of the code given that the processing of each light ray is independent from the others. While it may seem as a simple problem of nested loops parallelization, a difficulty arises from the memory access pattern. Looking to Figure 1, we can see that the distribution of the ray hits is not regular. Due to the nature of the problem, it is not possible to determine a non concurrent memory access pattern. We can also see that access to the matrix is sparse. Figure 2 shows the distribution of hits after one hundred source image lines have been processed. Therefore, we also have to deal with minimizing the number of cache misses that are intrinsic to this kind of accesses. Finally, a load imbalance situation arises due to the need to extend the magnification patterns beyond the area of the resulting matrix. This means that a significant percentage of rays will not resolve into a memory update which generates a certain level of imbalance. In order to parallelize the code this issues have to be taken into account.

We will show that the required speed up has been achieved by applying a simple strategy to the resulting matrix memory update problem. Different

parallel versions of the code are currently available. We have developed pure OpenMP and MPI versions, a Master-slave MPI approach (to reduce the impact of the load imbalance) and a hybrid MPI+OpenMP solution. Currently we are working in the `llc` ([5], [6]) implementation of the code. Experimental results for all these versions on different platforms will be provided for the final version of this contribution, if accepted.

Preliminary experimental results for the pure OpenMP implementation have been carried out on a Bull Novascale architecture with Intel Itanium2 processors interconnected with a Quadrics Network. The results (a speed up of 14.5 with 16 processors) show an almost linear behavior and good scalability. Our aim is to enlarge the computational experience, using different architectures and problem sizes.

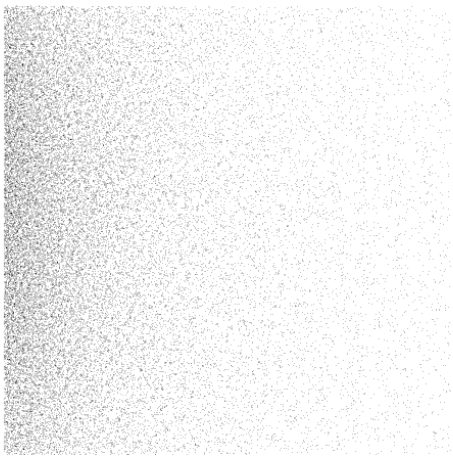

Fig. 2. Distribution of ray hits after 100 lines.

References

1. OpenMP Architecture Review Board, OpenMP Application Program Interface v. 2.5 (May 2005).
2. Message Passing Interface Forum, MPI: A Message-Passing Interface Standard, University of Tennessee, Knoxville, TN, 1995, `http://www.mpi-forum.org/`.
3. L. Smith, M. Bull, Development of mixed mode MPI/OpenMP applications, Scientific Programming 9 (2–3) (2001) 83–98.
4. F. Cappello, D. Etiemble, MPI versus MPI+openMP on IBM SP for the NAS benchmarks, in: Proceedings of Supercomputing'2000 (CD-ROM), IEEE and ACM SIGARCH, Dallas, TX, 2000, lRI.
5. A. J. Dorta, J. A. González, C. Rodríguez, F. de Sande, llc: A parallel skeletal language, Parallel Processing Letters 13 (3) (2003) 437–448.
6. A. J. Dorta, J. M. Bada, E. S. Quintana, F.Sande, Implementing OpenMP for clusters on top of MPI, in: Proc. of the 12th European PVM/MPI Users' Group Meeting, Vol. 3666 of Lecture Notes in Computer Science, Springer-Verlag, Sorrento, Italy, 2005, pp. 148–155.