Parallel Implementation of an Astronomical Algorithm for a Hybrid Computing with OpenACC

Ana Luisa Veroneze Solórzano
Universidade Federal de Santa Maria
Av. Roraima, 1000 - Camobi, Santa Maria, RS
Laboratório de Sistemas de Computação (LSC)
+55 51 99880-6691
alsolorzano@inf.ufsm.br

1. INTRODUCTION

To study cosmic evolution, Astronomy algorithms have to process big amounts of data about the universe. This data comes from dark matter simulations performed with N-body methods, and can be used to predict the formation of structures like galaxies, cluster of galaxies and superclusters [1, 6].

Friends-of-Friends (FoF) is a widely used percolation algorithm to identify structures in dark matter distribution based on physical proximity, because of its simplicity to implement, receiving one parameter and one input file [8, 3]. However, for higher resolutions, the processing of complete datasets can bring computational challenges.

Hybrid computing is an invaluable option to high performance processing. One approach to explore this environment is to run costly parts of the application in accelerator devices used as co-processors, such as FPGAs or GPUs. CUDA, OpenCL and OpenACC are some of the various tools for parallelizing programs with accelerators.

This work presents an approach to explore the use of GPUs as accelerators for a FoF algorithm parallelized with OpenACC. Since not all astronomers have knowledge about computing, our goal is to build a simple algorithm, without using complex data structures and with a simple and accessible description on the implementation to enable the people from the Astronomy field to reproduce our experiments and improve the algorithm according to their needs. The approach described in section 4 uses OpenACC directives to distribute computing in a hybrid environment, resulting in a GPU usage of up to 99% with minimal time spent with data transfering. This yields a higher speedup compared to other implementations, but also reveals limits, pointing new ideas for a better approach.

2. FRIENDS-OF-FRIENDS

Friends-of-Friends is a grouping algorithm for identifying structures in the universe according to a single free parameter (R), that determines the linking length between pairs of particles [5]. It receives an input file with dark matter data sampled by a list of particles of equal mass and three-dimensional positions.

The basic idea of this algorithm is: Consider a sphere of radius R around each particle. If there are other particles inside this sphere, they will be considered belonging to the same group and will be called friends. Then a sphere is drawn around each friend and the procedure continues using the rule of "any friend of my friend is my friend". The procedure stops when no new friends can be added to the group.

3. RELATED WORK

Kaehler uses a similar approach to parallelize N-body dark matter simulations in GPU using CUDA. However, their algorithm was implemented with an octree structure in a cluster with 256 GPUs [7]. Feng et al. presents a faster FoF algorithm using an hierarchical tree structure. This algorithm reduces the

number of operations and time complexity, but does not use parallel approaches [4].

The sequential FoF with time complexity $O(N^2)$ is highly CPU demanding. There are other FoF versions of complexity $O(N^2)$ parallelized in CPU using MPI and OpenMP, but they do not yield significant performance improvements [9, 2].

Despite using parallel approaches, none of them use accelerator devices. Seeing this, we proposed a novel approach, parallelizing the FoF algorithm for GPU in order to develop a faster algorithm for execution of complete datasets of dark matter simulations. We first tried approaches parallelizing the algorithm using only OpenACC directives, but the best version achieved only 35% of GPU usage and spent more than half the time with data transfers. We then started the implementation of a new version to better explore the hybrid environment.

4. METHODOLOGY

Our goal was to improve the performance of FoF in GPU, optimizing the use of computational resources, exploring the memory upper bound and minimizing transferences between CPU and GPU. We choose OpenACC because it is a highly portable tool that parallelizes applications with compiling directives, abstracting implementation details, such as interactions between host and accelerator.

4.1 FoF with OpenACC

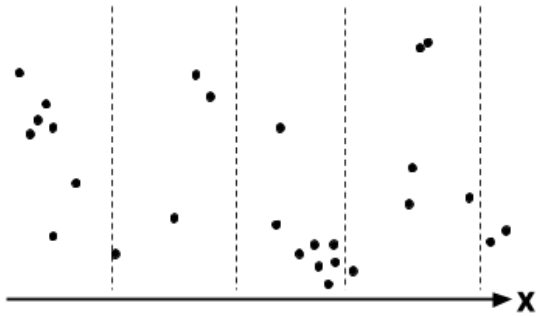

Figure 1: Example of the strategy for FoF parallelization, subdividing the input data into the GPU kernels. This division was based on coordinate x and used 5 kernels.

We based our first strategy in the test platform available composed of one CPU and one GPU. The main challenge of this implementation was to find an efficient load balancing scheme across the GPU. The FoF algorithm is sensible to the particles positions, that is, if most particles belong to few groups, the processing will be much faster and the processor will be idle.

Figure 1 show the basic idea, which was to construct a tessellation of the particles considering one of their three-dimensional position. To treat all the particles equally, we first ordered the input file by one coordinate and divided the sheet into chunks with equal number of particles. Each chunk is then assigned to different GPU kernels that run the FoF sequentially. The user is responsible to set the number of chunks by standard input.

Notice that there can be pairs of particles located in border zones, that are treated by different kernels but belonged to the same group. So, after the CPU waits all the kernels to finish their processing, it runs a post-processing to connect these groups.

4.2 Execution environment

We perform experiments in a server node with two Intel® Xeon® E5-2650 v3, each with 10 cores and Hyperthreading, amounting to 20 physical cores, 128 GB DDR4 RAM, and two GPUs NVIDIA Tesla K80 with 4992 CUDA cores.

To guarantee the algorithm correctness we first perform executions with small controlled data comparing results with other FoF versions. After, we focus on run tests to explore the available kernels and upper bound memory. The inputs used were chunks of the total Virgo Consortium[1], and we found the best number of kernels by tests with each input file.

5. RESULTS

To investigate the GPU and memory usage we use the NVIDIA's profiler nvprof[2]. First, we run executions to measure the algorithm performance, presented in Table 1. We achieved a speedup of 48.9 compared with the serial version with a file with 174,761 particles. The speedup did not increase gradually

[1] http://www.mpa-garching.mpg.de/Virgo/data_download.html
[2] https://developer.nvidia.com/nvidia-visual-profile

because the processing varies depending on the particles positions. With nvprof we observed a 99.8% of GPU usage for the file with 249,420 particles, spending 0.15% with copies between host and device.

Table 1. Performance results of the FoF with OpenACC.

Number of particles	Time (seconds)	Speedup
65,536	1.32	34.62
174,761	6.28	48.90
249,420	286.63	18.76

However, the algorithm presented errors to process complete datasets, working fine until the number of particles exceeds 2.5 millions. We are currently investigating this issue, with the strongest hypothesis being the memory capacity of the GPU.

6. CONTRIBUTIONS

With this work we hope to prove that high performance computing can be applied in many areas, using tools like OpenACC and nvprof to facilitate the implementation by anyone. We presented a version of a high performance FoF, an Astronomy algorithm, adapted to hybrid computing with OpenACC.

This FoF version was implemented to an environment with CPU and GPU, focusing on using the available GPU cores to improve performance. Even though it increases performance significantly, the algorithm is limited by the input file size because of its implementation. Our next step is to investigate where the problem is happening and find a better solution to rewrite the code, still avoiding complex data structures.

With more computational resources, we hope to modify this algorithm to run in a cluster, dispatching particles chunks to different cluster nodes, as well as different GPU kernels inside each node. Also, as future works, we hope to automate the load balancing by using an algorithm to find the optimal kernels and nodes number based on the input data. Our final goal is to make the algorithm easily available in a web Virtual Observatory effort for remote executions of Astronomy algorithms, that is being developed in our research.

7. REFERENCES

[1] Bertschinger, E. 2003. Simulations of structure formation in the universe. In Annual Review of Astronomy and Astrophysics. 36. 599-654. DOI= https://doi.org/10.1146/annurev.astro.36.1.599.

[2] Berwian, L., Zancanaro, E. T., Cardoso, D. J., Charão, A. S., da Ruiz, R. S. R., and Campos Velho, H. F. 2017. Comparação de estratégias de paralelização de um algoritmo friends-of-friends com openmp. In Anais da XVII ERAD-RS, Pages 279– 252. http://www.lbd.dcc.ufmg.br/colecoes/erad/2017/064.pdf.

[3] Duarte, M., and Mamon, G. A. 2014. How well does the Friends-of-Friends algorithm recover group properties from galaxy catalogues limited in both distance and luminosity?. In Monthly Notices of the Royal Astronomical Society, Volume 440, Issue 2, 11 May 2014, Pages 1763-1778. DOI= https://doi.org/10.1093/mnras/stu378.

[4] Feng, Y., and Modi, C. 2017. A fast algorithm for identifying Friends-of-Friends halos. In Astronomy and Computing. DOI=https://doi.org/10.1016/j.ascom.2017.05.004.

[5] Huchra, J. P., and Geller, M. J. 1982. Groups of galaxies. I - Nearby groups. In Astrophysical Journal, Volume 257, 423-437, June 1982. DOI= https://doi.org/10.1086/160000

[6] Igouchkine, O., Leaf, N., and Kwan-Liu Ma. 2016. Volume rendering dark matter simulations using cell projection and order-independent transparency. In SIGGRAPH ASIA 2016 Symposium on Visualization (SA '16). ACM, New York, NY, USA, Article 8, 8 pages. DOI= https://doi.acm.org/10.1145/3002151.3002163.

[7] Kaehler, R. 2017. Massively parallel computation of accurate densities for N-body dark matter simulations using the phase-space-element method, In Astronomy and Computing, Volume 20, Pages 68-76, ISSN 2213-1337. DOI= https://doi.org/10.1016/j.ascom.2017.05.005.

[8] More, S., Kravtsov, A. V., Dalal, N., and Gottlöber, S. 2011. The overdensity and masses of the friends-of-friends halos and universality of halo mass function. In Astrophysical Journal Supplement Series, Volume 195, Issue 1. DOI= https://doi.org/10.1088/0067-0049/195/1/4.

[9] Ruiz, R. S. R., Campos Velho, H. F., and Caretta, C. A. 2009. Parallel algorithm friends-of-friends to identify galaxies and cluster of galaxies for dark matter halos. In Workshop dos Cursos de Computação Aplicada do INPE, 9. (WORCAP).