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

Ana Luisa V. Solórzano[a]
Advisors: Andrea. S. Charão[a] and Haroldo F. de Campos Velho[b]
[a]Universidade Federal de Santa Maria, RS, Brazil
[b]Instituto Nacional de Pesquisas Espaciais, São José dos Campos, SP, Brazil

ABSTRACT

Astronomy algorithms process and classify large amounts of data about the Universe to study cosmic evolution. These algorithms are often developed for running with sequential instructions, being computationally costly to process large data-sets. Parallel programming is an invaluable approach to increase code performance by reducing its execution time. Hybrid computers have become a popular option to process costly sections of parallel programs in accelerator devices. A well-known algorithm to classify astronomical data is the Friends-of-Friends (FoF). In this work, we present a new parallel implementation of the FoF algorithm using OpenACC for a hybrid computing environment with CPU and GPU. The results show that the algorithm can be well explored in this context, with an expressive performance improvement compared with the original program for CPU. Also, during our preliminary investigations, we achieved a significant improvement when parallelizing FoF with minimal modifications, using only compiling directives.

KEYWORDS

parallel programming, hybrid computing, astronomical algorithm, Friends-of-Friends

ACM Reference Format:
Ana Luisa V. Solórzano[a] and Advisors: Andrea. S. Charão[a] and Haroldo F. de Campos Velho[b]. 2020. Parallel Implementation of an Astronomical Algorithm for a Hybrid Computing Environment with OpenACC. In *Proceedings of ACM SRC Grand Finals.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

To study cosmic evolution, researchers in Astronomy use algorithms to process data with information about the Universe. This data comes from Telescopes observation and simulations on the Universe dynamics. With the advances in technology, Telescopes are increasingly powerful, resulting in large data-sets to be processed.

There is a need to create high-performance algorithms to process such data for different analysis purposes. *Halo-finder* is a class of algorithms that uses computational and clustering methods, such as K-Means and AMR (Adaptive Mesh Refinement), to predict the formation of structures in the Universe, as galaxies, cluster of galaxies and superclusters [2, 11]. Some *halo-finders* are RockStar [1], Amiga Halo Finder [13] and Friends-of-Friends [9].

These software are often developed by Astronomy specialists with programming knowledge, who usually implement correct solutions using sequential programming, which can be computationally costly. Parallel programming is a field in Computer Science that encourages better use of high-performance architectures to optimize algorithm performance. The goal is to process more data in less time by splitting a problem into smaller problems and computing them in parallel. There are several libraries and APIs applied in textual programming languages to create parallel approaches.

AHF [1] and Rockstar[2] provide sequential and parallel implementations for CPU. However, the parallel execution of AHF, for example, does not bring a significant performance increase compared to the serial execution. Sometimes, there are license restrictions for access to software, and even if available, as is the case with AHF, they lack technical information about the development of the project and methods used, preventing reproducible analyzes [8].

Friends-of-Friends (FoF) is a percolation algorithm to identify structures in dark matter distribution based on physical proximity between particles that compose this dark matter. It is widely used because of its simplicity to being implemented, receiving one parameter, and one input file [6, 16]. However, for higher resolutions, the processing of complete data-sets can bring computational challenges.

Hybrid computing is an invaluable option for high-performance processing, where we run costly parts of a code in accelerator devices used as co-processors. Some hardware accelerators are FPGAs, Intel® Xeon Phi®, and GPUs. CUDA, OpenCL, and OpenACC are some of the various tools for parallelizing programs with accelerators. The OpenACC standard, used in this work, is based on the use of directives to distributed computing in a hybrid environment.

This work presents an approach to explore the use of GPUs for a FoF algorithm parallelized with OpenACC. Our goal was to build an algorithm without using complex data structures to be accessible to specialists in Astronomy, so they can reproduce our experiments and improve the algorithm according to their needs. Our approach resulted in minimal time spent with data transferring with a GPU usage of up to 99% to process FoF. It yields a higher speedup compared to other implementations [3, 19], but also reveals limits, pointing new ideas for a better approach.

ACM SRC Grand Finals, 2020,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[1] ttp://popia.ft.uam.es/AHF/Download.html
[2] https://bitbucket.org/gfcstanford/rockstar

2 RELATED WORK

High Performance Computing (HPC) is becoming a reality in scientific applications of different areas such as meteorology, finances, and astronomy [4]. Seen this, new implementations to explore the performance on multicore architectures and support the processing of large data-sets should deliver solutions accessible for scientists that are not specialists in computing.

There are different approaches to process data from N-body dark matter simulations in Astronomy. Kaehler uses CUDA, a low-level programming model for computing on GPUs. They implemented an algorithm using an oct-tree structure and dynamic load balancing schemes. Their algorithm scales up to 256 GPUs and supports TB-sized simulation in snapshot format [12].

Feng et al. presents a new halo-finder algorithm using the Friends-of-Friends clustering approach to cosmological data-sets [7]. They used a dual KD-tree correlation function and a hierarchical tree structure. Their algorithm reduces the number of operations and time complexity, increasing performance by reducing the code execution time. However, they do not use parallel approaches.

3 FRIENDS-OF-FRIENDS

Friends-of-Friends is a grouping algorithm for identifying structures in the Universe according to a single free parameter (l), that determines the linking length between pairs of particles [10]. 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 the FoF is: consider a sphere of radius l 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 friend can be added to the group.

The sequential FoF with time complexity $O(N^2)$ used in our parallel implementations is presented in Algorithm 1. The value of l is determined by b times the mean separation between particles, with the value of b depending on the applications nature [5]. The higher the l, the lower the density contrast between particles, and the higher the number of particles in each group [18]. We used l = 0.1 *Mpc* to identify galactic halos.

```
for (i = 0 ; i < N ; i++, k++) {
  while (igru[i] != 0 ) i++;
  igru[i] = k;
  for (j = i ; j < N ; j++) {
    if(igru[j] == k) {
      for (l = (i + 1) ; l < N ; l++) {
        if (igru[l] == 0) {
          dist = sqrt((x[j]-x[l]) * (x[j]-x[l])
          + (y[j]-y[l]) * (y[j]-y[l]) +
          (z[j]-z[l]) * (z[j]-z[l]));
          if (dist <= rperc)
            igru[l] = k;
        }
      }
    }
  }
}
```

Algorithm 1: Main process of the sequential FoF with time complexity of O(N^2).

4 BACKGROUND

Previous works based on the sequential FoF implementation presented on Section 3 created parallel versions using the MPI and the OpenMP standards for parallelization on CPU [3, 14]. Still, they do not yield significant performance improvements.

There is an implementation of a FoF with $O(NlogN)$ time complexity, using an oct-tree structure with cubes of length lower than l, using OpenMP in a post-processing section [15]. They achieved a significant advantage at run time, but it is important to note that their algorithm has properties that can generate overheads depending on the input data and the value of l.

Despite using parallel approaches, none of the previous works based on the FoF presented in Section 3 explored the use of a hybrid computing environment. We present a novel approach by parallelizing the FoF algorithm for an environment with CPU and GPU, to develop a faster version for execute cosmological data. We investigated two approaches described in the next section.

5 METHODOLOGY

Our first approach was an initial investigation with a few modifications in the FoF code. Then, we collected information about this execution and implemented a new FoF to explore the accelerator usage. We choose OpenACC because it is a highly portable tool to parallelize applications with compiling directives, abstracting implementation details, such as interactions between host and accelerator [17].

5.1 First investigation

We parallelized the most-inner loop in the code presented in Algorithm 1, since it does not have data dependencies. We used the OpenACC directive #pragma acc parallel that describes a region to be accelerated in the GPU cores. We notice that the variable to store the distance between particles should be private to avoid inconsistencies.

To guarantee data consistency, we used directives to copy data between devices. We performed tests using the #pragma acc copy, to copy data from CPU to GPU before entering a parallel region, and from GPU to CPU before leaving, and using the #pragma acc update device and the #pragma acc update self, that allocates an array before entering the parallel loop, and only copy modified values between devices during iterations.

5.2 Second investigation

Inspired by the first investigation, we wanted to optimize the use of computational resources, exploring the memory upper bound and minimizing data transference between CPU and GPU. We based our strategy on the test platform available composed by one CPU and one GPU.

Figure 1 shows our strategy, that 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 in different GPU kernels that run the FoF sequentially. The user is responsible to set the number of chunks by standard input.

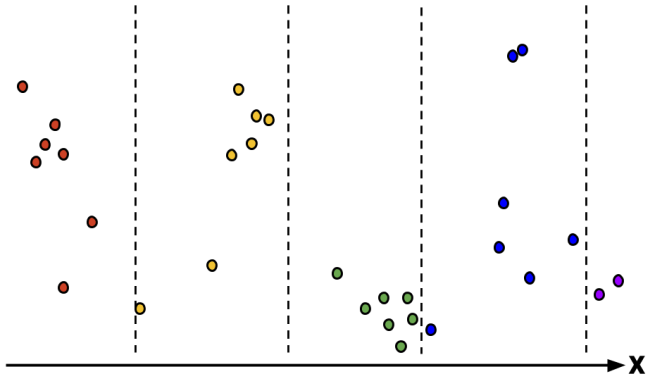

Figure 1: Strategy used for FoF parallelization, subdividing the input data in GPU kernels. This example uses five kernels and process particles based on coordinate X.

The main challenge of this implementation was to find an efficient load balancing scheme across the GPU. The FoF algorithm is sensitive to the particles' positions, that is, if most particles belong to a few groups, the processing will be much faster, and the processor will be idle. Otherwise, if most of the particles have short distances between them, the processor will have more work, cause will have more particles to apply the main algorithm.

After the kernels finish processing FoF, the array with the group indexes returns to CPU. There will be minimal data migration between devices: at the beginning, when sending the data to the kernels, and at the end when returning the group indexes to the CPU. Notice that there can be pairs of particles located in border zones, that are treated by different kernels but belong to the same group. So, after the CPU waits for all the kernels to finish their processing, it runs a post-processing to connect these groups.

The post-processing considers the distance between pairs of particles in the border zones and evaluates if it considering l as the distance limit between them. We first select the closest particle from the boundary zone of each kernel and compare its distance with the particles in the boundary in other kernels until finding a particle with a range higher than l, stopping the comparison, since the following particles will have a greater distance, belonging to different groups.

We notice that when a particle is reagrupated, it is necessary to change the particles associated with it, to classify them as belonging to the same group. This reclassification generated a new scan in several particles, using a for loop parallelized with the OpenMP.

6 EXPERIMENTS

We performed experiments in a server node with two Intel® Xeon® E5-2650 v3, each with ten cores and Hyperthreading, amounting to twenty physical cores. The server has 128GB of random access memory, and two GPUs NVIDIA Tesla K80 with 4992 CUDA cores. For this work, we used only one of the GPUs.

To guarantee the algorithm correctness, we first run executions with controlled data, comparing results with other FoF versions. Since some versions handle the relabel process and others do not, we spent time in this investigation. Our inputs data were chunks of the total Virgo Consortium[3].

We created three input files with 65,536 particles, 174,761, and 249,420. We used 200 kernels for File 1, 300 for File 2 and 600 for File 3, and percolation radius 0.1. We found the best number of kernels by tests with each input file. We run 15 executions of each case. To investigate the GPU and memory usage, we use the NVIDIA's profiler nvprof[4].

7 RESULTS

Table 1 presents the execution time of the serial FoF $O(NlogN)$ implementation, the performance results of both investigations using OpenACC, and the FoF of time complexity $O(NlogN)$, which is the previous investigation with better time performance. The speedup was calculated considering the serial version of the FoF $O(N^2)$. The execution of the FoF $O(NlogN)$ for the larger file was not completed due to an error of memory exceeded.

We can observe that, even bringing a small performance improvement, our first approach inserting OpenACC compiling directives in the sequential FoF code executed faster than the $O(NlogN)$ FoF, and did not have limitations for larger inputs. Our second investigation to perform in GPU with more code modification spent the fewer execution time for all files with a speedup of 48.9 for processing 174,761 particles.

The speedup did not increase gradually because the FoF processing varies depending on the particles' positions. With nvprof, we concluded that the second investigation spent less time in communication between devices for all executions. The first investigation spent around 46% of the total time in GPU to transfer data from CPU to the accelerator and 45% for the opposite, leaving approximately 7% the FoF. Otherwise, the second investigation spent more than 97% of GPU usage for all inputs, with 99.8% for the file with 249,420 particles, spending only 0.15% with copies between host and device.

Post-processing was not parallelized with OpenMP in our second investigation executions. For our data-sets and l of 0.1, fewer particles were classified as friends, and thus the number of particles to be reclassified after processing in GPU was smaller, making this parallelization costly. We found a limitation for the first and second investigation with OpenACC, that presented memory exceeded to process inputs with more than 2.5 millions of particles.

8 FINAL REMARKS

With this work, we expect to demonstrate that high-performance computing can be applied in many areas and that we have available tools, such as OpenACC, that can facilitate parallel implementations for novices in HPC. We presented a new version of the FoF, an Astronomy algorithm to identify structures in the Universe, to run in a hybrid computing environment with one CPU and one GPU.

We evaluated our implementation using an execution profiler for GPU and comparing our results with previous versions of the algorithm. Our goal was to explore the use of the available GPU cores and the memory upper bound to improve performance. Even

[3]https://wwwmpa.mpa-garching.mpg.de/Virgo/data_download.html
[4]https://docs.nvidia.com/cuda/profiler-users-guide/index.html

Table 1: Performance comparison of the two new investigations using OpenACC and of the previous investigation with better time performance. The times are in seconds, and the speedup is compared with the serial FoF $O(N^2)$.

	FoF $O(N^2)$	First Investigation		Second Investigation		FoF $O(NlogN)$
Number of particles	Time (s)	Time (s)	Speedup	Time (s)	Speedup	Time (s)
65,536	45.70	13.77	3.32	1.32	34.62	17.81
174,761	307.06	78.47	3.91	6.28	48.90	89.91
249,420	5375.83	1401.80	3.83	286.63	18.76	Memory exceeded

though our second investigation increased performance significantly, the algorithm is limited by the input file size.

Using more computational resources, we hope to modify this algorithm to run in a cluster, dispatching chunks of particles to different cluster nodes, as well as for different GPU cores inside each node. Also, as future works, we intend to investigate static and dynamic load balancing schemes to calculate an optimal number of particles to be processed in the GPU.

This work was part of a project to develop a web Virtual Observatory for remote executions of Astronomy algorithms. Our implementation is available in the environment, which is still being building. The source code of our implementation is available on: https://github.com/anaveroneze/acmsrc.

9 ACKNOWLEDGEMENTS

This research is part of a project in a partnership with the Federal University of Santa Maria (UFSM) and the National Institute for Space Research (INPE). The authors gratefully acknowledge financial support by the CNPq (project number: 136023/2018-5), the Brazilian agency for research support, who granted a two years CNPq/PIBIC-INPE Scientific Initiation scholarship to the first author.

REFERENCES

[1] Peter S. Behroozi, Risa H. Wechsler, and Hao-Yi Wu. 2013. The Rockstar Phase-space Temporal Halo Finder and the Velocity Offsets of Cluster Cores. *The Astrophysical Journal* 762, 2 (2013), 109. http://stacks.iop.org/0004-637X/762/i=2/a=109

[2] Edmund Bertschinger. 1998. Simulations of Structure Formation in the Universe. *Annual Review of Astronomy and Astrophysics* 36, 1 (1998), 599–654. https://doi.org/10.1146/annurev.astro.36.1.599 arXiv:https://doi.org/10.1146/annurev.astro.36.1.599

[3] L. Berwian, E. T. Zancanaro, D. J. Cardoso, A. S. Charão, R. S. R. Ruiz, and H. F. Campos Velho. 2017. Comparação de estratégias de paralelização de um algoritmo friends-of-friends com OpenMP. *Anais da XVII Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul* 9 (2017), 249–252. http://www.lbd.dcc.ufmg.br/colecoes/erad/2017/064.pdf

[4] Robert Birke, German Rodriguez, and Cyriel Minkenberg. 2012. Towards Massively Parallel Simulations of Massively Parallel High-Performance Computing Systems. In *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques (SIMUTOOLS '12)*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels, BEL, 291–298.

[5] C. A. Caretta, R. R. Rosa, H. F. Campos Velho, F. M. Ramos, and M. Makler. 2008. Evidence of turbulence-like universality in the formation of galaxy-sized dark matter haloes. *Astronomy & Astrophysics* 487, 2 (2008), 445–451. https://doi.org/10.1051/0004-6361:20079105

[6] Manuel Duarte and Gary A. Mamon. 2014. How well does the Friends-of-Friends algorithm recover group properties from galaxy catalogues limited in both distance and luminosity? *Monthly Notices of the Royal Astronomical Society* 440, 2 (03 2014), 1763–1778.

[7] Y. Feng and C. Modi. 2017. A fast algorithm for identifying friends-of-friends halos. *Astronomy and Computing* 20 (2017), 44 – 51. https://doi.org/10.1016/j.ascom.2017.05.004

[8] Juliana Freire, Philippe Bonnet, and Dennis Shasha. 2012. Computational Reproducibility: State-of-the-art, Challenges, and Database Research Opportunities. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD '12)*. ACM, New York, NY, USA, 593–596. https://doi.org/10.1145/2213836.2213908

[9] J. P. Huchra and M. J. Geller. 1982. Groups of galaxies. I - Nearby groups. *Astrophysical Journal* 257 (June 1982), 423–437. https://doi.org/10.1086/160000

[10] J. P. Huchra and M. J. Geller. 1982. Groups of Galaxies. I. Nearby groups. 257 (June 1982), 423–437. https://doi.org/10.1086/160000

[11] Oleg Igouchkine, Nick Leaf, 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)*. Association for Computing Machinery, New York, NY, USA, Article Article 8, 8 pages. https://doi.org/10.1145/3002151.3002163

[12] R. Kaehler. 2017. Massively parallel computation of accurate densities for N-body dark matter simulations using the phase-space-element method. *Astronomy and Computing* 20 (2017), 68 – 76. https://doi.org/10.1016/j.ascom.2017.05.005

[13] Steffen R. Knollmann and Alexander Knebe. 2009. AHF: Amiga's Halo Finder. *The Astrophysical Journal Supplement Series* 182, 2 (2009), 608. http://stacks.iop.org/0067-0049/182/i=2/a=608

[14] Otávio Migliavacca Madalosso, Andrea Schwertner Charão, Haroldo Fraga de Campos Velho, and Renata Sampaio da Rocha Ruiz. 2015. Implementação do algoritmo Friends of Friends de complexidade n*logn(n) para classificação de objetos astronômicos. In *Anais da XV Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul*. 245 – 248.

[15] Otávio Migliavacca Madalosso, Andrea Schwertner Charão, Haroldo Fraga de Campos Velho, and Renata Sampaio da Rocha Ruiz. 2015. Implementação do algoritmo Friends of Friends de complexidade n*logn(n) para classificação de objetos astronômicos. In *Anais da XV Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul*. 245 – 248.

[16] Surhud More, Andrey V. Kravtsov, Neal Dalal, and Stefan Gottlöber. 2011. The Overdensity and Masses of the Friends-of-Friends Halos and Universality of Halo Mass Function. *The Astrophysical Journal Supplement Series* 195, 1 (jun 2011), 4. https://doi.org/10.1088/0067-0049/195/1/4

[17] OpenACC. 2015. The OpenACC Application Program Interface. Available: http://www.openacc.org/sites/default/files/OpenACC_2pt5.pdf.

[18] Renata Sampaio da Rocha Ruiz. 2011. *Turbulência em cosmologia: análise de dados simulados e observacionais usando computação de alto desempenho*. Ph.D. Dissertation. Instituto Nacional de Pesquisas Espaciais, São José dos Campos. http://urlib.net/sid.inpe.br/mtc-m19/2011/04.27.19.23

[19] R. S. R. Ruiz, H. F. Campos Velho, and C. A. Caretta. 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 (2009).