MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

PLATAFORMA WEB PARA EXPERIMENTOS COM ALGORITMO FRIENDS-OF-FRIENDS PARALELO HÍBRIDO PARA CLASSIFICAÇÃO DE OBJETOS ASTRONÔMICOS

Ana Luisa Veroneze Solórzano (UFSM, Bolsista PIBIC/CNPq)
Haroldo de Campos Velho (LABAC/COCTE/INPE, Orientador)
Andrea Schwertner Charão (Informática-UFSM, Orientadora)

Relatório Final de Projeto de Iniciação científica (PIBIC/CNPq/INPE).

URL do documento original:
<http://urlib.net/>

INPE
São José dos Campos
2019

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3945-6923/6921
Fax: (012) 3945-6919
E-mail: pubtc@sid.inpe.br

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

PLATAFORMA WEB PARA EXPERIMENTOS COM ALGORITMO FRIENDS-OF-FRIENDS PARALELO HÍBRIDO PARA CLASSIFICAÇÃO DE OBJETOS ASTRONÔMICOS

Ana Luisa Veroneze Solórzano (UFSM, Bolsista PIBIC/CNPq)
Haroldo de Campos Velho (LABAC/COCTE/INPE, Orientador)
Andrea Schwertner Charão (Informática-UFSM, Orientadora)

Relatório Final de Projeto de Iniciação científica (PIBIC/CNPq/INPE).

URL do documento original:
<http://urlib.net/>

INPE
São José dos Campos
2019

AGRADECIMENTOS

Os autores agradecem ao PIBIC/CNPq/INPE pelo suporte na forma de bolsa de Iniciação Científica para a participante do projeto de nº 136023/2018-5.

RESUMO

Neste trabalho, é apresentado o desenvolvimento do projeto PIBIC/CNPq/INPE, intitulado "Plataforma Web para Experimentos com Algoritmo Friends-of-Friends Paralelo Híbrido para Classificação de Objetos Astronômicos". Apresenta-se o desenvolvimento da plataforma web multiusuário, permitindo a configuração e a execução remota de experimentos utilizando algoritmos da área de Astronomia, em especial o algoritmo para classificação de objetos astronômicos Friends-of-Friends (FoF). Apresenta-se estudos sobre o FoF, apontando versões do algoritmo com potencial de serem disponibilizados na plataforma em suas versões seriais e paralelas, realizando testes e experimentos para avaliação comparativa de desempenho entre eles. Por fim, estudou-se um possível ambiente de hospedagem da plataforma, o cluster Laquibrido, mantido pelo LAC/INPE.

Palavras-chave: Friends-of-Friends. Computação Híbrida. Plataforma Web.

WEB PLATFORM FOR EXPERIMENTS WITH THE FRIENDS-OF-FRIENDS HYBRID AND PARALLEL ALGORITHM FOR ASTRONOMIC OBJECTS CLASSIFICATION

ABSTRACT

This report presents the development of the "Plataforma Web para Experimentos com Algoritmo Friends-of-Friends Paralelo Híbrido para Classificação de Objetos Astronômicos" PIBIC/CNPq/INPE project. We present the multiuser web-platform developmet, that allows the configuration of the remote execution of algorithms in Astronomy, specially the Friends-of-Friends (FoF) algorithm, used for classification of astronomical objects. We present studies on FoF pointing algorithm's versions

with potential to be hosted on the platform in serial and parallel versions. We perform executions for performance analysis between them. Lastly, we present a study on a possible environment to host the platform, the Laquibrido cluster, maintained by LAC/INPE.

Keywords: Friends-of-Friends. Hybrid Computing. Web Platform.

LISTA DE ABREVIATURAS E SIGLAS

AHF	–	Amiga Halo Finder
CPU	–	Central Processing Unit
CUDA	–	Compute Unified Device Architecture
FoF	–	Friends-of-Friends
FPGA	–	Field-Programmable Gate Array
GPU	–	Graphics Processing Unit
MPI	–	Message-Passing-Interface
OpenACC	–	for Open Accelerators
OpenCL	–	Open Computing Language
OpenMP	–	Open Multi-Processing
OV	–	Observatório Virtual
RAM	–	Random Access Memory
UFSM	–	Universidade Federal de Santa Maria

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

SUMÁRIO

Pág.

1 INTRODUÇÃO

A área de Astronomia lida com grandes quantidades de dados observacionais obtidos através de simulações sobre o Universo. Devido aos avanços científicos e tecnológicos, os materiais utilizados para obtenção desses dados estão cada vez mais poderosos. Aliado a isso, os avanços na área de Computação, principalmente nos campos de processamento de dados e redes de computadores, permitiram a geração de máquinas potentes para armazenar e analisar grandes quantidades de dados.

Visto isso, surge a necessidade de criar soluções computacionais para processar as

saídas de simulações cosmológicas visando diferentes fins. Os chamados *halo-finders* são uma classe de algoritmos para processar dados de simulações sobre o Universo, utilizados para prever a formação de estruturas, como estrelas, galáxias e aglomerados de galáxias (MADSEN, 1996). Para isso, eles utilizam diferentes métodos computacionais e de agrupamento, como K-Means e AMR (Adaptive Mesh Refinement). Alguns *halo-finders* existentes são o RockStar(BEHROOZI et al., 2013), o Amiga Halo Finder(KNOLLMANN; KNEBE, 2009) e o Friends-of-Friends (Huchra; Geller, 1982), utilizado neste projeto.

Muitas vezes, esses *software* são desenvolvidos por especialistas em Astronomia com conhecimento em programação, que, geralmente, limitam-se a implementar soluções que tratem corretamente o problema, mesmo que sejam custosas. Entretanto, existem diversas abordagens e ferramentas computacionais que permitem explorar arquiteturas de alto desempenho, de modo a diminuir o tempo de execução dos algoritmos, permitindo processar mais dados em menos tempo.

Uma dessas abordagens é o uso de computação paralela que é a execução de tarefas ao mesmo tempo. Assim, uma solução possível é particionar trechos custosos do programa para serem executados em mais de um processador ao mesmo tempo, de forma paralela, diminuindo o seu tempo de execução em comparação a uma execução do código serial em apenas um processador, por exemplo, e aumentando o seu poder computacional. O AHF[1] e o Rockstar[2], disponibilizam implementações seriais e paralelas, porém, a execução paralela do AHF não explora o potencial de paralelismo do programa, sem grande aumento de desempenho em comparação a execução serial.

Outro problema, é que muitas vezes, esses *software* não estão disponíveis para download à comunidade. Supõe-se que isso ocorra por esses projetos serem financiados por iniciativas privadas, necessitando questões de licença. E, mesmo se disponíveis, como é o caso do AHF, eles carecem de informações técnicas sobre o desenvolvimento do projeto e métodos utilizados, impedindo análises reprodutíveis (FREIRE et al., 2012).

Uma alternativa para manter aplicações documentadas e facilmente disponíveis em plataformas *web*, é através de um Observatório Virtual (OV). OV, é uma estratégia para aproveitar os sistemas distribuídos de computadores integrados pela Internet, compartilhando recursos de *hardware* e *software* para armazenamento, processa-

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

mento e análise de dados. Os observatórios podem ser úteis para a comunidade científica e acadêmica, muitas vezes possibilitando execuções remotas dentro da plataforma.

Neste projeto, buscou-se implementar uma versão de OV para a execução de aplicações de alto desempenho da área de Astronomia. A plataforma foi desenvolvida utilizando o *framework* Django e foi baseada em uma implementação anterior de um Portal Web para execução remota do FoF (MADALOSSO et al., 2016a). O objetivo principal é fornecer um ambiente de acesso fácil, que permita aos usuários registrados realizarem execuções com diferentes arquivos de dados, obtendo como saída um arquivo com o resultado da execuções e um com o log sobre a execução. O OV foi desenvolvido como um código aberto, para que qualquer pessoa pudesse hospedar e adicionar seus próprios dados e aplicativos.

Os capítulos restantes deste relatório estão organizados da seguinte maneira: o Capítulo 2 apresenta os objetivos do projeto, o Capítulo 3 apresenta uma fundamentação teórica, visando contextualizar o projeto; o Capítulo 4 e 5 apresentam as atividades desenvolvidas durante o projeto, incluindo métodos, materiais utilizados durante o desenvolvimento, justificativas sobre a seleção dos mesmos e resultados; o Capítulo 6 apresenta os trabalhos publicados durante a pesquisa; e, por fim, o Capítulo 7 apresenta as conclusões e perspectivas do projeto.

2 OBJETIVOS DO TRABALHO

O objetivo principal do projeto é desenvolver um portal Web multiusuário, que permita a configuração e execução remota de experimentos de classificação de objetos astronômicos com algoritmo Friends-of-Friends (FoF) em versões seriais e paralelas.

A primeira etapa do trabalho consiste em estudos sobre as versões mais recentes do FoF e de outros algoritmos de agrupamento, que possam inspirar a otimização do FoF em sua versão serial. Após, pretende-se realizar a implementação de uma nova versão do algoritmo para um ambiente de computação híbrida composto por CPU e por GPU utilizando o padrão OpenACC.

A segunda etapa consiste na continuação da plataforma Web para a execução de experimentos com o algoritmo FoF, implementada anteriormente utilizando o *framework* Django. Dentre os próximos passos na plataforma estão: configurar o ambiente para hospedagem dos algoritmos, adaptar a interface para seguir o modelo de

sites do INPE, configurar a geração dos arquivos de saída e configurar o portal para facilitar sua hospedagem no cluster Laquibrido.

Na terceira etapa, pretende-se coletar métricas de execuções das diferentes versões do FoF em ambientes de computação de alto desempenho, incluindo execuções no cluster Laquibrido do LAC/INPE e por meio da plataforma Web.

Por fim, pretende-se tornar o portal disponível, hospedado no cluster Laquibrido do LAC/INPE, para a comunidade de astronomia/astrofísica; Esta será uma ação da Departamento de Linguagens e Sistemas de Computação (DLSC/UFSM) junto ao Instituto Nacional de Ciência e Tecnologia de Astrofísica (INCT-A, CNPq). Como resultados do trabalho, espera-se incentivar a implementação de novas versões do algoritmo e a utilização de novas tecnologias Web para implementação na plataforma.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 Friends-of-Friends

Simulações de N-corpos têm sido utilizadas para promover diversos avanços na área de Astronomia. Um exemplo disso, é o processamento dos dados resultantes dessas simulações para compreender o processo de formação e evolução de estruturas do Universo. Essa situação tem papel fundamental no estudo da evolução cósmica em tópicos como a distribuição de matéria escura em grande escala, a formação de halos de matéria escura, e a formação e evolução de galáxias e aglomerados de galáxias (BERTSCHINGER, 1998; Efstathiou et al., 1985; JENKINS et al., 1998).

Uma das abordagens utilizadas para identificar os chamados *halos* de matéria escura consiste em usar o algoritmo de percolação Friends-of-Friends (Huchra; Geller, 1982). O FoF é um algoritmo de percolação amplamente utilizado devido à sua simplicidade de implementação, recebendo como parâmetro um arquivo de entrada e um valor para o raio de percolação (MORE et al., 2011b; DUARTE; MAMON, 2014). Ele é utilizado para identificar estruturas na distribuição de matéria escura no Universo, com base na proximidade física entre partículas que a compõe. No entanto, para resoluções mais altas, o processamento de conjuntos de dados completos pode trazer desafios computacionais.

A ideia básica deste algoritmo é a seguinte: considere uma esfera de raio R ao redor de cada partícula do conjunto total; se dentro desta esfera existirem outras partículas,

elas serão consideradas pertencentes ao mesmo grupo e serão chamadas de amigas. Em seguida, toma-se uma esfera ao redor de cada amiga e o procedimento continua usando a regra "qualquer amigo de meu amigo é meu amigo". O procedimento finaliza quando nenhuma amiga nova puder ser adicionada ao grupo.

A implementação original do FoF possui complexidade $O(N^2)$, acarretando em alto custo computacional para processar simulações completas de N-corpos, mais próximas da realidade, com arquivos de dados contendo informações sobre bilhões de partículas. Assim, mostra-se necessária a implementação de versões mais eficientes do FoF, melhorando o seu desempenho. No trabalho de Ruiz et al. (RUIZ et al., 2009), apresentou-se uma versão paralela do algoritmo FoF utilizando o padrão MPI para comunicação entre os 27 processadores do *cluster* utilizado, alcançando uma aceleração significativa para esta arquitetura com memória distribuída. Como continuidade do trabalho de Ruiz et al., explorou-se a possibilidade de redução do custo computacional do algoritmo, sugerida em trabalhos como (SPRINGEL et al., 2000), criando-se um novo algoritmo FoF sequencial com complexidade $O(NlogN)$ (MADALOSSO et al., 2015), inspirado no método proposto por (BARNES; HUT, 1986). Além disso, também explorou-se o uso de arquitetura *multicore* implementando o FoF com OpenMP (BERWIAN et al., 2017),

O uso de computadores híbridos vêm se popularizando, visto que apresentam benefícios como proporcionar rapidez, permitindo múltiplas oportunidades de paralelismo em diferentes unidades de processamento trabalhando em conjunto. Atualmente, uma plataforma de aceleração bastante utilizada é a GPU, mesmo existindo outras como FPGAs e Intel® Xeon Phi®. Dentre as diversas ferramentas para paralelização de programas com aceleradores estão CUDA, OpenCL e OpenACC. Visto isso, resolveu-se explorar a implementação do algoritmo FoF em GPU utilizando OpenACC (NVIDIA, 2012).

3.2 Observatórios Virtuais

O campo da Astronomia lida com uma grande quantidade de dados observacionais e simulados. Existem vários algoritmos e *software* para processar esses dados, revelando informações sobre a dinâmica das estrelas, o centro das galáxias ou até mesmo a estrutura do Universo. Os Observatórios Virtuais são iniciativas internacionais para fornecer informações sobre arquivos de dados astronômicos em ambientes facilmente acessíveis por toda a comunidade.

OV preocupam-se em aproveitar os sistemas distribuídos de computadores integra-

dos pela Internet, compartilhando recursos de *hardware* e *software* para armazenamento, processamento e análise de dados. Com isso, eles são capazes de trazer benefícios significativos para a comunidade de pesquisa em astronomia e para qualquer outro grupo interessado, como estudantes e educadores.

Uma plataforma com interface Web pode ser aplicada neste contexto, ofereceria um recurso útil a pesquisadores interessados em obter conjuntos de dados, servindo também como plataforma de experimentação com diferentes versões de *software* disponíveis para execuções. Existem alguns projetos de OV disponíveis, como por exemplo:

- Observatório Virtual Europeu[1] (EURO-VO): Destina-se a implantar um OV europeu, fornecendo aplicativos científicos que podem ser usados para realizar execuções. No entanto, o seu registro não é público e, por isso, não pôde ser testado.

- Observatório Virtual Chileno[2] (ChiVO): Foi desenvolvido no Chile devido a grande quantidade de dados obtidos pelos observatórios astronômicos no país que precisavam ser armazenados, processados e analisados usando ferramentas e algoritmos. O ChiVO visa ser útil para astrônomos e estudantes, especialmente de escolas de áreas rurais. Alguns dos *software* oficiais disponíveis são: uma biblioteca Python para empilhar imagens de objetos astronômicos, um pacote Python para fornecer uma interface fácil e operações de entrada e saída para a utilização de bibliotecas astronômicas sobre notebooks Jupyter[3], um serviço virtual para gerar dados espectroscópicos sintéticos que podem ser usados para avaliar algoritmos avançados de computação para astronomia, e um pacote com algoritmos para astrônomos, incluindo alguns produzidos pelos pesquisadores do ChiVO, e suas versões utilizando computação de alto desempenho. Como o EURO-VO, o registro não é público e o serviço não pode ser utilizado fora do Chile.

- Virtual Astronomical Observatory[4] (VAO): A National Science Foundation dos EUA junto com a NASA financiaram esse projeto internacional para integrar grandes conjuntos de dados, ferramentas e serviços de astronomia à comunidade global. O VAO procura especialmente colocar ferramentas

[1] https://aladin.u-strasbg.fr/aladin.gml
[2] https://chivo.cl/
[3] https://jupyter.org/
[4] http://www.virtualobservatory.org/

astronômicas eficientes nas mãos de astrônomos, estudantes, educadores e líderes de alcance público dos EUA. Algumas das ferramentas disponíveis são executadas online e outras precisam ser baixadas e instaladas localmente. Além disso, disponibilizam um banco de dados para serem utilizados nas execuções, no qual o usuário pode ver o andamento de sua tarefa e baixar os arquivos utilizados.

4 FRIENDS-OF-FRIENDS

4.1 Friends-of-Friends - versão serial

Antes de abordar a nova versão do FoF utilizando OpenACC para um ambiente de computação híbrida, apresenta-se a versão serial do algoritmo, utilizada como base do trabalho. O algoritmo serial inicia recebendo por linha de comando o arquivo de dados de entrada e o raio de percolação a ser utilizado no cálculo da distância. O tamanho do raio(l) é geralmente dado por b vezes a separação média entre as partículas, com os valores de b e l dependendo da natureza da aplicação (CARETTA et al., 2008). Quanto maior for l, menor será o contraste de densidade entre partículas e maior o número de partículas anexadas aos grupos (RUIZ, 2011). Geralmente, o valor usado de b é 0.2 para identificar halos de aglomerados (LACEY; COLE, 1994; MORE et al., 2011a). Neste artigo, assim como em Caretta e Ruiz, utilizou-se $l = 0.1$ *Mpc* para halos de galáxias.

O algoritmo inicia lendo o valor do raio de percolação e as informações do arquivo de entrada. O *array* `igru`, armazenará os grupos de partículas, e os arrays `x`, `y` e `z`, armazenam a posição de cada partícula em um espaço tridimensional. O processamento principal do FoF é a etapa de classificação composta por três laços `for` aninhados apresentada em Algoritmo 4.1.

O laço mais externo percorre todas as partículas criando um novo grupo, caso a partícula atual ainda não esteja classificada. Caso contrário, passa-se para a próxima partícula sem classificação. Para novos grupos, calcula-se a distância entre uma partícula e as outras usando a fórmula da distância `d =` $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$. Se a distância entre elas é menor do que o raio de percolação definido pelo usuário, a nova partícula é considerada sua amiga e classificada como pertencente ao mesmo grupo.

```
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;
        }
      }
    }
  }
}
```

Algoritmo 4.1 - FoF sequencial com complexidade $O(N^2)$

4.2 Friends-of-Friends - versão paralela com OpenACC

4.2.1 OpenACC

O OpenACC (OPENACC, 2015) é um padrão para programação paralela proposto em 2011, em uma parceria entre NVIDIA, Cray, Portland Group(PGI) e CAPS Enterprise. Ele foi projetado para facilitar a programação em sistemas híbridos de computação, que contém pelo menos uma CPU e uma GPU. Para isso, o OpenACC disponibiliza diretivas de compilação para expressar a execução no acelerador, permitindo aplicar abordagens paralelas, definidas diretamente pela diretiva utilizada. Atualmente, ele é suportado em arquiteturas como X86 & Power CPUs e GPUs da NVIDIA.

OpenACC tem suporte a códigos nas linguagens C, C++ e Fortran para rodar em um sistema composto por um host(CPU) e um dispositivo acelerador(GPU), sem precisar de grande conhecimento sobre sua arquitetura, aplicando apenas diretivas de compilação. Para utilizar o OpenACC, deve-se ter um compilador que seja capaz de reconhecer as diretivas e gerar código otimizado para diferentes arquiteturas. O compilador utilizado neste trabalho é o Portland Group(PGI) edição Community (NVIDIA, 2012).

As diretivas de compilação utilizadas com o OpenACC facilitam a portabilidade do

código, permitindo criar programas de alto nível que implementam interações entre o host e o acelerador, com poucas modificações em relação ao código original. Além disso, o padrão abstrai do programador responsabilidades como a inicialização explícita do acelerador, o gerenciamento de dados a serem transferidos e configurações sobre a transferência de programas entre os dispositivos. Todos esses detalhes estão implícitos no modelo de programação e são gerenciados pelo compilador que suporta o OpenACC, mas alguns podem ser configurados de acordo com a preferência do usuário.

4.2.2 Primeira Versão

Trabalhos anteriores apresentaram a paralelização do FoF com MPI e OpenMP (MADALOSSO et al., 2015; BERWIAN et al., 2017). Entretanto, a paralelização do algoritmo para um ambiente de computação híbrida ainda não havia sido explorado. Assim, buscou-se paralelizar o FoF para um ambiente composto por uma CPU e uma GPU, de forma a utilizar o dispositivo acelerador para melhorar o desempenho sobre a sua execução. Dentre as ferramentas disponíveis para paralelização de programas com GPU como OpenCL e CUDA, utilizamos o padrão OpenACC, que possibilita a distribuição da computação em um ambiente híbrido através do uso de diretivas de compilação inseridas no código.

A metodologia adotada consistiu em realizar o mínimo possível de alterações no código original do algoritmo para paralelização em GPU. Após diversos experimentos, realizou-se uma análise de desempenho do algoritmo paralelizado com OpenACC utilizando como ferramenta auxiliar o *profiler* nvprof [1], um analisador de perfil de execução da NVIDIA.

A paralelização foi feita no laço mais interno do Algoritmo 4.1, visto que ele não possui dependência de dados. Para isso, foi utilizada a diretiva `#pragma acc parallel`, que descreve uma região do código a ser acelerada com os *kernels* da GPU. Notou-se que a variável `dist`, que armazena o valor da distância entre duas partículas, deve ser privada para evitar valores inconsistentes.

Como os comandos podem ler e escrever no vetor de grupos, chamado `igru`, encontrou-se duas possibilidades semelhantes para garantir a consistência dos dados armazenados nele: (i) usando a diretiva `#pragma acc copy`, que copia o vetor da CPU para a GPU antes de entrar na região paralela, e da GPU para a CPU antes de sair, ou (ii) alocando o vetor no acelerador antes de entrar no laço e usando

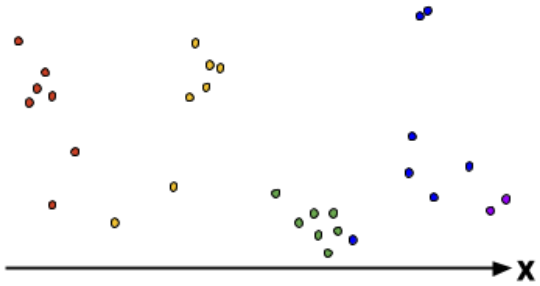
[1] https://docs.nvidia.com/cuda/profiler-users-guide/index.html

as diretivas de compilação `#pragma acc update device` e `#pragma acc update self`, que copiam apenas os valores modificados no vetor da memória local para o dispositivo(GPU) e do dispositivo para a memória local. Para os outros vetores, que são apenas lidos, como o `x, y` e `z`, usamos o `#pragma acc data copy` antes do laço mais externo, para copiar os valores de leitura para a GPU apenas uma vez.

4.2.3 Segunda Versão

O objetivo desta implementação foi melhorar o desempenho do FoF em GPU, considerando os pontos negativos da implementação anterior como: alto tempo de execução e alto tempo de processamento gasto com a transferência de dados entre dispositivos. Assim, esperou-se otimizar o uso dos recursos computacionais, minimizando o tempo gasto com transferências entre os dispositivos.

Figura 4.1 - FoF com OpenACC - entrada

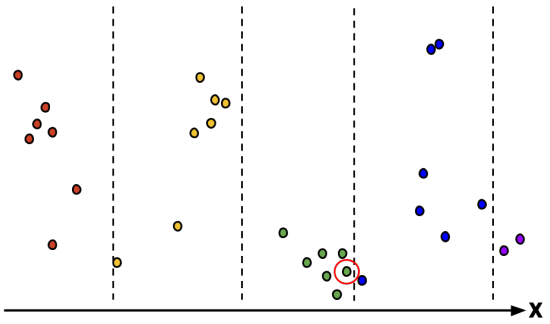


Exemplo de entrada, considerando observação das partículas pelo eixo x.
Fonte: Produção do autor.

A estratégia utilizada, baseou-se no ambiente de computação disponível, composto por uma CPU e uma GPU. O principal desafio desta implementação foi encontrar um esquema de balanceamento de carga eficiente para qualquer GPU. Porém, esta tarefa se mostrou mais complexa visto que o FoF é sensível às posições das partículas, ou seja, se a maioria das partículas pertencerem a poucos grupos, o processamento será muito mais rápido e o processador ficará ocioso, enquanto que se a maioria das partículas tiverem distâncias pequenas entre elas, pertencendo aos grupos, o processador terá mais trabalho, pois terá mais partículas para aplicar a abordagem principal do FoF vista no Algoritmo 4.1.

Assim, dada uma entrada como 4.1, com o conjunto de partículas apresentadas

Figura 4.2 - FoF com OpenACC - divisão



Exemplo da estratégia utilizada para a paralelização do FoF, subdividindo os dados de entrada em conjuntos, e enviando cada conjunto para ser processado em um dos núcleos da GPU. Esta divisão foi realizada com base na coordenada `x` utilizando 5 *kernels*.

Fonte: Produção do autor.

considerando o eixo de orientação espacial `x`, A figura 4.2 apresenta a proposta de construir uma tesselação das partículas considerando uma de suas posições tridimensionais. Para tratar todas as partículas igualmente, primeiro ordenou-se o arquivo de entrada por uma coordenada e dividiu-se o conjunto total de entrada em porções com igual número de partículas. Cada porção é então atribuída a um *kernel* de GPU e dentro de cada *kernel* o FoF é executado sequencialmente, conforme Algoritmo 4.1. O número de porções que irão ser definidas é obtida pelo usuário por entrada padrão do programa.

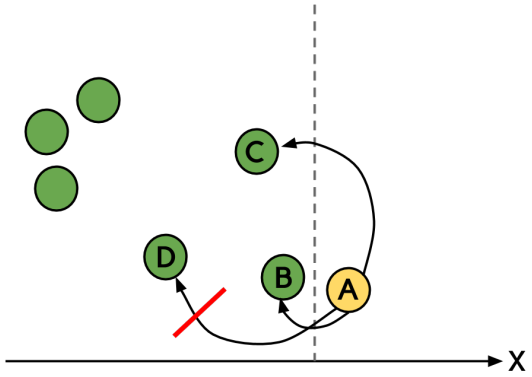
Após os *kernels* terminarem de processar o FoF sobre o seu conjunto de partículas, o vetor com os índices dos grupos é retornado à CPU. Com isso, haverá mínima migração de dados entre os dispositivos: no começo, ao enviar as partículas para os *kernels*, e ao final, ao retornar os índices de grupos atualizados para a CPU. Porém, apesar de realizar os cálculos em GPU corretamente, pode-se observar casos em que existem pares de partículas localizadas em zonas de fronteira, que são tratadas por *kernels* distintos, mas que pertencem ao mesmo grupo, conforme observado na Figura 4.2, onde a partícula circulada é o elo de ligação entre a partícula do outro grupo. Estes casos causaram atraso na implementação, pois não conseguia-se chegar a resultados equiparáveis aos obtidos com a versão serial. Assim, depois que a CPU obtém todos valores calculados em GPU, ela realiza um pós-processamento para tratar esses casos.

Para tratar esse problema, a primeira ação realizada foi ordenar as partículas do

arquivo de entrada por uma de suas coordenadas de posição logo após serem lidas. Esta ordenação facilitou o pós-processamento e também diminuiu o tempo gasto com o FoF, visto que partículas mais próximas em relação a um eixo tem mais chances de estarem conectadas por um valor de distância menor do que o raio de percolação. Após as execuções e classificações dos grupos realizadas por cada *kernel* da GPU, iniciou-se o pós-processamento em CPU.

No pós-processamento, calcula-se as distâncias de duas a duas partícula na zona de fronteira, considerando o raio de percolação como o limite de distância entre elas, e visto que estão ordenadas por um eixo, a partícula escolhida para comparação segue a ordenação anterior. A ideia básica é: selecionar a partícula mais próxima da zona de fronteira de um *kernel*, comparar a sua distância com as partículas na zona de fronteira tratadas em outro *kernel* até encontrar uma partícula com distância maior do que o raio de percolação, parando a comparação, visto que todas outras partículas estarão mais distantes do que o raio, ou seja, de grupos distintos.

Figura 4.3 - FoF com OpenACC - pós-processamento



Pós-processamento realizado em CPU, no qual compara-se a distância entre partículas nas zonas de fronteira, juntando grupos classificados como distintos por *kernels* diferentes, mas que estão próximos o suficiente para serem considerados do mesmo grupo.

A Figura 4.3 apresenta um exemplo em que se considera a partícula A como a partícula de comparação. Primeiro, compara-se a sua distância com a partícula B, assume-se que a distância entre elas é menor do que o raio e as classifica como pertencentes ao mesmo grupo; Após, compara-se a distância da partícula A em relação a partícula C, realizando o mesmo processo. Porém, ao comparar com a partícula D, o programa observa que a distância entre elas é maior do que o raio, assim, o pós-processamento para esta fronteira é concluído.

Observa-se que quando uma partícula na zona de fronteira é associada a outra de um grupo distinto, é necessário alterar o grupo de partículas associadas a ela, de modo a classificá-las como do mesmo grupo. Essa reclassificação gerou uma nova varredura em parte das partículas dos kernels a serem comparados, no qual usou-se um loop `for` paralelizado com o padrão OpenMP (DAGUM; MENON, 1998). Os códigos-fonte deste novo algoritmo podem ser acessados em: `https://github.com/anaveroneze/laquibrido/tree/master/ccis2019`.

4.3 Ambientes de execução

Durante a execução do projeto foram explorados diferentes ambientes de computação para a realização de testes e experimentos sobre os algoritmos. A seguir, apresenta-se os ambientes utilizados bem como desafios enfrentados nas execuções nos mesmos.

4.3.1 Lsc5

O ambiente de execução inicial, utilizado para os primeiros testes da nova implementação foi um servidor do Laboratório de Sistemas de Computação da UFSM chamado de *lsc5* composto por:

- Processador Intel R Xeon R E5620 com quatro cores físicos e oito virtuais; cache L1 de 32KB, cache L2 de 256KB e cache L3 de 12MB, e 12 GB de memória;
- GPU NVIDIA GeForce GTZ Titan X;
- Sistema operacional: Debian 9, versão do Linux 4.9.0-2;
- Compilador gcc versão 7.3.0.

Para executar a paralelização com OpenACC, foi preciso instalar localmente o compilador da Portland Group (PGI) edição Community (NVIDIA, 2012) versão 17.4-0. Entretanto, esta máquina teve de ser desligada por problemas técnicos na sala onde fica hospedado. Assim, utilizou um servidor da UFRGS para os próximos testes.

4.3.2 Hype

O servidor da UFRGS utilizado foi a *hype*, utilizando-se um nó, que é composto por:

- Dois processadores Intel® Xeon® E5-2650 v3 de dez cores físicos e vinte virtuais;

- Duas GPUs NVIDIA Tesla K80;
- Sistema operacional: Ubuntu 18.04.1 LTS;
- 128 GB DDR4 RAM;
- O compilador da PGI utilizado foi uma nova versão, 18.4-0.

4.3.3 Laquibrido

Outro ambiente de execução foi o cluster LaquiHíbrido, mantido pelo LAC/INPE, e adquirido com recursos financeiros do governo brasileiro. Para acessar o *cluster*, foram seguidas etapas de solicitação de acesso remoto via VPN. O processo durou cerca de um mês entre solicitação e encaminhamento de formulário. Após conseguir acesso ao *cluster*, começou-se a explorar o seu ambiente de execução. Nesta etapa notou-se que não havia documentação sobre o ambiente, gerando confusão sobre quais os recursos disponíveis no servidor, e em quais nós da máquina. Visto isso, a seção 5.2 apresenta estudos feitos visando entender melhor o ambiente.

Outro problema foi que muitas das ferramentas necessárias para as execuções do FoF (compilador para OpenACC) e para a hospedagem do portal (instalação do framework Django, de um gerenciador de pacotes para instalação de *software* em Python e de acesso a serviços relacionados à rede) não poderiam ser instalados devido a forte dependência dos responsáveis pelo seu gerenciamento. O escalonador de processos utilizado no *cluster* é o Slurm[2]. Esta ferramenta havia sido utilizada no acesso à *hype*, portanto, não gastou-se muito tempo para compreender sua utilização.

O compilador da PGI, para OpenACC foi instalado após solicitação, porém, não conseguiu-se realizar execuções do código por não conhecer os nós com GPUs, e quais as versões desses dispositivos. Para investigar essa situação e compreender se o erro era proveniente da instalação do compilador ou do ambiente alocado para execução, decidiu-se lançar execuções de um programa simples paralelizado para GPU com CUDA.

Foram encontradas três instalações do CUDA no diretório de acesso público da máquina: 9.1, 9.2 e 10. Todos testes aqui apresentados tentaram utilizar todas as versões. A primeira abordagem foi realizar a compilação e a execução no nó principal do *cluster*, porém ocorreu um problema do compilador não encontrar o caminho para

[2]https://slurm.schedmd.com/documentation.html

o comando *bin2c*, necessário de acesso de acesso privilegiado para poder editar essas configurações.

A segunda abordagem foi compilar o programa em um nó do *cluster*, porém ocorreu um erro de compilação alertando sobre não encontrar uma das funções da biblioteca padrão do C do GNU Linux, **libc**[3]. Assim, pensou-se em utilizar uma nova abordagem.

A terceira abordagem foi realizar a instalação local do CUDA sem privilégios de usuário, seguindo um tutorial disponível [4]. Porém, visto que a instalação era apenas dos arquivos executáveis para compilação, ela estava suscetível a enfrentar problemas de compilação devido a incompatibilidade do compilador com o *driver* instalado sobre o dispositivo.

A compilação local foi concluída, após editar variáveis de ambiente para os caminhos de execução corretos, porém, apresentou erros sobre a incompatibilidade com o *driver* do dispositivo, o que era esperado. A compilação nos nós do *cluster*, através do *script* para o Slurm não foi realizada, pois mesmo editando as variáveis de ambiente que determinam os caminhos corretos, a compilação ainda teve dificuldade de encontrar o diretório de destino.

4.3.4 Google Colab

Visando facilitar as execuções, sem dependências de acessos externos, utilizou-se o Google Colab[5]. O Google Colab é um serviço de nuvem gratuito voltado principalmente a pesquisas de Aprendizado de Máquina e Inteligência Artificial. Por isso, ele oferece um ambiente de execução composto por CPU e GPU, suporte a diversas linguagens de programação, como C e Python, suporte a comandos bash e demais bibliotecas e compiladores.

Apesar de uso claro e eficiente, algumas informações sobre o ambiente de execução não são acessíveis ao usuário padrão. Além disso, o compilador para OpenACC não é oficialmente suportado, o que não gerou problema pois o ambiente permitiu sua instalação local, apenas para o usuário utilizado. O ambiente é composto por:

- Intel(R) Xeon(R) (não detalhado);

[3] http://man7.org/linux/man-pages/man7/libc.7.html
[4] https://www.pgroup.com/userforum/viewtopic.php?t=6181
[5] https://colab.research.google.com/

- GPU NVIDIA Tesla K80;
- Sistema Operacional: Ubuntu 18.04.2 LTS.

4.4 Experimentos

Visando validar as execução da nova versão do FoF para um ambiente de computação híbrida, foram realizados testes sobre a execução da Primeira Versão 4.5 e sobre as execuções da Segunda Versão 4.5. Todos os testes apresentados aqui foram realizados na hype 4.3.2.

O número de *kernels* foi definido pelo usuário pela entrada padrão e o número de *threads* definido pela variável de ambiente OMP_NUM_THREADS. Foram realizadas 30 execuções com três amostras de dados observacionais do Consórcio Virgo[6]. As amostras tinham 65.536 partículas (Arquivo 1), 174.761 (Arquivo 2) e 249.420 (Arquivo 3), utilizando-se 200 *kernels* para o Arquivo 1, 300 para o Arquivo 2 e 600 para o Arquivo 3. Utilizou-se raio de percolação 0.1 conforme utilizado em Caretta et al. para processamento dos mesmos dados de simulação do Consórcio Virgo sobre galáxias.

4.5 Resultados

Os resultados obtidos pelas execuções sobre a Primeira Versão do FoF com OpenACC mostraram que, o algoritmo não explora todo o potencial de uso da GPU apenas com a paralelização no laço mais interno. Demandando um tempo excedente de execução, pois processa partes do programa na GPU e partes na CPU alternadamente.

A Tabela 4.1 apresenta as médias dos tempos de execução e o *speedup*, em relação ao tempo médio de execução do FoF serial, para a Segunda Versão em comparação a melhor abordagem da Primeira Versão . Desconsiderou-se o tempo de leitura da entrada em ambos os casos, pois sua implementação permaneceu a mesma, e se considerou a soma do tempo de execução do FoF em GPU e do pós-processamento em CPU para a nova abordagem.

A Figura 4.4 apresenta de forma mais clara a diferença do desempenho das duas versões. Destaca-se o *speedup* de quase 49 para a Segunda Versão 4.5 para o segundo arquivo de entrada, onde se encontrou um caso que explorou de maneira adequada

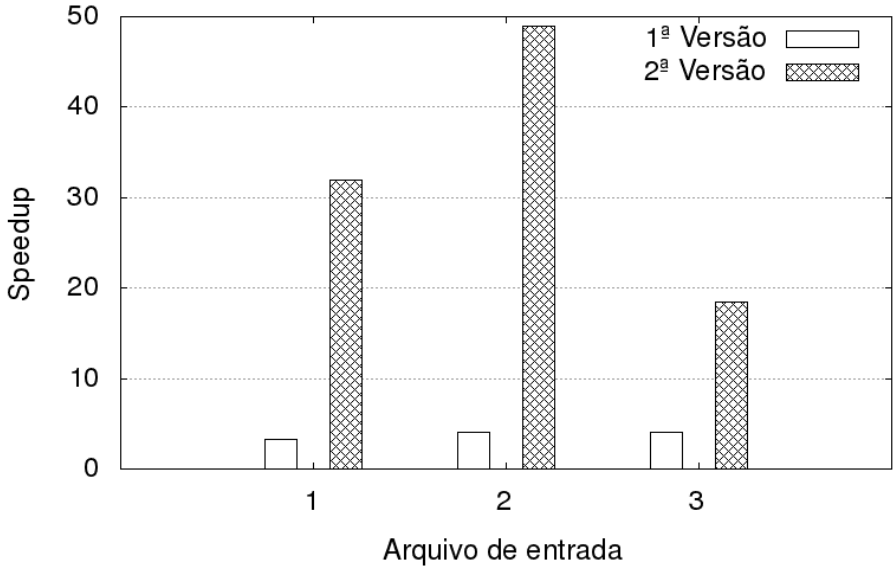
[6]http://www.mpa-garching.mpg.de/Virgo/data_download.html

Arquivos de Entrada	Tempos de Execução (s)		Speedup	
	Segunda Versão	Primeira Versão	Segunda Versão	Primeira Versão
1	1,43	13,79	31,96	3,31
2	6,27	76,61	48,97	4,00
3	290,17	1.335,67	18,52	4,02

Tabela 4.1 - Tempos de execução e *speedup* para as três amostras de dados

os *kernels* da GPU definidos para realizar o processamento. Para os outros arquivos, a nova versão teve desempenho superior à anterior.

Figura 4.4 - *Speedup* das implementações com OpenACC



O gráfico mostra o *speedup* da Primeira Versão do FoF em comparação a Segunda Versão, calculando o *speedup* em relação a execução serial do algoritmo.

Fonte: Produção do autor.

A partir dos resultados, notou-se ganho de desempenho da nova versão em relação à serial e em relação à versão anterior. Utilizando o nvprof, observou-se que, de fato, a nova versão faz melhor uso da GPU, utilizando, para um arquivo com 249.420 partículas, 99.8% do tempo de execução do algoritmo no acelerador para processar o FoF, e apenas 0.15% do tempo com cópias de dados entre CPU e GPU.

Para entender melhor esses resultados, utilizou-se o nvprof, que é executado por linha de comando e apresenta uma visão detalhada sobre o uso dos kernels da GPU

incluindo informações sobre tempo gasto com cópias entre dispositivos. Na opção utilizada, obteve-se o tempo total de execução e a porcentagem deste tempo que foi gasto em cada *kernel*.

Desempenho na GPU: Para a Primeira Versão4.5, os núcleos da GPU foram mais utilizados para realizar a comunicação entre o host (CPU) e o dispositivo (GPU), gastando aproximadamente 46% do tempo total no acelerador para transferir dados do host para o dispositivo e 45% para transferências do dispositivo para o host, deixando apenas 6% a 8% para executar o FoF. A segunda versão passou a maior parte do tempo transferindo dados de dispositivo para host, com 51 % para o primeiro arquivo até 77 % para o terceiro arquivo. Para transferir de um host para outro, gastou cerca de 39 % para o primeiro arquivo e 11 % para o terceiro. Com isso, a segunda versão usou mais tempo para processar o FoF na GPU: de 9,87 % para 10,92 %. Para a segunda abordagem, quase todo o tempo em GPU foi usado para processar o FoF (de 97 % para 99 %), uma vez que foi executado inteiramente em GPU, aproveitando ao máximo o dispositivo.

Visto que o FoF é um algoritmo sensível às informações dos dados de entrada, a ordenação das partículas no início do programa gerou uma classificação distinta de grupos em comparação à realizada pelo FoF serial. Isso acontece porque o algoritmo original não trata de casos de *relabel*, como relatado em (MADALOSSO et al., 2015), e por isso os valores podem divergir. Para garantir a corretude desta nova implementação, realizou-se execuções com entradas controladas que geram situações de *relabel*. Assim, os resultados foram coerentes com o esperado, sendo encontrados para o Arquivo 1 46.382 grupos na versão serial e 45.715 na nova versão com OpenACC que realizou a junção de grupos nas *zonas de fronteira*, 113.660 e 111.313 para o Arquivo 2, e 157.682 e 154.743 para o Arquivo 3.

A paralelização do pós-processamento com OpenMP não foi utilizada nessas execuções, pois utilizando um raio pequeno de 0.1, menor será o número de partículas associadas aos grupos e assim menor o número de partículas a serem reclassificadas, tornando essa paralelização custosa. Porém, com testes hipotéticos para raio de 1.7, a execução do Arquivo 2 com 200 *kernels* trouxe um ganho de desempenho de 1.3 com 4 *threads* no pós-processamento em relação à execução serial desta etapa.

Os resultados mostraram que é possível fazer bom uso da GPU para o processamento do FoF, obtendo-se ganho de desempenho em comparação à versão serial e à versão anterior com OpenACC. Porém, devido ao FoF serial não tratar de casos

de *relabel*, a consistência dos grupos definidos em cada versão deve ser melhor analisada realizando-se mais testes com entradas controladas inclusive em comparação com outras versões do algoritmo.

4.6 Estudos de trabalhos relacionados

Kaehler usa uma abordagem semelhante de divisão da entrada em conjuntos, para paralelizar simulações de matéria escura de N-corpos em GPU, mas usando CUDA em um *cluster* com 256 GPUs (KAEHLER, 2016). Em CPU, a paralelização utilizou OpenMP e MPI para comunicação entre nodos. Para melhorar o desempenho da aplicação, em especial quanto ao gerenciamento de memória para armazenar informações dos arquivos de dados, os autores utilizaram o método de *oct-tree* e dois métodos para balanceamento de carga, que passam porções de dados iguais para cada nó do cluster, minimizando a transferência de dados e ocupando melhor a memória dos dispositivos. Um dos métodos utiliza reduções sobre os nodos do cluster, para obter ao final o valor total das massas das partículas e o outro método transfere em tempo de execução a massa das partículas calculadas em cada nó para um único nó do *cluster*, que iria realizando a soma total. Assim, o autor se preocupou em realizar o balanceamento de carga em três níveis: entre os nós computacionais, entre as *threads* da CPU em cada nó e entre os *kernels* da GPU, lançados pelas *threads* da CPU.

Feng et al. apresenta um algoritmo FoF com tempo de execução reduzido em comparação ao FoF serial, usando uma estrutura de árvore hierárquica e uma função de correlação para associar partículas. Este algoritmo reduz o número de operações, evitando visitar mais de uma vez a mesma "vizinhança" de partículas, e reduz a complexidade de tempo utilizando esquema de árvore, agrupando as partículas em cada árvore e, após, juntando as árvores que podem ser classificadas como pertencentes ao mesmo grupo. Os autores utilizarem estruturas de dados conhecidas(listas e grafos) e visarem obter mínimo *overhead* de memória, mas não consideraram a utilização de abordagens paralelas (Feng; Modi, 2017).

O FoF sequencial com complexidade de tempo $O(N)$ é altamente exigente em termos de CPU, e por isso, alunos da UFSM em parceria com o INPE, desenvolveram versões do algoritmo FoF paralelizados para CPU utilizando MPI e OpenMP (RUIZ et al., 2009; BERWIAN et al., 2017). Apesar de usar abordagens paralelas, o grupo de pesquisa ainda não havia utilizado dispositivos aceleradores para o FoF. Visto isso, nesta pesquisa, apresentou-se uma nova abordagem, paralelizando o algoritmo FoF para GPU utilizando OpenACC, a fim de desenvolver um algoritmo mais rápido

para a execução de conjuntos completos de dados de simulações de matéria escura.

5 PORTAL WEB

O desenvolvimento do portal baseou-se em uma implementação anterior de um portal web para a execução remota do FoF (MADALOSSO et al., 2016a). Durante o desenvolvimento do portal surgiu a colaboração com um aluno da Engenharia da Computação na UFSM, Vinícius Monego. O seu estudo era sobre implementação de algoritmos para tratamento de imagens utilizando a técnica *wavelets*. Assim, para a continuação do desenvolvimento do portal, inicialmente foram realizadas modificações em sua implementação para agora, seguir o modelo de um OV, e não ser um portal para a execução de um algoritmo específico.

Atualmente, a plataforma oferece as versões seriais e paralelas do FoF apresentadas na Tabela 5.1 e algoritmos para a restauração de imagens usando filtragem *wavelet*. A plataforma foi implementada para ser hospedada em ambientes de computação *multicore* visando explorar arquiteturas paralelas. Os usuários podem definir seus parâmetros de execução, seus arquivos de entrada e, ao final de cada execução, o usuário pode realizar o *download* do arquivo de saída e do arquivo de log.

Tabela 5.1 - Versões do algoritmo Friends-of-Friends disponíveis no portal

Complexidade	Serial	Paralelo	Executa em	Ferramentas
$O(NlogN)$	X		CPU	
$O(N^2)$	X		CPU	
$O(N^2)$		X	CPU	OpenMP
$O(N^2)$		X	CPU + GPU	OpenACC

5.1 Ferramentas utilizadas

O portal web foi desenvolvido utilizando três ferramentas principais:

- Django Web Framework[1]: possibilita o desenvolvimento de aplicações web utilizando a linguagem de programação Python. Ele foi utilizado pois oferece ampla documentação e acesso a diversas extensões que facilitam o desenvolvimento do projeto, como mapeamento objeto-relacional e um padrão de projeto Model-View-Template(MVT). O Model é responsável pela interação com o banco de dados, o View é responsável pela lógica utilizada

[1] https://www.djangoproject.com/

sobre os dados, e o Template é responsável pela interação do usuário com a aplicação.

- Celery[2]: um gerenciador de tarefas assíncronas que utiliza trocas de mensagens entre o processo da aplicação que iniciará as tarefas e os trabalhadores, processos que executam as tarefas. Esta ferramenta facilita a escalabilidade de processos trabalhadores em máquinas distintas.
- Redis[3]: *software* que coordena o envio e o recebimento de mensagens entre os processos que atuam como trabalhadores e o processo responsável por manter a aplicação disponível para os usuários.

5.2 Cluster Laquibrido

Visando investigar os dispositivos disponíveis em cada partição do Laquibrido, utilizou-se o lstopo[4] da ferramenta hwloc[5], que apresenta informações sobre a topologia da arquitetura e indícios da existência de aceleradores em cada partição, mas sem maiores detalhes quanto a versões e tipos dos dispositivos.

Para facilitar a compreensão, foi criada uma representação visual utilizando o D3.js[6], uma biblioteca em JavaScript para manipular de forma gráfica, arquivos de dados, utilisando HTML, SVG e CSS. A Figura 5.1 apresenta a representação hierárquica do Laquibrido, como compreendida. A Figura 5.2 apresenta uma imagem obtida com a ferramenta lstopo, representando os dispositivos encontrados nas partições "host" e "fattwin-24". O código com a animação está disponível em: `https://github.com/anaveroneze/LAChibrido/tree/master/d3`.

5.3 Implementação

Inicialmente, foram necessários ajustes nas versões do FoF, reescrevendo trechos do código, para padronizar e facilitar a execução deles por linha de comando, e gerarem um arquivo de saída contendo a listagem das classificações das partículas, e um arquivo de log, contendo informações sobre a execução. Além disso, foram realizados ajustes na tela de administradores, para facilitar acesso de outros desenvolvedores na plataforma, e foram atualizadas as informações textuais e visuais sobre o portal. Utilizou-se *templates* do INPE para o *front-end* da plataforma.

[2]http://www.celeryproject.org/
[3]https://redis.io/
[4]https://linux.die.net/man/1/lstopo
[5]https://www.open-mpi.org/projects/hwloc/
[6]https://d3js.org/

Figura 5.1 - Animação sobre o Laquibrido criada com o D3.js

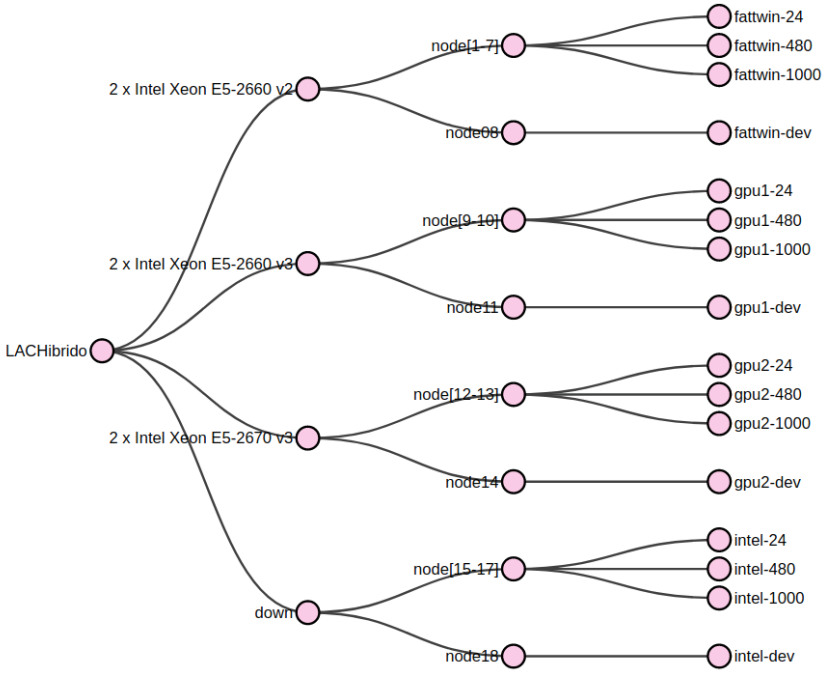


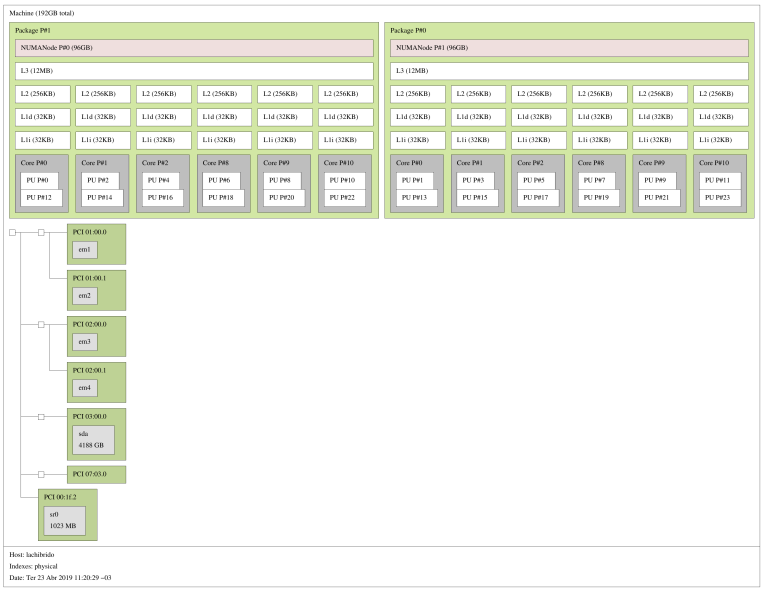
Imagem da animação criada utilizando o D3.js para representar a hierarquia dos dispositivos presentes no cluster. Os nodos são interativos, ao clicar nas informações sobre os processadores ele acessa a sua documentação oficial. Ao clicar nos nodos, ele abre imagens obtidas com o lstopo.

Fonte: Produção do autor.

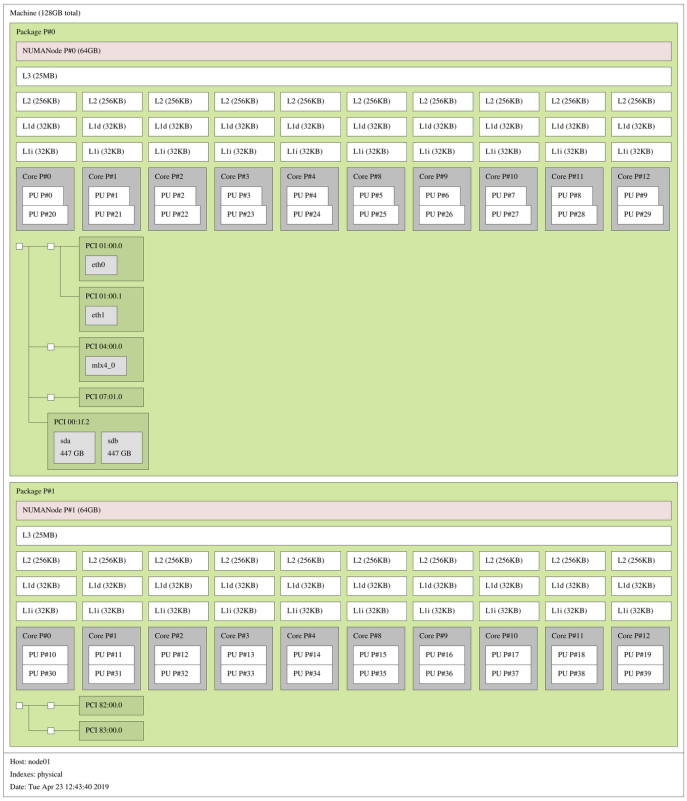
O projeto foi dividido em classes, visando facilitar a implementação colaborativa do portal, visto que o aluno Vinícius ficou responsável pela implementação do ambiente para execução do algoritmo de processamento de imagens. As classes dos algoritmos disponíveis foram implementadas separadamente, pois cada uma utiliza diferentes atributos, incluindo sua descrição, comando de execução e parâmetros. Uma classe em específico, mantém todas as informações sobre uma execução: o ID do usuário, a data da requisição da execução, o status de execução, o algoritmo utilizado, os arquivos de entrada, de saída e de log, e o tempo de execução. Outra classe, é responsável por mantém as informações do usuário, necessárias para autenticação, utilizando o sistema do Django para gerenciamento de usuários, e outras preferências do usuário relacionadas à interface do portal.

Com isso, existem três tipos de usuários distintos com acesso ao portal:

Figura 5.2 - Topologia do Laquibrido utilizando lstopo



(a) Nodos 1-7



(b) Host Laquibrido

- Anônimo: pode visualizar informações sobre a plataforma e os algoritmos cadastrados, cadastrar-se como novo usuário e entrar em contato com o

administrador do sistema;

- Registrado: pode executar experimentos com todos os algoritmos disponíveis, visualizar as execuções dos experimentos monitorando o status de seu trabalho e baixar seus respectivos dados de entrada, saída e registro ou cancelar a execução;

- Administrador: pode registrar novos algoritmos, editar a conta do usuário e qualquer outra operação relacionada ao banco de dados.

Para hospedagem do portal no Laquibrido, descobriu-se forte dependência do setor de gerenciamento de TI para autorizar a instalação de ferramentas e permitir o acesso aos serviços de rede necessários para a implantação da plataforma. Assim, uma opção de hospedagem do portal foi utilizar o sistema de computação em nuvem Heroku[7]. O Heroku é uma Plataforma como Serviço com suporte oficial a diversas linguagens de programação, a vários frameworks (incluindo Django) e com mais de 170 extensões disponíveis para auxílio na execução da aplicação.

Uma aplicação hospedada no Heroku é executada por *dynos*, que são contêiners Unix isolados e virtualizados, onde recebe o ambiente de execução necessário para sua utilização, além de lidar com requisições Web e processamento de trabalhos em segundo plano. Existem também os *web-dynos*, que possibilitam hospedar a aplicação sobre um domínio do Heroku, e os *one-off-dynos*, que são criados para executar comandos e tarefas esporádicas, acessados via sessão interativa com um interpretador de comandos.

A hospedagem de projetos no Heroku é via Git, e o fluxo de trabalho funciona como segue:(i) o usuário tem o código de sua aplicação e todos os programas necessários para a sua execução; (ii) o sistema do Heroku lê um arquivo intitulado *Procfile* que contém os comando que a plataforma deve utilizar para executar a aplicação; (iii) o usuário submete seu projeto para a nuvem (processo chamado de *deploy*) via git; (iv) o Heroku compila o código, interpretando o *Procfile* e, se tudo correr bem, disponibiliza a aplicação para o usuário.

Assim, limitou-se à implementação local do projeto e hospedagem no serviço em nuvem Heroku, para fins de visualização, visto que essa hospedagem não permite, por exemplo, a execução do FoF paralelizado para GPU. O portal no Heroku está disponível em: `https://observatoriovirtual.herokuapp.com/`

[7]https://heroku.com

5.4 Execução remota

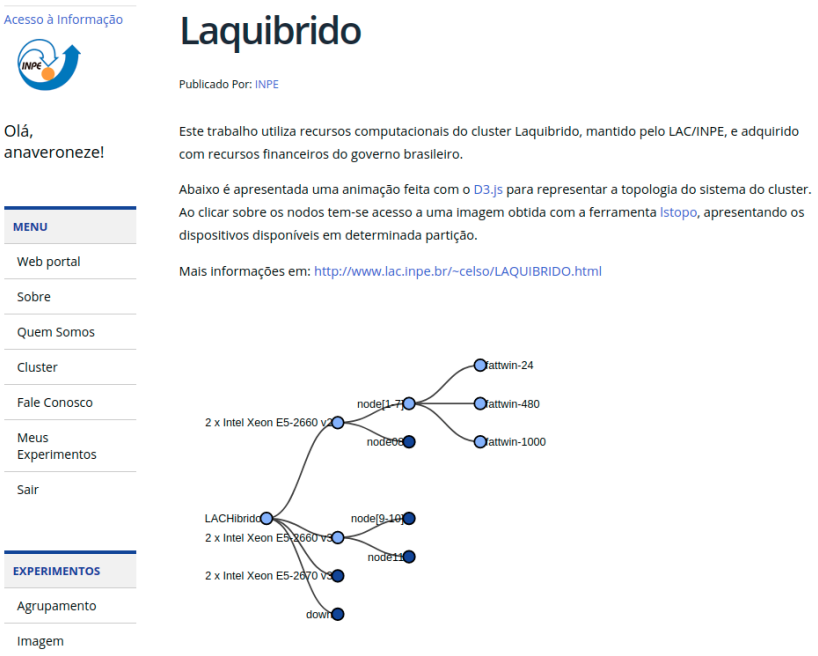
A página inicial da plataforma é apresentada na Figura 5.3. À esquerda, encontra-se um menu onde qualquer usuário tem acesso à página inicial, a uma página com mais informações sobre a plataforma, uma página com mais informações sobre os desenvolvedores, uma página com informações sobre o *cluster*, uma página de contato com os administradores do sistema, e à página de registro ou login.

Figura 5.3 - Página inicial do portal web

A Figura 5.4 apresenta a página com informações do Laquibrido. Nela, foi inserida a animação feita com o programa D3.js, apresentada anteriormente, desenvolvida durante estudos sobre o ambiente de execução.

Usuários registrados tem acesso a páginas contendo informações sobre os algoritmos disponíveis, no caso, o Friends-of-Friends e o algoritmo para remoção de ruído em imagens utilizando transformada Wavelet. A Figura 5.5 apresenta a descrição para o

Figura 5.4 - Informações sobre o Laquibrido



FoF, com detalhes sobre a lógica do algoritmo, referências e possibilidade do usuário realizar o *download* dos códigos-fonte dos algoritmos disponíveis no banco de dados do portal (no exemplo da imagem, apenas o algoritmo FoF versão serial havia sido inserido no banco de dados do portal).

Cada experimento escolhido no menu de “Experimentos” possui seu próprio formulário Django para configuração da execução. As Figuras 5.6 e 5.7 apresentam os formulários do Django para execução do FoF e do algoritmo com Wavelet, respectivamente. O usuário pode inserir seus dados e definir parâmetros de execução para ambos os algoritmos implementados. Assim que o usuário clicar no botão "Executar", ele será redirecionado para a página que apresenta a tabela de experimentos, agora atualizada com o novo experimento submetido.

A Figura 5.8 apresenta a tabela de experimentos, caso o usuário ainda não tenha realizado experimentos ela aparecerá vazia com a mensagem “Você ainda não realizou experimentos”. Esta tabela apresenta uma interface intuitiva para os usuários monitorarem e gerenciarem suas solicitações de execução (selecionar, visualizar, remover). Ele também permite que o usuário realize *download* dos arquivos contendo os dados de entrada submetidos pelo usuário, e contendo os dados de saída após a conclusão da execução.

Figura 5.5 - Informações sobre o FoF

Acesso à Informação

Olá,
anaveroneze!

MENU

- Web portal
- Sobre
- Quem Somos
- Cluster
- Fale Conosco
- Meus Experimentos
- Sair

EXPERIMENTOS

- Agrupamento
- Imagem

Agrupamento

Publicado Por: INPE

Aqui estão descritos os algoritmos de agrupamento disponíveis nesta plataforma e os links para download dos códigos.

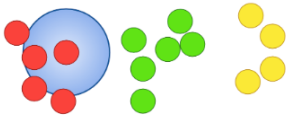
Friends-of-Friends

Friends-of-Friends (FoF) é um algoritmo usado para identificar estruturas no Universo baseado na distribuição de partículas que compõe a matéria escura do universo e a proximidade física entre elas [1]. Como entrada, o algoritmo recebe dados de simulação de N-corpos em um espaço tridimensional [2], usados para identificar halos e estruturas em simulações numéricas [3].

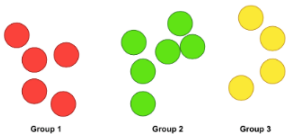
FoF is one of the most popular grouping percolation methods, specially because the other methods, as the Bayesian, for example, are not publicly available and are more difficult to code. Besides, the FoF uses just a single free parameter defined by the user, that determines the threshold of the linking length called percolation radius

The basic idea of the algorithm is: consider a sphere of radius R around each particle of a given set. If within that sphere exist other particles, they will be considered as belonging to the same halo and be called "friends". Then, we take a sphere around each friend and the procedure continues using the rule "any friend of my friend is my friend". The procedure stops when no new friend can be added to the group. The images below represents the FoF execution:

Posição inicial das partículas.



Considernaod uma esfera de raio R ao redor de cada partícula.



Grupos classificados pela sua posição gravitacional.

Algoritmos disponíveis:

- **FoF O(n²) Sequencial**

  Descrição: Algoritmo Friends-of-Friends

  Download

Referências

1. HUCHRA, J. P., GELLER, M. J. Groups of Galaxies I. Nearby groups. 1982. In: The astrophysical, v. 257, p. 423 - 437.
2. DAVIS, M., Efstathiou, G., Frenk, C.S., White, S.D.M., 1985. The evolution of large scale structure in a universe dominated by cold dark matter. In: Astrophys. J. 292, 371–394. doi:10.1086/163168.
3. KNEBE, A. et al.; Haloes gone MAD: The Halo-Finder Comparison Project. 2011. In: Monthly Notices of the Royal Astronomical Society. vol. 415(3), p. 2293-2318
4. DUARTE, M., Mamon, G. A. How well does the Friends-of-Friends algorithm recover group properties from galaxy catalogs limited in both distance and luminosity? 2014. In: Monthly Notices of the Royal Astronomical Society. 440(2). doi: 10.1093/mnras/stu378.

Figura 5.6 - Formulário de execução do FoF

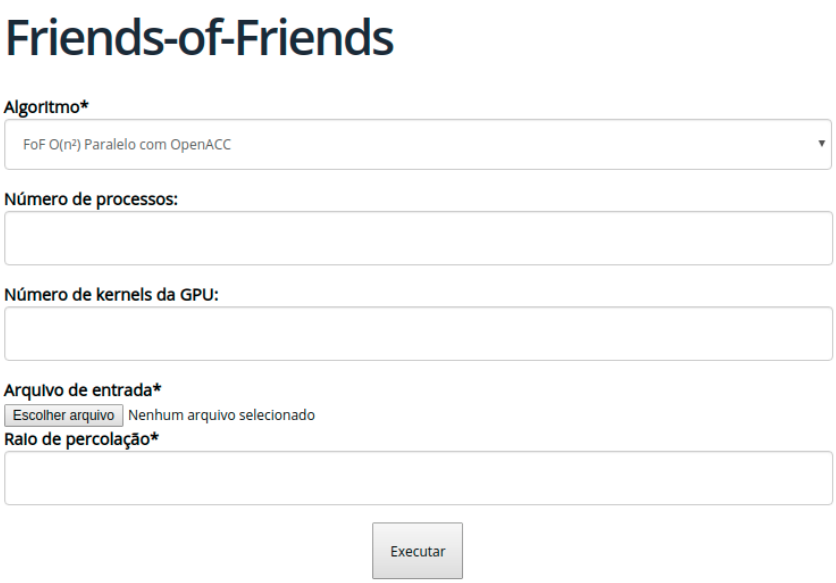


Figura 5.7 - Formulário de execução da Transformada Wavelet

O código-fonte do portal-web está disponível em um repositório no GitHub, onde é disponibilizada uma paǵina Wiki contendo informações sobre o projeto e instruções para instalação e execução local: https://github.com/anaveroneze/

Figura 5.8 - Tabela de Experimentos.

A tabela de experimentos apresenta as seguintes informações ao usuário: o Tempo de execução, que apresenta o tempo decorrido de uma execução e "concluído" para execuções concluídas, o Status, que informa ao usuário se uma solicitação já foi finalizada ou se ainda está aguardando a execução da tarefa.

observatoriovirtual/wiki.

6 REDAÇÃO DE ARTIGOS

A implementação da nova abordagem do FoF com OpenACC resultou no artigo "Exploração de Computação Híbrida com OpenACC em um Algoritmo Friends-of-Friends para Classificação de Objetos Astronômicos", apresentado na XIX Escola Regional de Desempenho da Região Sul (ERAD/RS) em abril de 2019, disponível nos anais do evento em: https://www.setrem.com.br/erad2019/download/Anais_ERAD_2019.pdf. O trabalho desenvolvido neste ano de projeto foi submetido ao SICINPE 2019, que será apresentado na Jornada Acadêmica Integrada(34ª JAI), na UFSM.

Foram aceitas duas publicações na Conference of Computational Interdisciplinar Science (CCIS 2019), apresentada pelo prof. Haroldo em março de 2019. Os artigos "Parallelization of the Friends-of-Friends Halo Finder Algorithm for a Hybrid Computing with OpenACC" e "Facilities for Remote Execution of High Performance Applications in Astronomy" estão disponíveis nos anais da conferência em: http://www.inpe.br/ccis2019/proceedings.php.

Além disso, a pesquisa do FoF com OpenACC foi aprovada para a sessão de pôsteres

da Grace Hopper Celebration[1], o maior evento mundial de mulheres nas tecnologias. Essa sessão, é voltada a estudantes de graduação ou pós-graduação que queiram apresentar suas pesquisas científicas, mesmo que ainda em andamento, esperando que a experiência proporcione *feedbacks* dos visitantes, sugerindo melhorias e novas abordagens ao trabalho. Dentre os 160 pôsteres aprovados, 24, incluindo esse, foram selecionados para participar do "ACM Student Research Competition"[2], uma competição patrocinada pela Microsoft que acontece nos principais eventos da ACM. Este trabalho será apresentado em setembro de 2019 em Orlando, Flórida.

7 CONCLUSÕES

Este projeto apresentou estudos sobre o algoritmo Friends-of-Friends, execução de testes com versões do algoritmo e implementação de novas versões utilizando OpenACC, para execução paralela em ambientes de computação híbrida composto por CPU e GPU. Estudou-se diferentes ambientes de computação, realizando testes sobre eles e *software* auxiliares para investigar os dispositivos disponíveis. Concluiu-se a implementação de um portal web para execução remota de algoritmos da área da Astronomia, com versões do FoF e de algoritmo para processamento de imagens utilizando a técnica *wavelet*, entretanto, mantendo como desafio a sua hospedagem no *cluster* Laquibrido.

REFERÊNCIAS BIBLIOGRÁFICAS

BARNES, J.; HUT, P. A hierarchical $o(nlogn)$ force-calculation algorithm. In: . [S.l.: s.n.], 1986. 5

BEHROOZI, P. S.; WECHSLER, R. H.; WU, H.-Y. The rockstar phase-space temporal halo finder and the velocity offsets of cluster cores. **The Astrophysical Journal**, v. 762, n. 2, p. 109, 2013. Disponível em: <http://stacks.iop.org/0004-637X/762/i=2/a=109>. 2

BERTSCHINGER, E. Simulations of structure formation in the universe. **Annual Review of Astronomy and Astrophysics**, v. 36, n. 1, p. 599–654, 1998. Disponível em: <https://doi.org/10.1146/annurev.astro.36.1.599>. 4

[1] https://ghc.anitab.org/
[2] https://src.acm.org/about

BERWIAN, L.; ZANCANARO, E. T.; CARDOSO, D. J.; CHARãO, A. S.; RUIZ, R. S. da R.; VELHO, H. F. de C. Comparação de estratégias de paralelização de um algoritmo friends-of-friends com openmp. In: **Anais da XVII Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul**. [S.l.: s.n.], 2017. p. 279 – 252. ISSN 2177-0085. 5, 9, 19

CARETTA, C. A.; ROSA, R. R.; VELHO, H. F. de C.; RAMOS, F. M.; MAKLER, M. Evidence of turbulence-like universality in the formation of galaxy-sized dark matter haloes. **Astronomy & Astrophysics**, v. 487, n. 2, p. 445–451, 2008. Disponível em: <http://dx.doi.org/10.1051/0004-6361:20079105>. 7

DAGUM, L.; MENON, R. Openmp: an industry standard api for shared-memory programming. **Computational Science & Engineering, IEEE**, IEEE, v. 5, n. 1, p. 46–55, 1998. 13

DUARTE, M.; MAMON, G. A. 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**, v. 440, n. 2, p. 1763–1778, 03 2014. ISSN 0035-8711. Disponível em: <https://doi.org/10.1093/mnras/stu378>. 4

Efstathiou, G.; Davis, M.; White, S. D. M.; Frenk, C. S. Numerical techniques for large cosmological N-body simulations. , v. 57, p. 241–260, fev. 1985. 4

Feng, Y.; Modi, C. A fast algorithm for identifying friends-of-friends halos. **Astronomy and Computing**, v. 20, p. 44–51, jul. 2017. 19

FREIRE, J.; BONNET, P.; SHASHA, D. Computational reproducibility: State-of-the-art, challenges, and database research opportunities. In: **Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data**. New York, NY, USA: ACM, 2012. (SIGMOD '12), p. 593–596. ISBN 978-1-4503-1247-9. Disponível em: <http://doi.acm.org/10.1145/2213836.2213908>. 2

Huchra, J. P.; Geller, M. J. Groups of galaxies. I - Nearby groups. **Astrophysical Journal**, v. 257, p. 423–437, jun. 1982. 2, 4

JENKINS, A.; FRENK, C. S.; PEARCE, F. R.; THOMAS, P. A.; COLBERG, J. M.; WHITE, S. D. M.; COUCHMAN, H. M. P.; PEACOCK, J. A.; EFSTATHIOU, G.; AND, A. H. N. Evolution of structure in cold dark matter

universes. **The Astrophysical Journal**, IOP Publishing, v. 499, n. 1, p. 20–40, may 1998. Disponível em: <https://doi.org/10.1086%2F305615>. 4

KAEHLER, R. Massively parallel computation of accurate densities for n-body dark matter simulations using the phase-space-element method. **Astronomy and Computing**, v. 20, 12 2016. 19

KNOLLMANN, S. R.; KNEBE, A. Ahf: Amiga's halo finder. **The Astrophysical Journal Supplement Series**, v. 182, n. 2, p. 608, 2009. Disponível em: <http://stacks.iop.org/0067-0049/182/i=2/a=608>. 2

LACEY, C.; COLE, S. Merger rates in hierarchical models of galaxy formation - part two - comparison with n-body simulations. **Monthly Notices of The Royal Astronomical Society - MON NOTIC ROY ASTRON SOC**, v. 271, 12 1994. 7

MADALOSSO, O. M.; CHARãO, A. S.; VELHO, H. F. de C.; RUIZ, R. S. da R. 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**. [S.l.: s.n.], 2015. p. 245 – 248. 5, 9, 18

_____. A web portal framework for remote execution of high performance applications in astronomy. In: **Proceedings of the 4th Conference of Computational Interdisciplinary Science**. [S.l.]: Pan-American Association of Computational Interdisciplinary Sciences, 2016a. 3, 20

MADSEN, M. S. In: **The Dynamic Cosmos - Exploring the Physical Evolution of the Universe**. New York, NY, USA: Chapman & Hall, 1996. ISBN 0412623005. 2

MORE, S.; KRAVTSOV, A. V.; DALAL, N.; GOTTLöBER, S. The overdensity and masses of the friends-of-friends halos and universality of halo mass function. **Astrophysical Journal Supplement Series - ASTROPHYS J SUPPL SER**, v. 195, 02 2011. 7

MORE, S.; KRAVTSOV, A. V.; DALAL, N.; GOTTLöBER, S. THE OVERDENSITY AND MASSES OF THE FRIENDS-OF-FRIENDS HALOS AND UNIVERSALITY OF HALO MASS FUNCTION. **The Astrophysical Journal Supplement Series**, IOP Publishing, v. 195, n. 1, p. 4, jun 2011. Disponível em: <https://doi.org/10.1088%2F0067-0049%2F195%2F1%2F4>. 4

NVIDIA. **PGI Community Edition**. 2012. Available: https://developer.nvidia.com/openacctoolkit. 5, 8, 13

OPENACC. **The OpenACC Application Program Interface**. 2015. Available: http://www.openacc.org/sites/default/files/OpenACC_2pt5.pdf. 8

RUIZ, R. S. d. R. **Turbulência em cosmologia: análise de dados simulados e observacionais usando computação de alto desempenho**. 145 p. Tese (Doutorado) — Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2011-05-26 2011. Disponível em: <http://urlib.net/sid.inpe.br/mtc-m19/2011/04.27.19.23>. Acesso em: 25 nov. 2017. 7

RUIZ, R. S. R.; VELHO, H. F. C.; CARETTA, C. A. Paralelização do algoritmo friends-of-friends para identificar halos de matéria escura. **IX Workshop do Curso de Computação Aplicada**, 2009. 5, 19

SPRINGEL, V.; YOSHIDA, N.; WHITE, S. Gadget: A code for collisionless and gasdynamical cosmological simulations. **New Astronomy**, v. 6, p. 79–117, 03 2000. 5