UNIVERSIDADE DE PASSO FUNDO

INSTITUTO DE CIÊNCIAS EXATAS E GEOCIÊNCIAS

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

INTEGRAÇÃO DA TECNOLOGIA CUDA AO MODELO DE PREVISÃO DO TEMPO ETA

Henrique Gavioli Flores

Passo Fundo

2018

UNIVERSIDADE DE PASSO FUNDO
INSTITUTO DE CIÊNCIAS EXATAS E GEOCIÊNCIAS
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

INTEGRAÇÃO DA TECNOLOGIA CUDA AO MODELO DE PREVISÃO DO TEMPO ETA

Henrique Gavioli Flores

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Computação Aplicada na Universidade de Passo Fundo.

Orientador: Marcelo Trindade Rebonatto
Coorientador: Carlos Amaral Hölbig

Passo Fundo
2018

CIP – Catalogação na Publicação

F634i Flores, Henrique Gavioli
Integração da tecnologia CUDA ao modelo de previsão do tempo Eta / Henrique Gavioli Flores. – 2018.
116 f. : il. color. ; 30 cm.

Orientador: Prof. Dr. Marcelo Trindade Rebonatto.
Coorientador: Prof. Dr. Carlos Amaral Hölbig.
Dissertação (Mestrado em Computação Aplicada) – Universidade de Passo Fundo, 2018.

1. CUDA (Arquitetura de computador). 2. Meteorologia. 3. Programação paralela. I. Rebelatto, Marcelo Trindade, orientador. II. Hölbig, Carlos Amaral, coorientador. III. Título.

CDU: 004.272

Catalogação: Bibliotecária Juliana Langaro Silveira – CRB 10/2427

UPF Universidade de Passo Fundo | PPGCA Programa de Pós-Graduação em Computação Aplicada
Instituto de Ciências Exatas e Geociências - ICEG

ATA DE DEFESA DO TRABALHO DE CONCLUSÃO DE CURSO DO ACADÊMICO

HENRIQUE GAVIOLI FLÔRES

Aos vinte e dois dias do mês de março do ano de dois mil e dezoito, às 14 horas, realizou-se, no Instituto de Ciências Exatas e Geociências, prédio B5, da Universidade de Passo Fundo, a sessão pública de defesa do Trabalho de Conclusão de Curso **"Integração da Tecnologia CUDA ao Modelo de Previsão de tempo e clima Eta"**, de autoria de Henrique Gavioli Flôres, acadêmico do Curso de Mestrado em Computação Aplicada do Programa de Pós-Graduação em Computação Aplicada – PPGCA/UPF. Segundo as informações prestadas pelo Conselho de Pós-Graduação e constantes nos arquivos da Secretaria do PPGCA, o aluno preencheu os requisitos necessários para submeter seu trabalho à avaliação. A banca examinadora foi composta pelos doutores Marcelo Trindade Rebonatto, Carlos Amaral Hölbig, Willingthon Pavan e Jorge Luís Gomes. Concluídos os trabalhos de apresentação e arguição, a banca examinadora considerou o candidato APROVADO. Foi concedido o prazo de até quarenta e cinco (45) dias, conforme Regimento do PPGCA, para o acadêmico apresentar ao Conselho de Pós-Graduação o trabalho em sua redação definitiva, a fim de que sejam feitos os encaminhamentos necessários à emissão do Diploma de Mestre em Computação Aplicada. Para constar, foi lavrada a presente ata, que vai assinada pelos membros da banca examinadora e pela Coordenação do PPGCA.

Prof. Dr. Marcelo Trindade Rebonatto - UPF
Presidente da Banca Examinadora
(Orientador)

Prof. Dr. Carlos Amaral Hölbig - UPF
(Coorientador)

Prof. Dr. Willingthon Pavan – UPF
(Avaliador Interno)

P.P.
Prof. Dr. Jorge Luís Gomes - INPE
(Avaliador Externo)

Prof. Dr. Rafael Rieder
Coordenador do PPGCA

AGRADECIMENTOS

Gostaria de agradecer meu orientador Marcelo Trindade Rebonatto, meu coorientador Carlos Amaral Hölbig, Alex Mello, Henrique Becker, Isabella Czamanski e Michell Siqueira por me auxiliarem no desenvolvimento deste trabalho.

INTEGRAÇÃO DA TECNOLOGIA CUDA AO MODELO DE PREVISÃO DO TEMPO ETA

RESUMO

Objetivo: Esta dissertação tem por objetivo principal realizar a implementação da tecnologia CUDA, para processamento de dados, no modelo de previsão do tempo Eta, buscando reduzir seu tempo de execução.

Materiais e Métodos: É apresentado um estudo sobre os modelos de previsão e a linguagem Fortran, a qual foi utilizada para a escrita do Eta. A forma de realização do processamento do modelo Eta foi estudada e mapeada, indicando que o processamento dos dados se dá pela divisão de tarefas MPI em dois grupos: servidores de I/O, responsáveis pelo armazenamento dos dados gerados, e processos de previsão, que computam cada parte da área a ser prevista. Foi realizada uma análise do código, buscando identificar atividades computacionalmente intensivas, com o intuito de identificar pontos se poderia aplicar mais de um nível de paralelismo. Após definido um modelo de paralelismo de dois níveis, de paralelismo, foi escolhida a tecnologia CUDA para tentar obter um ganho de desempenho sem alterar a lógica utilizada pelo modelo.

Resultados: O modelo foi implementado usando CUDA, sendo escolhidos quatro pontos de maior complexidade de código e tempo de execução para implementar o segundo nível de paralelismo. Foram realizadas 20 repetições para diversas combinações de números de processos, horas a serem previstas, tamanho de área da previsão e se foi utilizado ou não CUDA. Os resultados obtidos foram analisados usando os testes de Kolmogorov-Smirnov, Shapiro-Wilk, Teste de Levene e Teste-T, para verificar a distribuição dos dados das repetições e a significância dos dados resultantes.

Discussão dos resultados: Na análise dos resultados observou-se um ganho significativo fazendo uso de CUDA integrado ao modelo, apesar de em alguns casos ocorrer uma piora do desempenho, porém, os testes realizados ocorrem em ambientes computacionais não próprios para experimentos de aplicações de alto desempenho, podendo assim prejudicar os resultados gerados.

Conclusões: Pode-se afirmar que esta dissertação contribuiu para uma melhora do modelo Eta. O conhecimento obtido servirá como base de novos trabalhos que envolvam o modelo. Ressalta-se que devido à indisponibilidade de um ambiente computacional otimizado para aplicações de alto desempenho tornam-se necessários novos testes e ampliar os benefícios que podem ser obtidos fazendo uso da tecnologia CUDA em conjunto com MPI no modelo Eta de previsão do tempo.

Palavras-chave: Eta, meteorologia, CUDA, modelo paralelo.

INTEGRATION OF CUDA TECHNOLOGY TO THE ETA FORECAST MODEL

ABSTRACT

Objective: The main objective of this dissertation is to implement the CUDA technology for data processing in the Eta weather model in order to reduce its execution time.

Methods and Materials: A study on the prediction models and the Fortran language was presented, which was used for the writing of the Eta. The Eta model processing was studied and mapped, indicating that the data processing is done by dividing the MPI tasks into two groups: I / O servers, responsible for storing the generated data, and forecasting processes, which compute each part of the area to be predicted. A code analysis was performed to identify computationally intensive activities, in order to identify points if more than one level of parallelism could be applied. After defining a two-level parallelism model, CUDA technology was chosen to try to obtain a performance gain without changing the logic used by the model.

Results: The model was implemented using CUDA, being chosen four points of greater complexity of code and execution time to implement the second level of parallelism. Twenty replicates were performed for various combinations of process numbers, hours to be predicted, size of the forecast area, and whether or not CUDA was used. The results were analyzed using the Kolmogorov-Smirnov, Shapiro-Wilk, Levene Test and T-Test, to verify the distribution of the replicate data and the significance of the resulting data.

Discussion of the results: In the analysis of the results a significant gain was observed, making use of CUDA integrated to the model, although in some cases a performance worsening occurs, however, the tests performed occur in computational environments not suitable for experiments of high applications performance, and may therefore harm the results generated.

Conclusions: It can be stated that this dissertation contributed to the improvement of the Eta model. The knowledge obtained will serve as a basis for new work involving the model. It is emphasized that due to the unavailability of an optimized computing environment for high-performance applications, new tests and the benefits that can be obtained using CUDA technology in conjunction with MPI in the Eta model of weather are required.

Keywords: Eta, meteorology, CUDA, parallel model.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

CPTEC - Centro de Previsão de Tempo e Estudos Climáticos

CUDA - Compute Unified Device Architecture

ERAD – Escola Regional de Alto Desempenho

GFS - Global Forecast System

GPU – *Graphic processing unit*

INPE - Instituto Nacional de Pesquisa Espaciais

MPI - *Message Passing Interface*

NCEP - National Center Environment Prediction

OpenCL - Open Computing Language

OpenMP - *Open Multiple-Processing*

PGI – The Portland Group, Inc

PND - Modelos de Previsão Numérica de Tempo

RAM – *Random Access Memory*

SPSS - Statistical Package for the Social Sciences

ULA – Unidade Lógicas Aritmética

UPF – Universidade de Passo Fundo

SUMÁRIO

1. INTRODUÇÃO

O estudo da meteorologia auxilia direta e indiretamente a vida de todas as pessoas, seja apenas para saber se é necessário sair carregando um guarda-chuva, saber se é preciso irrigar o solo de uma plantação ou mesmo se o local onde um indivíduo se encontra será atingido por fortes chuvas, podendo ocorrer grandes catástrofes naturais.

A agricultura é um dos campos de pesquisa que mais faz uso de dados meteorológicos, devido ao fato de que elementos como radiação solar, temperatura e umidade relativas do ar influenciam diretamente na produção agrícola. Temperaturas extremas, baixa umidade e muita radiação solar são fatores que podem causar grandes impactos na época de colheita, prejudicando não apenas o produtor rural, mas sim todos que se beneficiam dos produtos dali gerados [1]. Uma ferramenta que auxilia agricultores e pesquisadores com relação a meteorologia e clima são os modelos de previsão.

Os modelos de previsão numérica de tempo (PND) são ferramentas que trabalham com uma grande massa de dados e que consomem um alto poder computacional, consumindo muitos recursos para gerar um resultado, dependendo do tamanho da área a ser calculada e do período proposto para a previsão. É importante ressaltar que, em uma previsão realizada por esses modelos, caso o período de tempo a ser previsto seja relativamente grande, as previsões tendem a se tornarem imprecisas.

Na literatura é possível encontrar, principalmente, duas classificações para os modelos de previsão, sendo esses, modelo de previsão global e modelo de previsão regional. Modelos globais trabalham com condições meteorológicas do planeta inteiro [2]. Os modelos regionais diferem dos globais, pois trabalham com dados de uma região específica, normalmente definida por um usuário. Uma grande vantagem dos modelos regionais em relação aos globais é que devido ao fato deles trabalharam com uma área menor, sendo que o resultado obtido tende a ter um maior detalhamento de informações sobre a área desejado [3].

O modelo Eta, para previsão de eventos atmosféricos voltado para pesquisas meteorológicas, não é uma exceção aos modelos regionais de previsão. São necessários muitos dados de entrada e um tempo considerável de processamento para que seja possível realizar a previsão de uma área. As previsões, geradas pelo Centro de Previsão de Tempo e Estudos Climáticos (CPTEC) ocorrem duas vezes ao dia, uma com condição inicial às 00:00 UTC e outra às 12:00 UTC, cobrindo a área de praticamente toda a América do Sul [4].

O modelo Eta, atualmente, é utilizado por diversos países como Algéria, Argentina, Bélgica, Brasil, Camarões, China, Costa Rica, República Tcheca, Dinamarca, Egito, Finlândia, Alemanha, Grécia, Islândia, Índia, Israel, Itália, Malta, Tunísia, Turquia, Peru, Filipinas, Servia e Montenegro, África do Sul, Espanha, Suíça e Estados Unidos [5].

O uso de técnicas de processamento paralelo de software estão cada vez mais presentes na literatura, nas mais diversas áreas de aplicação. Essas técnicas têm por objetivo dividir um trabalho que normalmente consome muitos recursos, em diversos menores para serem processados simultaneamente por vários computadores, para assim, obter uma redução no tempo total de processamento da aplicação. O uso de processamento paralelo de informações ocorrem, principalmente, em clusters compostos por diversos computadores, ou máquinas com alto poder de processamento.

Graphic processing unit (GPU) são dispositivos compostos de circuitos eletrônicos projetados para manipular e alterar rapidamente a memória e acelerar a criação de imagens destinadas ao dispositivo de saída [6]. Com o passar dos anos, essas unidades gráficas passaram a serem utilizadas para processamento de operações matemáticas, especialmente em cálculos matriciais, além de sua função principal. Visando facilitar o uso das GPUs para processamento de dados a NVIDIA criou a plataforma de computação paralela Compute Unified Device Architecture (CUDA), onde torna-se possível enviar dados para ser processado direto na unidade de processamento gráfico, enquanto a CPU pode estar disponível para outras tarefas [7].

Devido a estrutura da GPU, composta por diversas unidades lógicas aritméticas (ULA), esse dispositivo torna-se, por muitas vezes, uma ferramenta poderosa para processamento paralelo de informações [8].

Visando a importância do modelo de previsão Eta e a popularização, perante a literatura, do uso de GPUs para processamento de dados, o objetivo deste trabalho é realizar uma modificação da implementação integrando tecnologias de processamento paralelo em GPUs, usando a tecnologia CUDA, para reduzir o tempo de processamento total do modelo.

O presente trabalho foi estruturado em 8 capítulos. No capítulo dois, uma revisão bibliográfica sobre os modelos de previsão, o que são GPUs e a tecnologia CUDA, linguagem de programação Fortran e seu compilador PGI, a tecnologia para multiprocessamento MPI, MPICH e OpenMP são apresentados. Após essa revisão bibliográfica é abordado o funcionamento detalhado do modelo de previsão do tempo Eta, desde sua divisão de processos até o armazenamento dos resultados. O capítulo quatro apresenta os trabalhos relacionados presentes na literatura e o próximo os modelos de solução propostos. A implementação do

modelo Eta é apresentada no capítulo seis, sendo os resultados analisados no capítulo sete. Para finalizar, o último capítulo contém as considerações finais do trabalho.

2. REVISÃO BIBLIOGRAFICA

Este capítulo tem por objetivo apresentar, de modo geral, o que são modelos de previsão, comentar sobre GPUs e suas vantagens em relação a CPU, apresentar a tecnologia CUDA, o compilador PGI, a linguagem de programação Fortran e, por fim, as tecnologias para uso de processamento paralelo MPI MPICH e OpenMP.

2.1. MODELOS DE PREVISÃO NUMÉRICA E DE TEMPO

Modelos de previsão numérica e de tempo (PND) são software que fazem uso de modelos matemáticos com dados da atmosfera e dos oceanos no período atual visando prever a meteorologia e o clima para os próximos períodos determinados de tempo [9]. Estes modelos fazem uso de sistemas de equações diferenciais baseadas nas leis da física, movimentos dos fluídos e química, usando sistemas de coordenadas geográficas que dividem o planeta em uma grade de três dimensões [10].

Estes PNDs podem ser classificados em modelos de ponto de grade, que utilizam diferenças finitas, e modelos espectrais, os quais fazem uso de funções harmônicas e espectrais. Ambos os tipos representam a atmosfera baseados na resolução horizontal e em níveis verticais, onde a resolução horizontal refere-se à distância entre dois pontos de grade ou à distância da menor onda, no caso de modelos espectrais, e os níveis verticais referem-se à distribuição dos níveis verticais [11]. A Figura 1 apresenta a ilustração de um modelo de ponto de grade, cujo qual é o tipo do modelo Eta, apresentado no decorrer do trabalho.

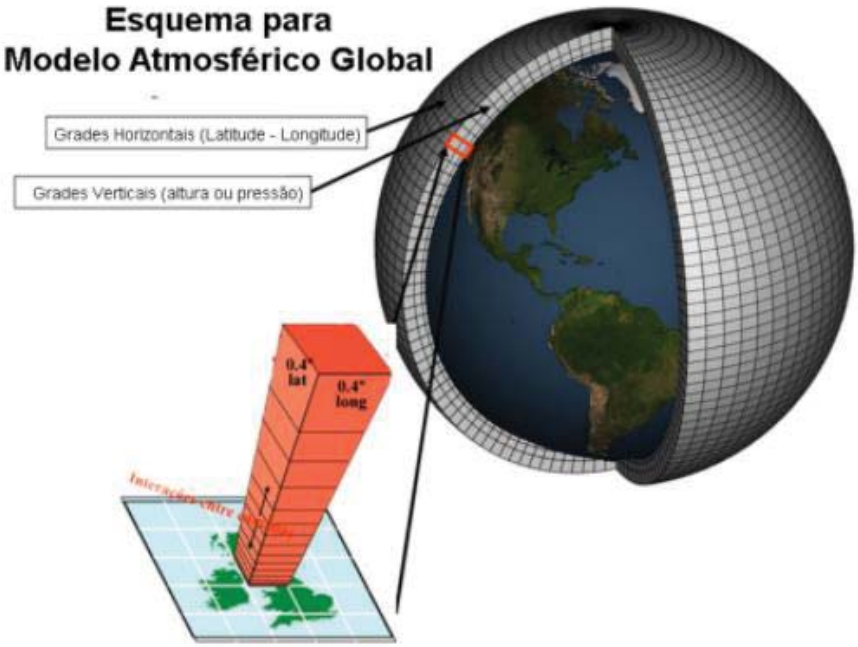

Figura 1. Representação visual de um PND global em ponto de grade, sendo representados por células de grades horizontais e níveis verticais [3]

A Figura 1 exemplifica o modelo em ponto de grade, que é representado por células (caixas) de grades horizontais e níveis verticais. Pressupondo-se, assim, que a atmosfera seja homogênea em cada uma dessas células, sendo necessário conhecer os dados climáticos de um ponto por célula [3].

Fatores como vento, transferência de calor, radiação solar, umidade relativa e hidrologia da superfície são calculados para cada célula da grade e essas informações interagem com as células vizinhas para calcular as propriedades atmosféricas para o futuro [10].

Na literatura é possível encontrar dois tipos de PNDs, que são os modelos regionais e globais. Modelos globais trabalham verificando as condições meteorológicas do planeta inteiro. Um exemplo de modelo global é o Global Forecast System (GFS), desenvolvido pela National Center Environment Prediction (NCEP), com o qual é possível obter uma previsão para todo o globo terrestre de até 16 dias no futuro [2]. Já os modelos regionais trabalham com dados de uma região específica, normalmente definida pelo usuário. Estes modelos tendem a ter um detalhamento de informações sobre a área desejada maior do que os modelos globais [3].

O modelo Eta é um PND de fenômenos atmosféricos regional usado para pesquisa e decisões operacionais. Ele trabalha com uma área de tamanho A para um período B, onde essas informações são definidas pelo usuário, sendo necessário alimentar o modelo com os

dados meteorológicos coletados da mesma área. Seu nome, “Eta”, é derivado da letra grega η (*Eta*), que significa coordenada vertical, sendo que o modelo é baseado na utilização de coordenadas verticais, que permanecem aproximadamente horizontais em áreas montanhosas, tornando-o adequado para estudos de regiões topográficas íngremes, como, por exemplo, a cordilheira dos Andes [12].

A estrutura do modelo Eta é formada por grades horizontais (Grade E de Arakawa) e coordenadas verticais. As resoluções das grades horizontais são de 40 km, 15 km, 5 km e 1 km [13], e essas estruturas possuem 38 camadas horizontais, onde a maior resolução se encontra nos níveis mais baixos, diminuindo em proporção que a altura aumenta [14].

No Brasil, o modelo Eta é utilizado no Centro de Previsão de Tempo e Estudos Climáticos (CPTEC) do Instituto Nacional de Pesquisa Espaciais (INPE). Este modelo é utilizado operacionalmente desde 1997 para previsões meteorológicas, e desde de 2002 para previsões climáticas, ambas abrangendo a América do Sul [12].

O modelo é alimentado com os dados referentes à área e à data a ser coberta pela previsão. Os dados de condições iniciais para o modelo são obtidos por meio de análise estatística, na qual a estimativa inicial é ajustada de acordo com as observações do horário da análise [14]. Ao final do processamento, é obtido o resultado da previsão para o período de tempo delimitado de, no mínimo, 6 horas, ou o número de horas desejada pelo usuário. Quanto maior o número de horas a serem previstas, menor será a precisão da resposta do modelo.

O modelo Eta é um descendente do modelo Hydrometeorological Institute and Belgrade University (HIBU), e seu desenvolvimento começou na década de 70 por Mesinger e Janjic [5]. Ele teve diversas versões desde sua criação e, atualmente, é largamente utilizado em vários países [5]. Sua codificação foi escrita usando a linguagem de programação Fortran 77 e reescrita para Fortran 90 com o passar dos anos.

2.2. GPU E CUDA

Graphic processing unit (GPU) é um circuito eletrônico projetado para manipular e alterar rapidamente a memória para acelerar a criação de imagens em um buffer de quadro destinado à saída para um dispositivo de exibição. GPUs são utilizadas nos mais diversos tipos de sistemas, celulares, videogames, computadores, etc. [6]

Além de processamento gráfico, GPUs podem ser utilizadas para processamento de operações matemáticas, especialmente em cálculos matriciais. Em aplicações com uma base

matemática robusta torna-se viável, em alguns casos, a implementação do uso integrado da GPU e da CPU para processamento.

CUDA é uma plataforma de computação paralela criada pela NVIDIA. CUDA permite que seja realizado processamento de dados utilizando a unidade de processamento gráfico (GPU) para isso. As linguagens de programação C, C++ e Fortran são habilitadas para o envio de códigos diretamente para GPU [7].

O processamento de operações de ponto flutuante nas GPUs, de acordo com a NVIDIA, é de aproximadamente 3000 operações por segundo em dados com precisão dupla, enquanto na CPU essas operações são em torno de um pouco mais de 500 em operações de precisão dupla [15], como mostrado na Figura 2.

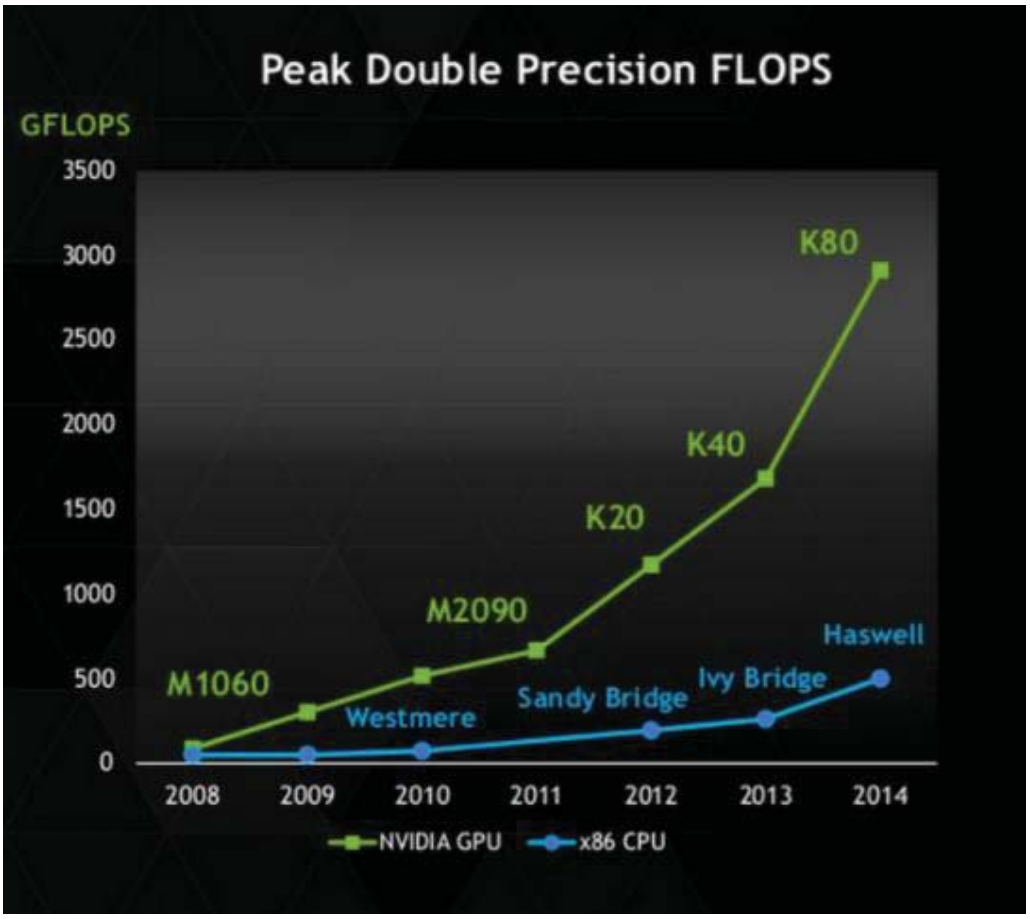

Figura 2. Número de operações de ponto-flutuante por segundo capaz de ser executado pela CPU em comparação a GPU [15].

Essa diferença na capacidade de cálculos de operações de ponto-flutuante se dá porque a GPU é uma unidade especializada em computação intensiva e computação altamente paralela, graças ao uso de muitas unidades lógicas aritméticas (ULA) a mais do que as utilizadas

pela CPU. A CPU destina 25% da sua área para unidades lógicas aritméticas enquanto a GPU tem aproximadamente 95% de sua área para essas unidades, vide Figura 3.

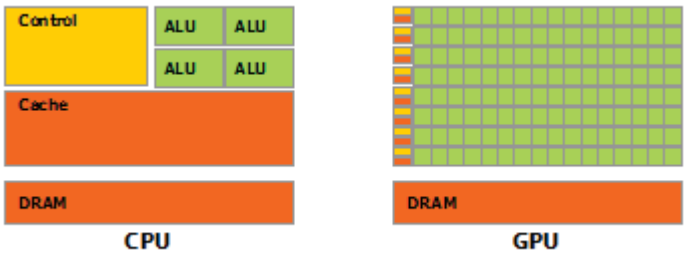

Figura 3. A GPU (à esquerda) dedica mais ULAs para processamento de dados em comparação com a CPU (à direita) [8].

Como se pode observar na Figura 3, a GPU tem uma área maior destina a cálculos numéricos em relação a uma CPU, porém, devido às poucas unidades de controle (em relação ao número de ULAs) a GPU se torna excelente na execução de uma mesma tarefa inúmeras vezes, desde que não haja dependências. É possível a execução de múltiplas tarefas distintas na GPU, mas isso é relativamente complexo de ser feito [16].

O uso de placas gráficas para processamento de grandes conjuntos de dados se tornou algo comum nos dias atuais devido à complexidade e o tamanho das operações, onde o tempo de consumo que era consideravelmente demorado, sendo que com o uso das GPUs esse tempo foi reduzido significativamente [7].

Para utilizar o CUDA com a linguagem de programação Fortran podem ser utilizados compiladores PGI. É válido ressaltar que, para a execução de códigos que utilizem CUDA, é necessário obter uma GPU compatível com a tecnologia, ou seja, a GPU deve possuir CUDA *Cores.*

2.3. FORTRAN E PGI

Fortran é uma linguagem de programação proposta por John W. Backus para a IBM no ano de 1957, com intuito de desenvolver uma alternativa mais prática para linguagem Assembly, utilizada para programar o computador IBM 704. Os desenvolvedores que trabalharam no projeto da linguagem foram Richard Goldberg, Sheldon F. Best, Harlan Herrick, Peter Sheridan, Roy Nutt, Robert Nelson, Irving Ziller, Lois Haibt e David Sayre. A linguagem foi adotada por cientistas para a escrita de programas com uma base matemática relativamente

complexa, influenciando os desenvolvedores de compiladores para o desenvolvimento de ferramentas de compilação que pudessem otimizar cada vez mais os códigos [17].

Apesar de ser uma linguagem de programação consideravelmente antiga, Fortran é largamente utilizada até hoje por pesquisadores que necessitam realizar grandes operações matemáticas e de grande complexidade [18].

The Portland Group, Inc, ou PGI, é uma companhia que desenvolve e vende compiladores para as linguagens de programação Fortran, C e C++. O foco dessa empresa é o desenvolvimento de compiladores de alta performance computacional. Levando em consideração o foco em alto desempenho, os compiladores suportam diversas diretivas de otimização como OpenMP, MPI, OpenACC, CUDA Fortran, OpenCL e High Performance Fortran (HPF) [19].

Atualmente, o modelo Eta é compilado utilizando uma das versões do compilador da PGI. Para a execução desse trabalho foi realizada uma parceria com o INPE, onde foi liberada uma versão paga do compilador PGI versão 10.5, porém ela não permite a compilação de diretivas CUDA. Portanto, para o trabalho realizado fazendo uso de GPUs foi necessária a versão de teste do compilador, o qual pode ser obtida gratuitamente por até um ano para fins de estudo.

2.4. MPI E MPICH

Message Passing Interface (MPI) é um padrão para comunicação de dados na computação paralela. É possível obter comunicação de dados entre processos, sendo apenas de um processo para outro ou comunicação coletiva entre os processos. O objetivo do MPI é a portabilidade, alto desempenho e escalabilidade das aplicações [20].

MPICH é uma implementação no MPI visando alto desempenho e portabilidade. O "CH" vem da palavra "*Chameleon*" (camaleão), pois este animal é um símbolo de adaptabilidade no ambiente onde vive. MPICH foi criado para elevar ainda mais a eficiência e a portabilidade de aplicações que fazem uso do MPI [21].

O funcionamento do MPI se dá por uma aplicação constituída de um ou mais processos que se comunicam por troca de mensagens. É possível que cada processo execute diferentes funções ao mesmo tempo, se existir mais de uma unidade processadora (*core*) disponível [20]. Normalmente, os desenvolvedores não utilizam mais processos que o número de núcleos disponível no processador, pois isto reduziria o ganho de desempenho, já que os processos ficariam realizando troca de contexto para processar [22].

MPICH se torna necessário para o modelo Eta pois muitas vezes a área em que se deseja realizar a previsão meteorológica ou o estudo do clima é relativamente grande, tornando o processamento demorado mesmo nos melhores computadores.

As rotinas MPI podem ser utilizadas para criar paralelismo nas linguagens Fortran, C e C++. Desta forma, a programação em Fortran e o uso das diretivas MPI se tornam relativamente simples de ser utilizadas, bastando apenas a inclusão da biblioteca e assim já sendo possível a utilização das funções específicas do MPI.

2.5. OPENMP

Open Multiple-Processing (OpenMP) é uma interface de programação com objetivo de prover multiprocessamento em uma aplicação. Esse recurso permite que uma aplicação executando em um único fluxo (*thread master*), se divida (*fork*) em vários fluxos (*threads*) escravos, estes realizando o processamento que cada fluxo foi designado e se una (*join*) novamente um único fluxo, vide Figura 4 [21].

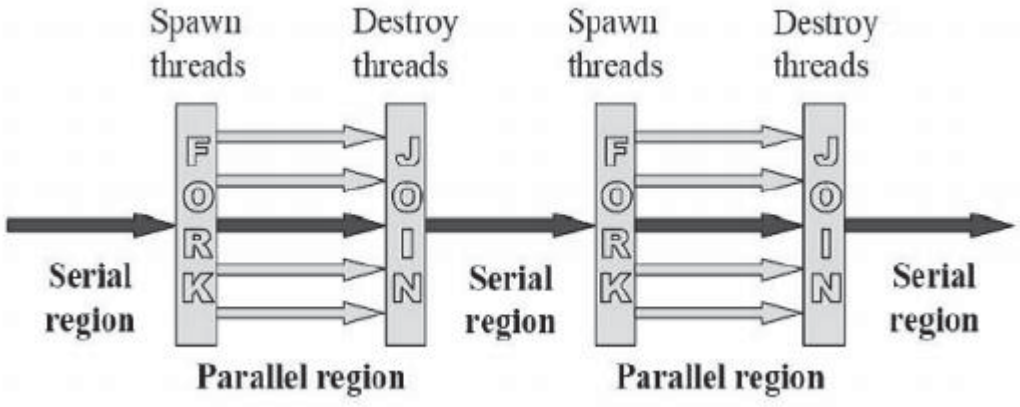

Figura 4. Modelo de execução *Fork*/*Join* implementado pela interface de programação OpenMP [21]

OpenMP é atualmente suportado pelas linguagens de programação C, C++ e Fortran, porém é de domínio público, podendo ser modificado por qualquer pessoa para adição de novas funcionalidades ou correções.

Assim como MPI, Fortran aceita as diretivas OpenMP sem grande complexidade no desenvolvimento de uma aplicação, bastando apenas a inclusão da biblioteca de OpenMP e então o código já está apto a fazer uso das diretivas da biblioteca.

3. PROCESSAMENTO DO ETA

Neste capítulo é apresentado detalhadamente o processo de como o PND Eta funciona, desde a inicialização dos processos, a divisão da área a ser processada entre os processos, o trabalho de previsão e, até mesmo, o armazenamento dos resultados gerados.

3.1. INTRODUÇÃO

Como já dito anteriormente, o Eta é um programa escrito na linguagem Fortran, sendo desenvolvido desde o ano de 1974 por diversos desenvolvedores. Atualmente seu código fonte encontra-se na linguagem Fortran versão 90 (Fortran 90). Uma vez que a previsão do tempo pode consumir muitos recursos e tempo de processamento, o Eta é paralelizado utilizando MPI. Esta seção tem por objetivo descrever o funcionamento do Eta sob o ponto de vista do paralelismo e do uso do MPI.

O processamento começa com a inicialização dos processos MPI e pela divisão das tarefas, classificando-as em "servidores de I/O" ou "tarefas de previsão". A região geográfica sobre a qual a previsão é realizada é definida e dividida entre os processos, com os mesmos conhecendo seus vizinhos e sabendo a área correspondente de cada processo. A previsão é realizada com os processos comunicando-se, dependendo do momento da execução do modelo de previsão. Ao final de cada ciclo de previsão, os resultados são armazenados e o ciclo se reinicia.

3.2. INICIALIZAÇÃO DOS PROCESSOS MPI

O Eta utiliza o MPI para permitir a sua execução em paralelo. O MPI deve estar operante e funcional na máquina onde o Eta será executado. Mesmo que o modelo possa ser executado apenas em um computador, o MPI é utilizado para gerenciar os processos que são criados durante a execução.

Para a execução do modelo, são criados dois tipos de processos: os que serão responsáveis pelas tarefas referentes ao cômputo do modelo de simulação, denominados aqui como "tarefas de previsão", e os que são responsáveis pelo armazenamento dos resultados produzidos, denominados aqui como "servidores de entrada/saída (I/O)". As quantidades dessas tarefas são definidas em variáveis de ambiente, configuradas no arquivo "set_parmeta_$$$",

onde $$$ é o nome do experimento. A multiplicação dos valores das variáveis INPES e JNPES resulta no número de tarefas de previsão, enquanto que o valor da variável NPIOSERVER representa a quantidade dos servidores de I/O.

Os servidores de I/O não são obrigatórios, de forma que eles podem não existir numa execução, bastando que se defina seu valor em zero (0). Nesse caso, os processos que executam tarefas de previsão irão executar as tarefas definidas como “CheckOut”, que representam, entre outras funções, o armazenamento em arquivos dos resultados da previsão.

O MPI é iniciado no procedimento denominado SETUP_SERVERS, que também prepara o ambiente para a comunicação entre os servidores de I/O e as tarefas de previsão. Ao iniciar uma aplicação com uso do MPI, deve-se passar a quantidade de processos que serão executados, que no caso do Eta é o resultado de: INPES * JNPES + NPIOSERVER.

As tarefas de previsão irão dividir a área a ser trabalhada e computar as tarefas de previsão da subárea correspondente. A cada fase calculada, os resultados são enviados ao CheckOut. As tarefas de previsão possuem como identificação MPI os *ranks* mais baixos, ou seja, começando em 0 e indo até INPES * JNPES – 1.

Os servidores de I/O são organizados em grupos e contexto de comunicação (comunicadores MPI) distintos, ou seja, quando há mais de um servidor de I/O, vários novos comunicadores são criados. Qualquer execução do Eta é limitada a no máximo 100 (cem) servidores de I/O. Caso seja passado um número maior, o valor será ajustado para o limite. Os grupos de comunicação criados incluem todos os processos de previsão, além do processo que será o servidor de I/O. A Figura 5 ilustra um exemplo de configuração.

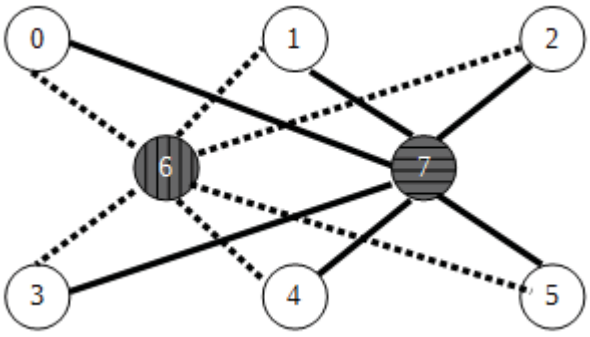

Figura 5. Representação dos servidores de I/O (6, 7), tarefas de processamento (0 a 5) e comunicadores entre os processos.

Os processos nas bordas da Figura 5 (0 a 5) representam tarefas de previsão, enquanto que os situados ao centro da Figura 5 e hachurados (6 e 7) representam os servidores de I/O. São criados dois novos comunicadores identificados pelas linhas sólidas e tracejadas,

que permitem a comunicação das tarefas de previsão com os servidores de I/O. Não é obrigatório que exista apenas um servidor de I/O em cada grupo, desta forma no grupo do servidor de I/O identificado com 7, poderia existir mais servidores de I/O vinculados. Convém salientar que, apesar da criação de novos comunicadores, o comunicador padrão do MPI (MPI_COMM_WORLD) é mantido presente e sem alterações.

3.3. DIVISÃO DA ÁREA DE PROCESSAMENTO

A imagem a ser processada inicia pelo ponto central, identificado no arquivo "set_parmeta_$$$", nas variáveis Lon e Lat, respectivamente identificando a longitude e latitude. A partir deste ponto central, é estabelecida a área onde a simulação da previsão do tempo é realizada, a partir das variáveis IM (latitude) e JM (longitude). A grade é expandida de acordo com o especificado na variável Res, onde Res é uma distância em Km definida na execução do modelo e a área de cada "elemento" é dada por Res ao quadrado. A Figura 6 ilustra a definição inicial da área a ser simulada.

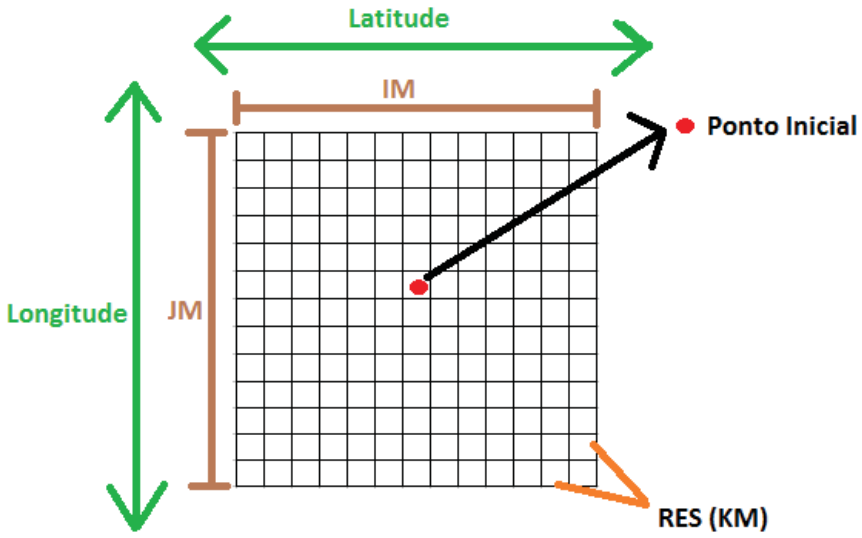

Figura 6. Representação da definição da área a ser processada pelo modelo, partindo de um Ponto Inicial estendendo-se, latitude e longitude, definidas pelo usuário.

Como é possível observar na Figura 6, a imagem é trabalhada no formato de matriz, onde as variáveis JM e IM são o número de elementos em cada eixo, IM sendo o número de pontos no eixo X e JM, o número de pontos no eixo Y. Após isso, a matriz é dividida em pedaços maiores, para que vários processos possam computar diferentes partes ao mesmo

tempo. Para que isso ocorra é necessário informar em quantos processos serão divididos o eixo X e o eixo Y. O número de vezes em que a matriz é dividida é definido nas variáveis INPES e JNPES, sendo o número de partes em que a matriz será dividida sobre o eixo X e o eixo Y, respectivamente. Como exemplo foi definido INPES = 2 e JNPES = 3, como é mostrado na Figura 7.

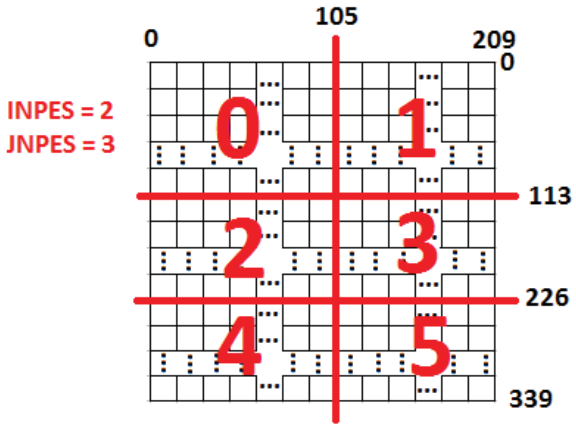

Figura 7. Divisão da área a ser processados entre os processos de previsão.

A matriz será dividida em 6 partes e, como visto na Figura 7, esse valor é obtido pela multiplicação das variáveis INPES e JNPES, que devem resultar no número total de processos que computarão a matriz. No caso, o eixo X será dividido em duas partes enquanto o eixo Y, em três partes. Cada processo de previsão irá computar apenas uma porção da área a ser simulada. Essa porção é definida internamente como CHUNK, e no exemplo o CHUNK "zero" é o intervalo entre os elementos 0-105 e 0-113, nos eixos x e y, respectivamente. Os números dentro de cada área da figura representam o número do processo responsável por computar este CHUNK.

Além do número de processos que irão "trabalhar" na matriz, deve ser definido o número de servidores de entrada e saída (I/O), sendo definido na variável NPIOSERVER. Nos exemplos a seguir definimos o valor de NPIOSERVER= 2.

3.3.1. Divisão da imagem em CHUNKS

Cada processo calculará quantos elementos serão computados por ele, a partir da área da imagem e do número de processos que a dividirão. O número total de elementos de cada eixo é dividido pelo número de processos que o eixo foi dividido, por exemplo, IM/INPES e

JM/JNPES. Uma vez que é recomendado que o total de elementos em cada eixo deve ser ímpar, podem ocorrer casos onde um processo irá computar pelo menos um elemento a mais que os demais.

Analisando a Figura 7 anteriormente, pode-se notar que os processos 0, 2 e 4 ficaram com um elemento a mais (105=105-0) que os processos 1, 3, 4 (104=209-105). A definição do número de elementos de cada processo influencia diretamente no limite de cada processo. Os limites são trabalhados em 4 variáveis, sendo início e fim sobre o eixo X, e início e fim sobre o eixo Y. Os valores de início e fim representam os intervalos de elementos que processo computará, ou seja, se um processo X tiver início = 1 e fim = 3, isso quer dizer que este processo computará os elementos 1, 2 e 3.

Os limites de cada processo podem ser observados na Figura 7 e detalhados na Tabela 1.

Tabela 1. Valores globais de localização, sendo o começo e o fim da área a ser processada por cada processo de previsão.

	Processos					
Variáveis	**0**	**1**	**2**	**3**	**4**	**5**
Início Global X	0	106	0	106	0	106
Fim Global X	105	209	105	209	105	209
Início Global Y	0	0	114	114	227	227
Fim Global Y	113	113	226	226	339	339

Na Tabela 1 são apresentados os valores de limite para cada processo com base nos valores citados como exemplo na seção 4.2. As variáveis globais armazenam os índices da imagem definido o início e o fim da área que aquele processo é responsável. Outra função que as variáveis globais exercem é a ligação de um processo com o outro, isso quer dizer que, a sequência direta dos elementos do processo 1 estão em seus processos vizinhos, por exemplo, é possível recriar toda a imagem original juntando os limites de cada processo.

Além das variáveis em nível global, existem as de nível local, que se limitam a armazenar o intervalo de elementos que cada processo armazenará com o objetivo de facilitar o acesso dessa informação em um nível local.

3.3.2. Valores locais e vetores de localização

Cada processo calcula suas variáveis em nível local, apresentadas no tópico anterior. Estas variáveis definem o número de elementos que cada processo terá para cada eixo. Vide Tabela 2.

Tabela 2. Valores locais de localização, sendo o começo e o fim da área a ser processada por cada processo de previsão.

	Processos	
Variáveis	**0, 2 e 4**	**1, 3 e 5**
Início Local X	0	0
Fim Local X	105	104
Início Local Y	0	0
Fim Local Y	113	113

Os processos terão os seus valores muito semelhantes, como pode ser visto na Tabela 2, devido ao fato que isto serve para um controle interno de cada um não dependendo das informações dos demais.

Cada processo define 4 vetores (L2GI, L2GJ, G2LI, G2LJ), que conterão a sua posição em um nível local e em um nível global. Os objetivos destas estruturas são informar aos demais processos quais e quantos elementos cada um computará e auxiliar nos cálculos da região cada elemento. Os vetores L2GI e L2GJ armazenarão os valores globais enquanto os G2LI e G2LJ conterão os valores locais de cada processo. Exemplo apresentado na Tabela 3.

Tabela 3. Área de atuação de cada processo.

	Processos					
Vetores	**0**	**1**	**2**	**3**	**4**	**5**
L2GI	[0-106]	[105-210]	[0-106]	[105-210]	[0-106]	[105-210]
L2GJ	[0-114]	[0-114]	[113-227]	[113-227]	[226-340]	[226-340]
G2LI	[0-106]	[0-105]	[0-106]	[0-105]	[0-106]	[0-105]
G2LJ	[0-114]	[0-114]	[0-114]	[0-114]	[0-114]	[0-114]

A Tabela 3 confirma que cada posição do vetor será preenchida pelo índice de cada elemento que será computado por cada processo, seja os índices globais ou local. É possível observar que o vetor L2GI do processo 0, por exemplo, tem 107 posições com valores validos, sendo esses valores os elementos que compõem aquela parte da imagem.

3.3.3. Definição dos vizinhos de cada processo

Cada processo descobre qual dos demais está computando a parte mais próxima dos elementos por eles sendo processado. Ao descobrir, os seus "vizinhos" são armazenados em um vetor de 8 posições onde cada índice representa um ponto cardeal, sendo 1 norte, 2 leste, 3 sul, 4 oeste, 5 nordeste, 6 sudeste, 7 sudoeste e 8 noroeste. Caso um ponto não tenha vizinhos o valor atribuído para aquela posição é -1 ao invés do número do processo. A Figura 8 ilustra a distribuição dos vizinhos de um processo e a relação com os pontos norte, sul, leste e oeste.

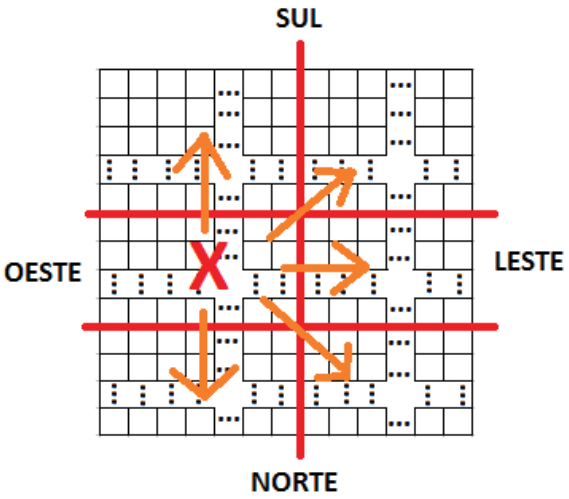

Figura 8. Representação do mapeamento realizado por cada processo para definir os processos vizinhos.

A Figura 8, apresenta um exemplo da definição dos vizinhos do processo 2, marcado com X. Levando em consideração os pontos cardeais calculados anteriormente, podemos definir que:

- Na posição ao sul ficou localizado o processo 0;
- Ao sudeste ficou localizado o processo 1;
- Ao leste, o processo 3;
- Ao nordeste, o processo 5;
- E ao norte, o processo 4;
- Nas posições ao noroeste, oeste e sudoeste o valor atribuído será -1;

Tabela 4. Vetor de processos vizinhos criados por cada um dos processos.

Posição	1	2	3	4	5	6	7	8
Processo	4	3	0	-1	5	1	-1	-1
Direção	Norte	Leste	Sul	Oeste	Nordeste	Sudeste	Sudoeste	Noroeste

A visualização dos dados no formato de um vetor pode ser observada na Tabela 4, onde cada posição recebe o processo correspondente ao exemplo mostrado na Figura 8.

3.3.4. Comunicação entre processos durante a divisão da imagem

Para que cada processo saiba quais elementos cada um dos demais está computando, é necessário o envio dos dados de definição global (início e fim global X e Y) calculados em cada processo. Estes dados serão enviados por cada processo para todos os demais, assim, ao final dessa operação cada processo conseguirá individualmente montar uma matriz que representa a imagem completa, com a identificação do processo que irá computar cada elemento, de acordo com a imagem original.

Depois disso, cada processo cria uma matriz de tamanho igual ao número de elementos que o processo computará. Em cada posição, desta matriz será armazenada a identificação do processo, como mostrado na Figura 9.

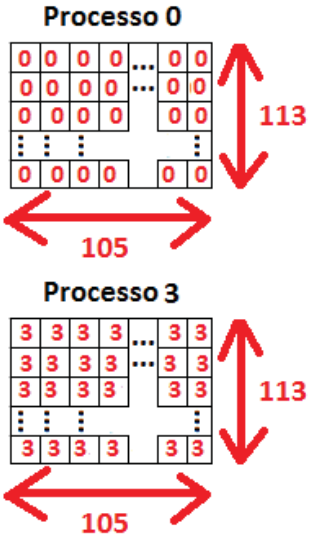

Figura 9. Valores armazenados nas matrizes de cada processo de previsão, sendo este o identificador do processo responsável por computar a área.

Na Figura 9, é ilustrado o exemplo levando em consideração apenas o processo 0 e o processo 3, estes criarão uma matriz com 113 linhas e 105 colunas, com todas as posições com valor 0 e 3, respectivamente, que é a identificação do processo. Esta matriz é enviada para os demais processos assim ao final desta operação cada processo "junta" todas as matrizes em uma única, do tamanho da imagem original onde cada elemento terá o valor do processo correspondente por computar aquela região, vide Figura 10.

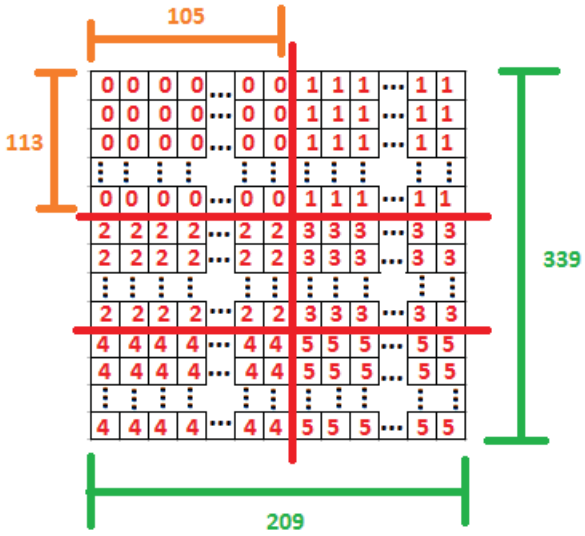

Figura 10. Matriz gerada ao final da comunicação entre os processos, representando qual parte da área inicial será computada por cada processo individualmente.

Na Figura 10, é ilustrada a matriz completa gerada por todos os processos ao final desta última operação comentada. É possível observar, que cada elemento tem registrado o identificador do processo responsável por aquela região.

Durante os cálculos da previsão, cada processo utiliza alguns elementos dos seus vizinhos para calcular as posições localizadas na borda da sua área. Esses elementos são comunicados entre os processos vizinhos por meio de uma função responsável por armazenar e enviar os elementos que serão compartilhados. Exemplo na Figura 11.

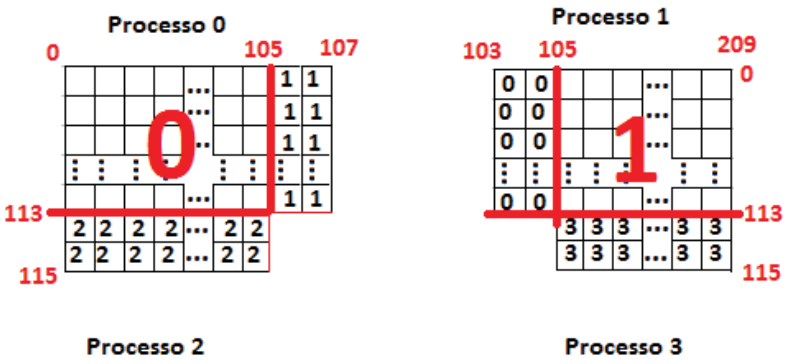

Figura 11. Elementos dos vizinhos para cálculos de borda.

Na Figura 11 é possível observar um exemplo dos elementos dos vizinhos usados para os cálculos nas partes mais extremas da área delimitada para aquele processo. O exemplo mostra que o processo 0 necessita de informações de alguns elementos do processo 1 e 2, enquanto o processo 1 precisa de informações do 0 e 3 e isto ocorre com os demais processos. No exemplo citado acima, foram utilizadas 2 linhas e 2 colunas para troca de informações entre os processos, mas dependendo dos cálculos que serão realizados nos processos de previsão é possível que mais ou menos linhas e colunas sejam compartilhadas, podendo variar de 1 a 5.

3.4. PROCESSOS DE PREVISÃO

Nesta seção será comentado sobre alguns cálculos das previsões climáticas realizadas pelo Eta, desde a leitura dos dados iniciais até os cálculos específicos do solo, nuvens, e demais influentes no processo de simulação climática.

3.4.1. Leitura dos dados iniciais para processamento

Neste ponto, todas as variáveis que contém dados de outras execuções são zeradas para evitar possíveis problemas de dados. Logo após, são lidas apenas pelo processo identificado por zero (0) as informações sobre número atômico efetivo, NHB, informações de controle do *timestep* e períodos de acumulação.

Após essa leitura inicial é realizado um teste para verificar se as informações iniciais necessárias para os cálculos do clima estão corretas. Se estes não estiverem corretos, as informações são novamente lidas e quando válidas são enviadas para todos os processos. Caso não precise reler os arquivos, as informações são apenas enviadas para cada processo para que

seja possível começar os cálculos. As informações referidas neste trecho são todas referentes a dados climáticos/químicos, como Pd, sH2O, albedo, etc.

O processo 0 lê de um arquivo as informações adicionais sobre as condições climáticas e envia para os demais processos, e então esses dados começam a ser computados por cada processo. Após iniciar as variáveis com as informações sobre física, é calculada a temperatura da superfície e do subsolo.

Cada processo inicializa as informações sobre o campo de nuvens, a umidade, etc, da sua região levando em consideração as variáveis globais que foram apresentas na seção 3.3.1, logo após isso são realizados cálculos dos níveis dos campos de nuvens para cada processo e apenas o processo 0 realizada o cálculo entre a distância não dimensional entre o sol e a terra, mandando o resultado para os demais processos via MPI_BCAST.

Todas as informações são lidas de arquivos apenas pelo processo 0 e então enviadas aos demais processos para que sejam computadas em paralelo. Essas leituras são realizadas apenas uma vez durante a execução do modelo.

3.4.2. Cálculos climáticos para cada região da imagem inicial

Primeiramente são executados os cálculos de *short* e *longwave*, junto com o ponto zenith da área a ser computada por cada processo.

Os cálculos, a seguir, são feitos para cada elemento de cada área a ser processada. São processadas as informações sobre a pressão da superfície, temperatura, latitude geodésica, ângulo zenith e albedo. São realizados cálculos das nuvens estratiformes, umidade específica e relativa e, após isso, são realizados testes para verificar se é possível realizar cálculos de radiação para as nuvens naquele ponto da imagem. Os pontos referentes por cada processo são os apresentados na divisão da imagem Figura 7.

São realizados vários cálculos, além dos citados com relação ao clima de cada área da imagem, porém não existe comunicação entre processos ou alguma operação em paralelo que será de suma importância para o trabalho.

Após essa primeira fase de cálculos, são atualizadas as informações de temperatura, umidade e pressão nas bordas do domínio utilizando as tendências pré-computadas em cada *timestep*. As informações necessárias são lidas de um arquivo pelo processo de previsão de *rank* zero e transmitida para os outros processos de previsão por comunicação global (MPI_BCAST). Em seguida, é calculada quando ocorrerá a próxima condição de borda.

Neste momento é calculada a contribuição da advecção horizontal levando em consideração a umidade e a quantidade de água nas nuvens. Após esse cálculo ser realizado, é realizado um MPI_ALLREDUCE para realizar o somatório e armazenar o resultado em todos os processos.

Além desses cálculos, são computados os processos microfísicos de condensação e precipitação. No final são utilizadas cinco operações de comunicação global (MPI_REDUCE) para calcular estatísticas que serão mostradas no arquivo de saída do modelo.

É válido ressaltar que todos os cálculos citados nesta seção são executados por todos os processos.

3.5. ARMAZENAMENTO DOS RESULTADOS

O processo responsável por armazenar os resultados da previsão realizada é o CHKOUT. Além disso, durante a sua execução, ele calcula estatísticas e tendências de variáveis climáticas, tais como temperatura e pressão, entre outras. O armazenamento dos resultados é realizado com os processos de previsão enviando dados aos servidores de I/O ou por eles próprios.

Ao ser inicializado, o processo responsável por armazenar os resultados realiza atribuições preliminares de valores para variáveis como, por exemplo, topografia e radiação. Em seguida é verificada em que etapa da previsão o programa se encontra. Caso a hora atual da previsão for múltiplo da variável IntFct, definida no set_parmeta_$$$, ocorre o processo de armazenamento propriamente dito, denominado *output*. Caso contrário, as variáveis são reinicializadas e o processamento do CHKOUT é finalizado.

Antes de executar o output, são realizadas chamadas de comunicação global (reduções globais – MPI_Reduce) para calcular estatísticas de temperatura, sendo atualizado num processo (identificado como 0) os valores. Em seguida, os processos são sincronizados com uma barreira de comunicação (MPI_Barrier). Após os processos passarem do ponto de sincronização, o *output* começa. Ele pode ser realizado com uso de servidores de I/O ou pelos processos de previsão.

3.5.1. *Output* com acesso direto

Neste caso, o *output* é realizado pelos processos de previsão, onde é gerado um arquivo nomeado por meio da expressão regular “\rstrt([0-9]{6})\”, sendo concatenada a

expressão “.quilt” ao final. Na expressão regular acima, “rstrt” é o prefixo do nome do arquivo e “[0-9]{6}” representa uma sequência de seis dígitos entre zero e nove.

Um exemplo de um arquivo de saída seria “rstrt000001.quilt”. Isso significa que é o primeiro arquivo a ser gravado por acesso direto. Por padrão é gerado um arquivo para cada hora de simulação realizada. Seguindo o exemplo anterior, para três horas de simulação serão gerados os arquivos: “rstrt000001.quilt”, “rstrt000002.quilt” e “rstrt000003.quilt”.

Todos os processos de previsão atuam sobre o mesmo arquivo, sendo calculado o ponto dentro do arquivo a partir do qual cada processo irá gravar, garantindo que os dados não sejam sobrescritos. Para tanto, a partir do total de informações a serem gravadas por cada processo é realizada uma redução global, buscando a maior quantidade de dados a serem armazenados. A partir desse valor, cada processo consegue calcular seu ponto de inserção. A Figura 12 mostra a interação entre os processos para criar o arquivo por acesso direto.

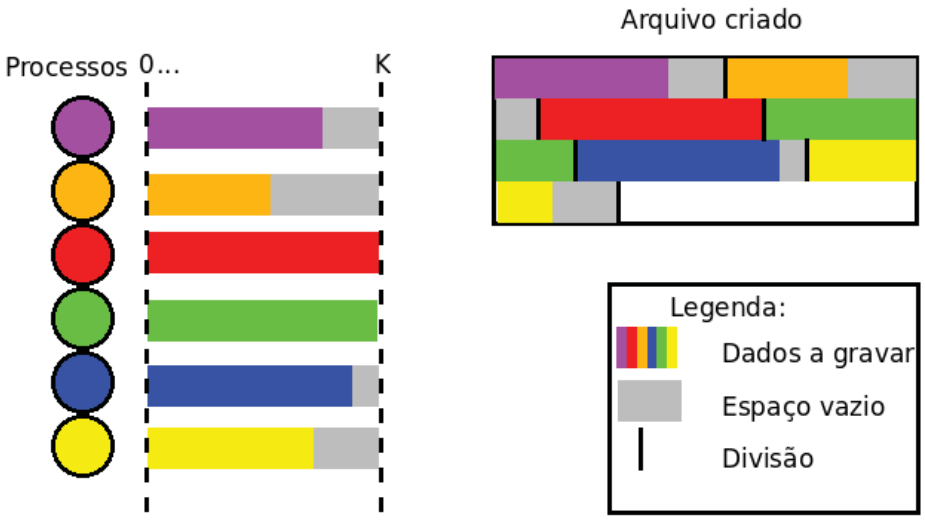

Figura 12. Exemplo da gravação de um arquivo pelo método de *output* por acesso direto, onde é criado um espaço em um arquivo de saída referente ao tamanho do maior processo, deixando espaços em branco nos demais para não ocorrer sobre escrita.

A Figura 12 exemplifica o *output* por acesso direto. O espaço que cada processo possui dentro do arquivo criado é igual ao tamanho ocupado pelo maior *buffer* entre todos os processos. Como cada processo possui um tamanho diferente, é normal espaço inutilizado no meio do arquivo.

3.5.2. *Output* com servidores de I/O

Assim como no acesso direto, o nome do arquivo gerado é dado pela mesma expressão regular descrita no *output* com acesso direto, porém ao final é concatenado “.t00s”

pelo servidor de I/O. Nomes válidos seriam “restrt000001.t00s”, “restrt000002.t00s” e “restrt000003.t00s”.

O processo de previsão com *rank* global zero envia o número do arquivo a ser gravado para o processo de *rank* local zero do grupo de servidores de I/O sendo usado. Um grupo de servidores de I/O pode possuir um ou mais servidores. Cada processo de previsão calcula antecipadamente com qual servidor de I/O do grupo ele irá se comunicar.

Com base no número de servidores de I/O no grupo, é utilizada uma função para calcular o intervalo de iteração de cada servidor. Cada tarefa de previsão utilizará o servidor de I/O cujo *rank* caiba no intervalo gerado (inicio < *rank* < fim). O processo usa uma chamada de comunicação não bloqueante (ISEND) para enviar o *buffer* ao servidor de I/O. Após o envio, é definido o próximo grupo de servidores de I/O para a execução seguinte. Os dados enviados pelos processos de previsão são recebidos nos servidores de I/O e tratados na função QUILT. Por padrão, cada grupo de servidores de I/O será composto por apenas um (1) servidor, porém, o cálculo para definir o servidor de I/O usado do grupo é realizado sempre.

Se o arquivo a ser gravado for o último (final da execução), além de mandar os dados para o servidor de I/O, cada processo gera um arquivo individualmente, contendo as mesmas informações armazenadas no *buffer*. Cada arquivo sendo nomeado pela mesma expressão regular já citada, e ao final sendo concatenado o identificador do processo de previsão. Como exemplo, ao final de uma simulação de três horas com um *output* a cada hora de simulação, o processo zero cria o arquivo chamado “restrt000003.0”, processo de *rank* um cria “restrt000003.1” e assim por diante.

3.5.3. Inicialização dos processos de I/O

O arquivo QUILT é o subprograma que os servidores de I/O executam. Durante a execução do SETUP_SERVERS são definidos quais processos serão utilizados para previsão, os restantes serão servidores de I/O. Ao retomar o fluxo principal do modelo, os processos de previsão continuam a execução normal e os destinados a servidor de I/O entram no QUILT. Os servidores de I/O recebem das tarefas de previsão os resultados da execução do modelo e os gravam em disco.

No início da execução do QUILT é lido o arquivo que contém informações sobre a execução do sistema, como controle de *timestep* e períodos de acumulação, e em seguida é verificado se é uma execução válida. Caso contrário, a execução é interrompida (MPI_ABORT).

Após esta etapa, o servidor de I/O de *rank* zero recebe (MPI_RECV) do CHKOUT o valor da hora de *output*. Esse valor complementa o nome do arquivo a ser criado, que é definido pelo prefixo “restrt” seguido de seis dígitos de zero a nove sendo adicionado ao final o sufixo “.t00s”. Por exemplo, o primeiro arquivo a ser gravado é “restrt000001.t00s”, enquanto o arquivo do tempo de simulação 3 é “restrt000003.t00s”. O valor da hora de *output* é transmitido para todos os demais servidores de I/O do grupo de servidores sendo usado (MPI_BCAST). Os servidores de I/O recebem (MPI_RECV) os resultados dos processos de previsão e armazenam essas informações em memória. Salientando novamente que por padrão um grupo de servidores terá apenas 1 servidor de I/O, neste caso, todos os processos se comunicam diretamente com o servidor de I/O de *rank* 0, pois este é o único no grupo. No entanto, visto que os cálculos realizados suportam a existência de mais de um servidor em cada grupo, será utilizado 2 servidores para o exemplo seguinte. A Figura 13 exemplifica a comunicação entre processos de previsão e servidores de I/O.

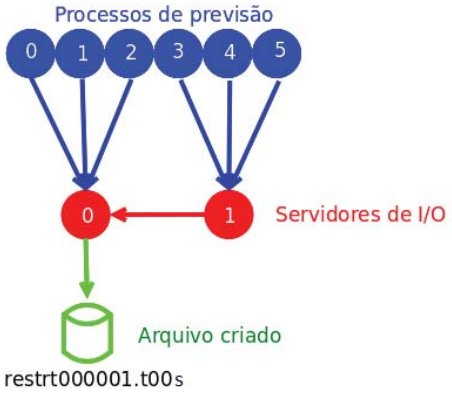

Figura 13. Fluxo de comunicação entre tarefas de previsão e servidores de I/O para a criação do arquivo de saída, sendo que, cada processo se comunica com um determinado servidor de I/O, estes se comunicam entre si (caso exista mais de um) e é realizada a geração do arquivo de saída.

Os valores dos resultados da previsão realizada são efetivamente armazenados pelos servidores de I/O com *rank* local zero de cada grupo, a partir dos dados recebidos dos demais servidores de I/O daquele grupo. Caso exista mais de um grupo de servidores de I/O, em cada tempo de *output*, apenas um grupo de servidores de I/O é acionado, ou seja, o tempo 1 (restrt000001.t00s), é armazenado por um grupo diferente do tempo 2 (restrt000002.t00s). Dessa forma, é garantido que apenas um processo manipule um arquivo. Convém salientar que um grupo de servidores de I/O pode ser formado por apenas um processo.

Ao final da simulação, os servidores com *rank* local zero de todos os grupos de servidores de I/O, recebem uma mensagem especial indicando final de execução. Essa mensagem é repassada aos demais componentes de seu grupo e todos encerram sua execução.

3.6. DIAGRAMAS

Para que o processo de execução do modelo Eta fosse melhor visualizado e compreendido foi montado os diagramas apresentados nas Figuras 14 e 15.

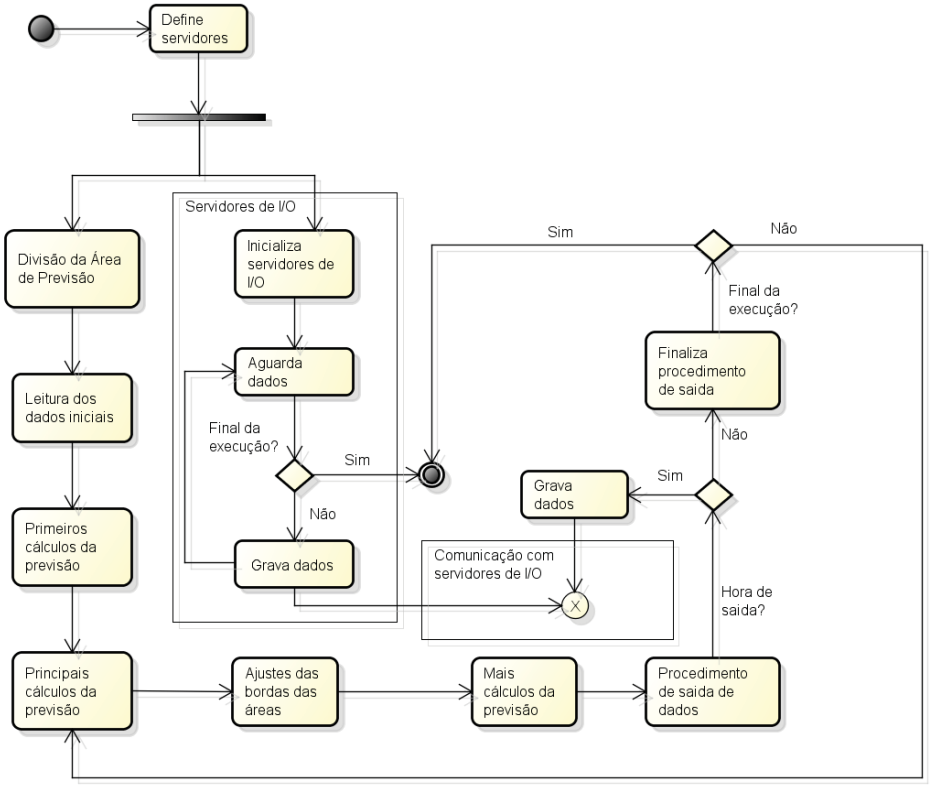

Figura 14. Diagrama de execução do modelo Eta.

O diagrama representado na Figura 14 apresenta o processo de divisão da área de previsão, desde a definição de sua área inicial até a comunicação da área a ser computada por cada processo.

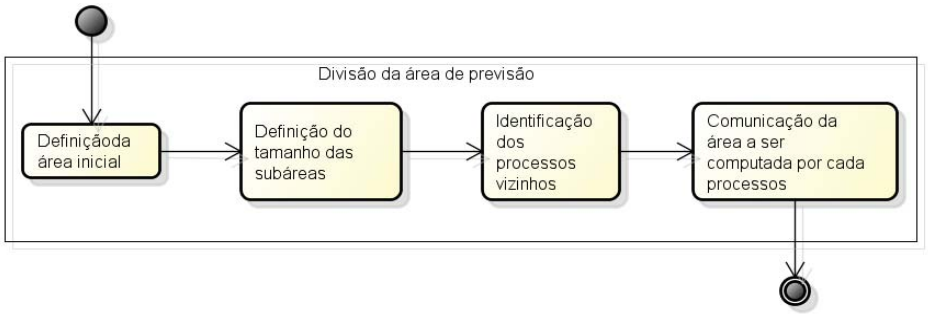

Figura 15. Diagrama parcial do Eta representando a divisão da área de previsão.

A Figura 15 complementa a Figura 14 demonstrando os outros passos do processamento executado pelo modelo Eta. Uma dificuldade encontrada com relação aos diagramas foi a representação da comunicação das informações a serem gravadas com os servidores de I/O e a representação do paralelismo existente na execução dos cálculos do modelo.

4. TRABALHOS RELACIONADOS

Antes de iniciar a implementação do presente trabalho, foi realizada uma busca na literatura com o objetivo de encontrar trabalhos relacionados ao tema. Os trabalhos encontrados foram divididos em três grupos: modelos de previsão matemática que utilizam técnicas de paralelismo, problemas gerais implementados com uso de paralelismo para fazer uso de multi processamento e sistemas diversos que fazem uso das tecnologias MPI e CUDA.

Existem diversos modelos de simulação e previsão semelhantes ao Eta utilizados para os mais variados fins. Um exemplo destes é o *Ocean-Land-Atmosphere Model* (OLAM), que consiste em discretizar, por meio da técnica de volumes finitos, as equações de Navier-Stokes, aplicadas sobre a atmosfera planetária, com a formulação de equações que respeitem as leis de conservação de massa, momento e temperatura potencial, e de operações numéricas que incluem passos discretos de tempo [23]. Além da semelhança de ser um modelo de previsões atmosféricas, outra similaridade com o Eta é o uso da tecnologia MPI em sua implementação e, além disso, vários níveis de processamento [24], onde foram utilizados CUDA e OpenMP que tem por objetivo uma melhora do desempenho dos cálculos realizados pelo modelo. Apesar de serem citadas três tecnologias que visam utilizar do paralelismo, nenhuma implementação do OLAM faz uso das três simultaneamente.

O trabalho apresentado por [25] usa uma abordagem semelhante ao OLAM, onde aplica-se dois níveis de paralelismo, criando uma solução híbrida. São utilizados MPI e CUDA com o objetivo de melhorar o método de Roe [25] na implementação de um sistema de águas rasas usando duas camadas de paralelismo. Este sistema trabalha fazendo uso de intenso processamento e grande quantidade de dados para os cálculos a serem realizados, sendo assim, um cenário parecido ao do modelo Eta. O resultado obtido pela implementação foi uma eficiência maior fazendo uso de um cluster de computadores onde cada computador possui uma GPU para processamento em relação a implementação tradicional fazendo uso apenas da CPU.

Na literatura é possível ainda encontrar diversos trabalhos que buscam explorar o paralelismo usando vários níveis. Em [26] são citadas diversas aplicações que integram OpenMP com MPI, OpenMP com CUDA e MPI com CUDA, onde as complexidades dos problemas são baixas, sendo por muitas vezes apenas estratégias de decomposição para algoritmos fazerem uso de GPU e usufruir do paralelismo resultante da combinação das tecnologias.

A aplicação de técnicas de paralelismo utilizando MPI combinadas com o processamento em GPU usando CUDA estão cada vez mais presentes na literatura. Alguns exemplos de áreas onde as combinações dessas tecnologias podem ser aplicadas:

- Multiplicação de matrizes esparsas para o pré-condicionamento do método dos gradientes conjugados. [27]
- Implementação e otimização usando paralelismo de equações esparsas e mínimo quadrado. [28]
- Uma implementação de um método eficiente para o cálculo de coeficiente de correlação de Pearson em matrizes. [29]
- Implementação de algoritmos de recuperação de senha. [30]
- Analise a performance de uma implementação do LU *benchmark.* [31]

O uso de GPU para vários níveis de processamento é explorado também em [26], onde é realizada a combinação de OpenMPI e StarSs [33], resultando na tecnologia OmpSs, que trata-se de um modelo que busca permitir que o desenvolvedor se concentre na computação própria sem ter que se preocupar com a distribuição de trabalho e dados para múltiplas GPUs em um cluster [32].

O trabalho de Huismann, I. et al. [34], consistiu na implementação de um algoritmo de mecânica de fluidos explorando a heterogeneidade de hardware trabalhando com as tecnologias OpenMP, MPI e OpenACC. Neste caso é apresentado uma implementação de três níveis de paralelismo mas diferente dos demais trabalhos. Não foram realizados testes em relação a seu desempenho, e sim, apenas na possibilidade da implementação realizada.

Os resultados obtidos pelos trabalhos, [27, 28, 29, 32, 24, 30, 26, 31, 25] são semelhantes. Após as implementações fazendo uso da GPU foram obtidas reduções nos tempos de processamento em relação aos software que não fazem uso da GPU.

Na Tabela 5 é apresentada uma sumarização dos trabalhos relacionados citados, sendo elas, ano de publicação, tecnologia utilizada para implementação, níveis de paralelismo e linguagem de programação.

Tabela 5. Tabela sumarizadora de trabalhos, tecnologias e conclusões em relação ao desempenho da aplicação

Trabalhos	**Ano**	**Tecnologias envolvidas**	**Níveis de paralelismo**	**Linguagem**
[31]	2011	MPI, CUDA	2	Fortran 77
[25]	2011	MPI, CUDA	2	C++
[29]	2011	MPI, CUDA	2	C

Trabalhos	Ano	Tecnologias envolvidas	Níveis de paralelismo	Linguagem
[28]	2012	MPI, CUDA	2	C
[32]	2012	OpenMPI, StartSs	2	C, Fortran
[24]	2012	OpenMP, CUDA, MPI	2	Fortran 90
[30]	2012	MPI, CUDA	2	C
[26]	2012	MPI, CUDA	2	C, Fortran
[27]	2014	MPI, CUDA	2	C
[34]	2015	OpenMP, MPI, OpenACC	3	Fortran

Na Tabela 5 é possível verificar que nos últimos quatro anos, as tecnologias como MPI e CUDA, vem sendo utilizadas para explorar o paralelismo em trabalhos das mais diversas áreas, sendo implementados dois ou mais níveis de paralelismo nas aplicações, quase que sempre integradas com as linguagens de programação Fortran e C.

5. MODELO DE SOLUÇÃO PROPOSTO

Neste capítulo são abordadas as análises realizadas no código do modelo Eta e os três modelos de implementações teoricamente possíveis de serem implementadas visando melhorar o desempenho do modelo.

5.1. ANÁLISE DO CÓDIGO DO ETA

Com o decorrer dos anos, diversos desenvolvedores já trabalharam no código do modelo, visando ajustes, correções ou até mesmo melhorias. Para que fosse possível montar um modelo de pesquisa que busca melhorar o desempenho do Eta foram realizadas análises no código do mesmo, buscando pontos de melhorias e partes que consomem um grande período de tempo.

Devido à grande quantidade de informações com que este modelo trabalha, foi possível encontrar nos códigos laços de repetição com uma complexidade chegando até $O(n^4)$. Analisando essa alta complexidade em diversos lugares do modelo foram estudadas as tecnologias OpenMP e CUDA, pois com estas é possível obter ganho de desempenho sem que seja necessário alterar a lógica utilizada pelo modelo.

É possível que existam melhorias em relação aos cálculos meteorológicos que o modelo executa, porém no trabalho realizado, não pretende-se alterar a lógica existente nestes cálculos, pois os envolvidos neste projeto não possuem a quantidade necessária de conhecimento em estudos de meteorologia e física para que essas mudanças possam ocorrer sem prejudicar o resultado final gerado pelo modelo.

Nos estudos já realizados no modelo Eta, foi analisado o uso atual da tecnologia MPI e foram buscados pontos de melhorias para estas, porém o código apresenta uma boa escrita e está bem acoplado com o modelo, ficando pouco viável uma melhoria desse processo.

5.2. MODELO MPI COM OPENMP

Como citado na seção 3.2, o modelo Eta trabalha utilizando a biblioteca MPICH, que faz uso de paralelismo para otimizar e acelerar o processamento dos dados. A utilização dessas duas tecnologias pode ser encontrada na literatura como programação híbrida, onde são

usadas as melhores características de cada uma das tecnologias. Por exemplo, é possível utilizar a paralelização explícita de grandes tarefas utilizando MPI e somar usando a paralelização OpenMP para as tarefas mais simples. Tendo uma visão de alto nível, o programa apresenta-se hierarquicamente estruturado com uma série de tarefas MPI, cujo o código sequencial utiliza as diretivas OpenMP para fazer uso da memória compartilhada e do multiprocessamento [35]. Vide Figura 16.

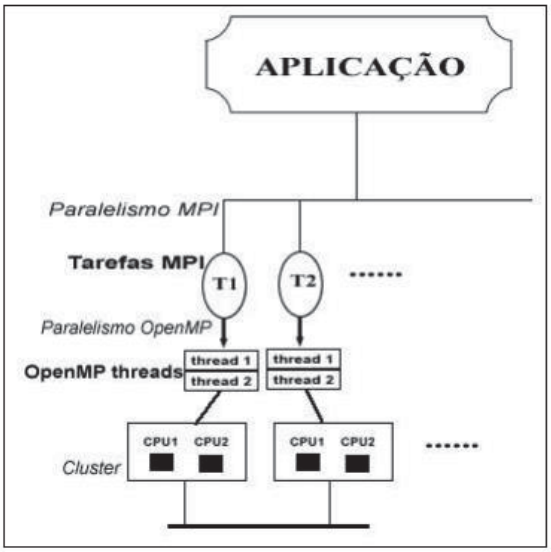

Figura 16. Modelo de paralelismo híbrido utilizando OpenMP com MPI [35]

Na Figura 16 é possível observar como funcionaria o comportamento da integração dos dois métodos de paralelismo trabalhando em forma conjunta, onde as tarefas são divididas utilizando MPI e dentro dos fluxos MPI existiria o paralelismo OpenMP fazendo uso de múltiplos núcleos da CPU. Essa solução se torna viável em um computador com múltiplos núcleos na CPU.

5.3. MODELO MPI COM CUDA

Utilizando uma solução semelhante a MPI com OpenMP, a utilização de CUDA para processamento na GPU pode ser estruturado de forma com que cada tarefa MPI tem por dever de computar parte do seu código sequencial em um bloco da GPU, com o objetivo de executar as tarefas de maior complexidade deixando que a CPU compute apenas os cálculos mais simples ou que contenham dependências que não possam ser processadas na GPU.

Outra possível solução utilizando MPI com CUDA é o uso de múltiplas GPU, onde cada processo MPI mande diversas instruções para serem processadas nas placas gráficas em paralelo [36], vide Figura 17.

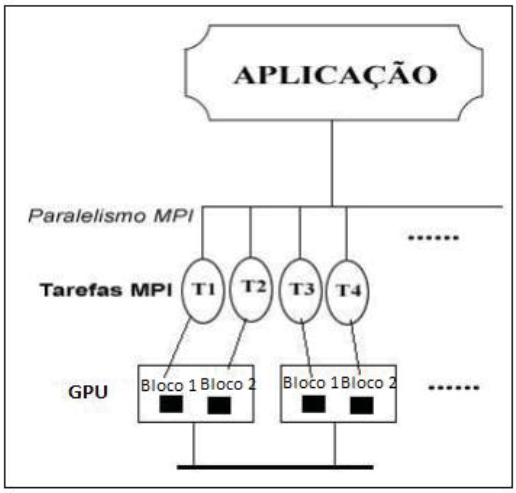

Figura 17. Modelo de solução fazendo uso de múltiplas tarefas MPI processando em diversas GPUs

Como pode ser observado na Figura 17, uma aplicação é dividida em diversas tarefas MPI, onde estas, são transmitidas e processadas pela GPU em diferentes blocos ou até mesmo em diferentes GPUs.

Uma outra alternativa, utilizando GPU, visando a melhoria do modelo é o uso da tecnologia OpenACC. Esta funciona de maneira similar ao OpenMP, isto é, utiliza-se funções específicas para controlar a paralelização de um determinado trecho do código, porém diferente do OpenMP esse recurso faz uso da GPU para o processamento ao invés da CPU. Essa tecnologia foi desenvolvida pela Cray, CAPS, NVIDIA e PGI buscando simplificar a programação paralela em GPUs [37]. Apesar de não ser o foco inicial deste trabalho, é possível que essa tecnologia possa ser útil ao modelo Eta.

5.4. MODELO MPI COM OPENMP COM CUDA

Outra solução possível a ser explorada é a utilização das três tecnologias ao mesmo tempo. Onde, utilizando Eta como exemplo, o MPI dividiria a área a ser processada e dentro de cada tarefa MPI seriam disparadas N *threads* OpenMP para processar e designar blocos de códigos para ser processados pela GPU, sendo uma ou mais GPU, como mostrado na Figura 18.

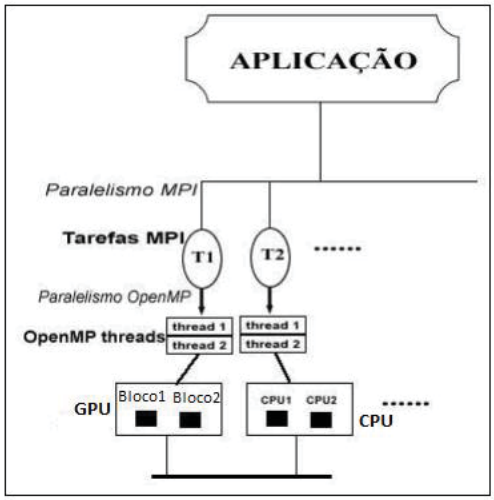

Figura 18. Modelo de solução fazendo uso das tecnologias MPI, OpenMP e CUDA em uma mesma aplicação

Como mostrado na Figura 18, múltiplas tarefas MPI são divididas em múltiplas *threads* OpenMP e algumas dessas utilizam a GPU para processamento enquanto outras executaram suas tarefas utilizando a CPU em paralelo.

5.5. CONSIDERAÇÕES SOBRE OS MODELOS

Os modelos de paralelismo citados nestes blocos são modelos teóricos que não haviam sido testados em relação ao modelo Eta, sem sabe se a comunicação e a agregação dessas técnicas de paralelismo iriam devidamente funcionar de acordo com o especificado neste documento.

O presente trabalho seguiu a tendência dos trabalhos relacionados apresentados, onde foi proposto e implementado um modelo de dois níveis de paralelismo no sistema de previsão numérica Eta, utilizando as tecnologias MPI e CUDA na implementação.

6. IMPLENTAÇÃO DO MODELO PROPOSTO

Neste capítulo é apresentado a implementação do modelo proposto realizada no código do modelo Eta e a validação deste juntamente com os experimentos iniciais realizados.

6.1. IMPLEMENTAÇÃO

A implementação citada nesta seção e nos resultados é a aplicação de uso de MPI juntamente com a tecnologia CUDA, devido ao fato que, a tecnologia OpenMP não pode ser integrada ao modelo Eta, por causa do uso de blocos de memória compartilhada (*Common Blocks*) que o modelo utiliza. A combinação de OpenMP e esses blocos de memória resultou em erros de processamentos das informações do modelo. A solução encontrada para esta questão seria a remoção dos blocos de memória compartilhada, necessitando assim, a reescrita de partes do modelo. [38]

O desenvolvimento de aplicações fazendo uso de GPU se dá primeiramente por meio da definição de número de blocos e a quantidade de *threads* que cada bloco vai computar. Dentro de cada bloco, cada *thread* possui um identificador único (ID) para cada dimensão, sendo eles *threadIdx%x*, *threadIdx%y* e *threadIdx%z*, para as dimensões X, Y e Z, respectivamente.

Os índices da matriz são calculados com base na ID de cada *thread*. As *threads* que possuem ID menor que o inicio da matriz ou maior que o final serão finalizadas, sendo utilizado exatamente o número de elementos da matriz a ser processado.

Foi alterado o código fonte (VTAD), cuja versão original possui 934 linhas de código e, com a adição da exploração da GPU passou para 1127 linhas. Foram implementados três *kernels* dentro da subrotina onde os principais cálculos de previsão do tempo do modelo Eta são realizados. Esses *kernels* quando chamados são executados no *device*, no caso, a GPU. Cada *kernel* possui um código cujo o resultado é igual ao executado pelo modelo sem o uso de GPU. Com isso, é necessária a passagens dos parâmetros para o processamento das informações, sendo estes parâmetros cópias das matrizes ou das variáveis envolvidas nos cálculos e as coordenadas referentes aos índices da GPU onde as informações serão computadas.

Após o processamento das informações, o resultado é copiado do *device* e atribuído para a matriz de resultado da implementação clássica e o fluxo de execução normal é retomado. Essa execução é reproduzida para cada processo MPI criado, podendo ser computada diversas partes da matriz ao mesmo tempo na GPU.

O *kernel* A é responsável por realizar por realizar a multiplicação de duas matrizes de três dimensões elemento por elemento e a multiplicação do valor resultado com um vetor de uma dimensão com N elementos, sendo sua complexidade O(m*n*p), onde m representa o número de linhas, n representa o número de colunas e p o número de elementos na profundida da matriz. O *kernel* B, por sua vez, realiza a operação de multiplicação de uma matriz de três dimensões com um vetor vezes a subtração de elementos de uma matriz de três dimensões por outros elementos dela mesma. Após essa operação é realizada a soma de três matrizes, sendo uma delas o resultado da operação anterior. A complexidade destas operações é O(m*n*p). Já o *kernel* C realiza uma multiplicação de duas matrizes índice a índice simplesmente, com complexidade O(n*m*p).

Durante uma iteração da execução do modelo um dos *kernels* A vira grid duas vezes, sendo executado em dois pontos diferentes do processamento. Os outros *kernels* (B e C) são executados apenas uma vez para cada iteração do processamento do modelo. Para cada execução dos *kernels* é transportada da memória do computador para a GPU a área a ser processada e as variáveis, vetores ou matrizes específicas de cada cálculo.

As modificações realizadas e os códigos originais do modelo Eta podem ser visualizados no Apêndice B.

6.2. VALIDAÇÃO

A validação do modelo proposto foi implementada pelo aluno Alex Mello como trabalho de conclusão [46] do curso de Ciência da Computação da UPF. Foi realizada implementação de um *kernel* a ser executado pela GPU e o ambiente onde os testes foram realizados foi um computador com CPU Intel Core i7 920 de 64 *bits*, com um *clock* de 2,66GHz, 4 *cores* e 8 *threads*, 8 Gb de RAM sistema operacional Linux, distribuição Ubuntu versão 16.04 LTS para processadores de 64 *bits*, GPU GeForce GTX 770 da Nvidia, possuindo 1536 CUDA *cores* e 2 Gb de memória de interface GDDR5, com largura de 256 *bits* e banda de 224.3 Gb por segundo.

Para os testes de validação foram utilizados dois tamanhos de áreas diferentes, uma com 201x251 elementos e outra com 101x151, apresentada na Figura 19, utilizando a resolução de 10 Km para ambas.

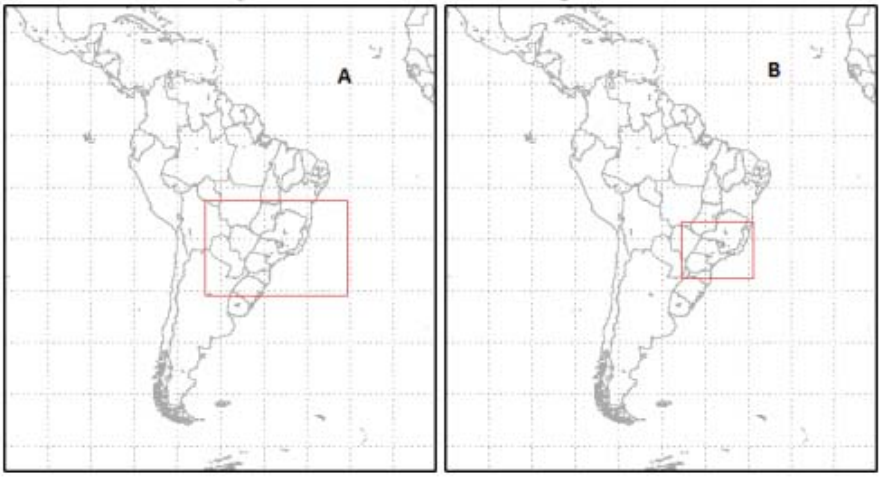

Figura 19. Visualização da área processada, pelo modelo Eta, para fins de validação.

A Figura 19, apresenta as áreas processadas para fins de validação, sendo a Figura 19A a representação da área maior (201x251) e a 19B a área menor (101x151). Os testes para estas áreas foram executados com dois números distintos de processos, sendo estes, sete e quatro processos, para uma quantidade de 6 horas de previsão. A Tabela 6 apresenta os resultados da validação.

Tabela 6. Tempo médio de execução em segundos em função da área coberta.

Execução/Área	Área menor (Figura 18B)	Área maior (Figura 18A)
7 processos	218,43	790.40
7 processos com CUDA	214,37	795.82
4 processos	217,51	805,32
4 processos com CUDA	215,16	788,16

A Tabela 6 apresenta os resultados dos testes, sendo o resultado a média de 50 execuções excluindo o menor e o maior resultado. É possível observar que a execução do modelo com quatro processos com CUDA apresentou um tempo menor, para ambas as áreas, em relação a versão sem as modificações. Para a área de 101x151 utilizando quatro processos, o tempo médio com CUDA foi aproximadamente 1,24% menor em relação a versão normal do modelo, já para a área de 201x251 o ganho foi de aproximadamente 2,13% na execução do modelo em comparação a sua versão fazendo uso apenas da GPU.

Para execução com sete processos, os testes apontaram uma redução no tempo de execução de aproximadamente 1,85% para área menor, já para área maior, houve um aumento de aproximadamente 0,68% no tempo de execução, cujo qual, pode ser explicada uma vez que a quantidade de elementos da área total a ser processada é maior que o número de CUDA cores existentes na GPU utilizada pelo maior número de processos competindo pelo seu uso.

As modificações da validação ocorreram em apenas dois pontos de melhoria do Eta, sendo apenas um *kernel*, utilizado para ambos os pontos. A complexidade deste trecho de código substituído é de O(m*n), correspondente a uma multiplicação de valores de diferentes matrizes.

Os resultados obtidos na validação foram promissores para a maior parte dos testes realizados, tornando válida novas implementações em outros pontos do código do modelo Eta, novos testes com diferentes áreas, mais horas de previsão e mais computadores para processamento.

7. EXPERIMENTOS REALIZADOS E RESULTADOS

Neste capítulo são apresentadas as quantidades de repetições dos experimentos, juntamente com os tamanhos das áreas, o número de horas processadas, informações sobre os ambientes computacionais utilizados e os resultados gerados após as execuções.

7.1. EXPERIMENTOS REALIZADOS

Para verificar os resultados gerados após as modificações no modelo Eta foram realizados testes com três tamanhos de áreas diferentes, mantendo a resolução de 10 km para cada uma das áreas. Vide Figura 20.

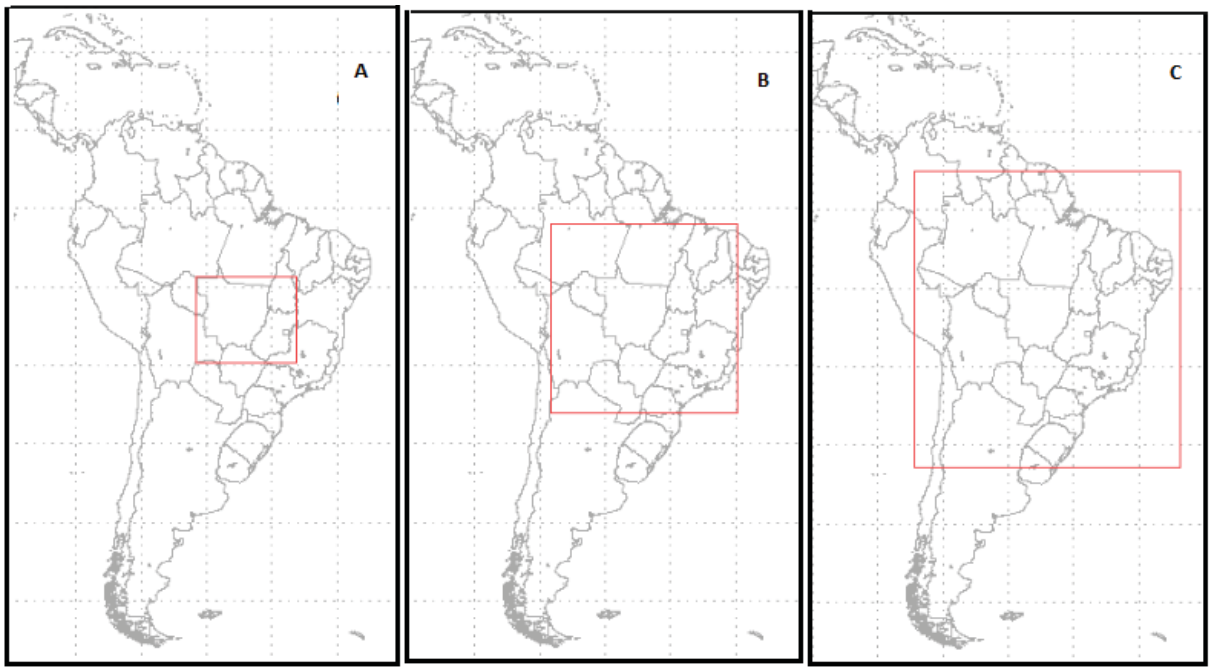

Figura 20. Representação das áreas processadas, pelo modelo Eta, nos experimentos realizados.

As Figura 20A, 20B e 20C representam a área de testes pequena com 101x159 elementos, média com 181x349 elementos e grande com 251x581 elementos, respectivamente. Para todas as áreas foram realizados testes com três diferentes horas de previsão, sendo estas 6, 12 e 24 horas para o ambiente de testes A e 6 e 8 horas para o ambiente de testes B. A diferença de horas de previsão do ambiente computacional A e B se dá devido ao fato de que decidiu-se por limitar o tempo de simulação dos experimentos a fim de garantir o prazo de conclusão do trabalho.

Para cada combinação de horas e área a ser processada foram utilizados diferentes números de processos lançados em cada computador. No ambiente de testes A foram lançadas baterias de testes com 40, 60 e 80 processos. Já no ambiente B foram realizados testes com 9, 15 e 30 processos.

No total foram realizadas 20 repetições de cada combinação de área processada, horas de previsão e números de processos, sendo retirados o maior e o menor valor de tempo de processamento resultante do modelo. Após isto, foi realizada a média dos demais valores. Uma validação estatística dos valores foi realizada e é descrita na seção 7.3.1.

7.2. AMBIENTE DE EXECUÇÃO DOS TESTES

Os recursos utilizados para testes da implementação deste trabalho foram fornecidos pela Universidade de Passo Fundo (UPF), sendo criados dois ambientes de testes distintos para verificar a qualidade e o desempenho da versão do Eta fazendo uso de GPUs, em relação a versão com apenas MPI. Os ambientes são denominados de A e B, sendo o primeiro junto ao Laboratório Central de Informática e, o segundo, localizado no Núcleo de Visualização e Modelagem Computacional no UPF-Parque.

O ambiente computacional de testes A fez uso de 20 computadores com CPU Intel Core i7-3770 com *clock* de 3,40 GHz contendo 4 núcleos físicos e 4 lógicos; 8 Gb de memória RAM; Sistema operacional Linux, distribuição Ubuntu versão 16.04 LTS para processadores 64 bits; GPU GeForce GT 630 possuindo 384 CUDA *cores*, 2 Gb de memória, interface 64 bit-DDR3, com 14.4 Gb de largura de banda por segundo [39]. Este ambiente contou com um 21º computador como servidor de I/O, CPU Intel Core i7 920 de 64 *bits*, com um *clock* de 2,66 GHz, 4 *cores* e 8 *threads*; 8 Gb de RAM; Sistema operacional Linux, distribuição Ubuntu versão 16.04 LTS para processadores de 64 bits; GPU GeForce GTX 770 da Nvidia, possuindo 1536 CUDA *cores* e 2 Gb de memória, interface 256-bits GDDR5 e largura de banda de 224.3 Gb por segundo [40].

O ambiente computacional de teste B foi montado com três computadores CPU Intel Core i7-6950X com *clock* de 3.5 GHz, possuindo 10 núcleos físicos e 10 lógicos; 32 Gb de memória RAM; Sistema operacional Linux, distribuição Ubuntu versão 16.04 LTS para processadores 64 bits. Dois computadores equipados com GPU GeForce TITAN X da Nvidia, com 3072 CUDA *cores* com 12 Gb de memória de interface 384-bits GDDR5 e largura de banda de 336.5 Gb por segundo [41] e o terceiro computador possui uma GPU GeForce Tesla K40 da Nvidia, com 2880 CUDA *cores*, exclusivos para processamento de dados, 12 Gb de

memória, interface 384-bits, com largura de banda 288 Gb por segundo [42] e outra GPU GTX 660 da Nvidia [43] para controle de vídeo, sem uso no processamento.

Tanto no ambiente A, quanto no B, as GPUs utilizadas possuem um limite de 1024 *threads* por bloco, dimensões máximas de 1024 e 1024, 64 para x, y e z, respectivamente, e *wrap size* de 32, significando que independentemente do tamanho de bloco, o número de *threads* sempre será múltiplo de 32 [39]. Apesar das GPUs do ambiente A e do B serem de arquiteturas diferentes, Kepler (GTX 770) no ambiente A, Maxwell 2.0 (TITAN X) e Kepler (Tesla K40) no ambiente B, estas possuem as mesmas dimensões máximas e o número de *threads* por bloco, não sendo necessária modificação na implementação dos *kernels* [44].

Para a compilação do modelo Eta fazendo uso de CUDA foi utilizado o compilador PGI edição comunitária, nas versões 16.5 para o ambiente computacional A e 17.10 para o ambiente computacional B. Estas edições do compilador são gratuitas no período de um ano [40]. Para que seja reconhecido o uso de CUDA na aplicação, é necessário adicionar a *flag* "-Mcuda" ao arquivo responsável por controlar as *flags* de compilação (Makefile) e ter instalado no computador o *driver* CUDA.

7.3. RESULTADOS DOS EXPERIMENTOS

Os resultados dos experimentos citados na seção 7.1 podem ser visualizados nos itens abaixo, sendo que nestes são apresentados os dados de variações das execuções e a média dos resultados gerados junto a uma análise destes dados.

7.3.1. Variações das execuções

Para verificar a validade dos dados resultantes dos experimentos foram realizadas 20 repetições, em ambos ambientes computacionais de cada combinação de horas de previsão com os diferentes números de processos. O ambiente computacional A foi ocupado por aproximadamente 5 dias (123 horas) consecutivos para a realização de todos os experimentos. O ambiente computacional B foi ocupado por aproximadamente 16 dias (392 horas) consecutivos, porém nesse tempo somente os experimentos com 6 e 8 horas haviam terminado, assim sendo, decidiu-se por limitar o tempo de simulação dos experimentos a fim de garantir o prazo de conclusão do trabalho.

A fim de verificar a confiança dos resultados obtidos, foi realizada uma análise estatística acerca das repetições dos casos testados em ambos os ambientes, usando o software

Statistical Package for the Social Sciences (SPSS). Foram testados 54 conjuntos de dados (repetições) no Ambiente A (27 com CUDA e 27 sem) e 36 no ambiente B (18 com CUDA e 18 sem). Para verificar a normalidade da distribuição dos dados de cada um dos experimentos, foi utilizado o teste de Kolmogorov-Smirnov, com um intervalo de confiança para a média de 95%. Num total de 90 conjuntos de dados verificados, em apenas sete casos a normalidade do conjunto foi rejeitada: cinco com CUDA e quatro sem. Nesse conjunto de sete casos, quase a metade teve a normalidade confirmada pelo teste de Shapiro-Wilk (Apêndice A). Dessa forma, a média pode ser considerada uma medida robusta para representar os dados destes experimentos.

Com o propósito de confirmar que as variações do tempo entre múltiplas execuções com os mesmos parâmetros não influenciam decisivamente nas variações do tempo entre diferentes parâmetros, foi calculado o coeficiente de variação das execuções. Dessa forma, é possível saber que variações não invalidam a análise da próxima seção.

No ambiente computacional A, a variação foi maior quando a área e a quantidade de horas são menores, o que implica em menor tempo de execução. Isso era esperado, uma vez que uma variação pequena, causada por latência de rede, por exemplo, causa uma diferença relativa maior em uma execução que dura pouco tempo do que em uma que leva várias horas. Salienta-se que todos os coeficientes de variação são menores que 4,8%, sendo assim, não afetam significativamente os resultados apresentados nas próximas seções.

No ambiente computacional B não ocorreram coeficientes de variação maiores que 2,2%. Isso pode ser explicado pelo fato de que o ambiente B é composto de menos computadores que o ambiente A (somente três, enquanto o ambiente A possui 20), e, assim sendo, uma menor variação da latência de rede era esperada.

Com relação às diferenças das médias obtidas com CUDA e sem CUDA, foi utilizado o Teste-T para amostras independentes, a fim de verificar se houve ou não diferenças significativas, com significância de 95%. Antes de aplicar o Teste-T, foi verificado se o mesmo poderia ser usado para comparações, por meio do Teste de Levene para igualdade das variâncias. Dos 45 conjuntos de casos analisados (27 no ambiente A e 18 no ambiente B), pode-se realizar a comparação por meio do Teste-T em aproximadamente 70% dos conjuntos, sendo que em oito, o Teste de Levene descartou a utilização do Teste-T para verificar se as médias obtiveram diferenças significativas, sendo três no ambiente A e cinco no ambiente B. Em relação à diferença significativa das médias, com CUDA versus sem CUDA, foi encontrada uma diferença significativa em aproximadamente 46% dos casos analisados (21 casos), sendo

que nos demais 16 casos, a diferença das médias foi considerada não significativa, para o intervalo de confiança de 95%.

7.3.2. Resultados no ambiente computacional A

Nesta seção são apresentados os resultados dos experimentos obtidos no ambiente computacional A. Estas informações podem ser visualizadas nas Tabelas 7, 8 e 9, onde são apresentados os resultados para as áreas grande, média e pequena, respectivamente. As tabelas a seguir são estruturadas pelo número de processos em relação às horas de previsão, com informações se foi feito o uso de CUDA ou não, os resultados da média das execuções junto do desvio padrão em parênteses e a coluna Valor-p é o resultado do Teste-T, este que representa uma diferença significativa onde o resultado é abaixo de 0,05. Vide Tabela 7.

Tabela 7. Resultados dos tempos de execução da área grande no ambiente computacional A.

		Horas de previsão					
		6		12		24	
Procs	Cuda	Média (DP)	Valor-p	Média (DP)	Valor-p	Média (DP)	Valor-p
80	Não	299,20 (6,22)	0,029	582,87 (3,74)+	0,001	1170,16 (8,58)	0,065
80	Sim	303,32 (5,88)		588,04 (5,39)+		1165,36 (7,36)	
60	Não	302,69 (3,79)	0,117	577,09 (4,05)	0,031*	1118,55 (6,89)	0,014
60	Sim	300,54 (4,63)		573,35 (6,24)		1111,89 (9,26)	
40	Não	275,36 (4,62)	0,006	544,04 (5,38)	0,640	985,32 (5,40)	0,000
40	Sim	271,46 (3,89)		543,23 (5,51)		998,59 (5,99)	

* valor desconsiderado para comparações entre médias.
\+ valores a uma distribuição não normal

A Tabela 7 apresenta os resultados dos experimentos para a área grande onde é possível notar que houve variações significativas com relação ao uso de CUDA, para os casos de 60 processos com 12 e 24 horas de previsão e com 40 processos com 6 horas de previsão. Os casos utilizando 80 processos com seis horas, e 40 processos com 24 horas obtiveram resultados significativos, porém estes aumentam o tempo de execução total do modelo fazendo uso da tecnologia CUDA.

Tabela 8. Resultados dos tempos de execução da área média no ambiente computacional A.

		Horas de previsão					
		6		12		24	
Procs	Cuda	Média (DP)	Valor-p	Média (DP)	Valor-p	Média (DP)	Valor-p
80	Não	183,58 (3,67)	0,001	357,67 (2,69)	0,003	700,14 (4,46)	0,033
80	Sim	187,24 (2,86)		361,06 (4,06)		704,06 (6,52)	
60	Não	172,32 (5,12)+	0,000	332,19 (3,47)	0,270	656,24 (7,19)	0,603
60	Sim	178,31 (2,80)		330,77 (4,51)		655,18 (5,49)	
40	Não	148,06 (3,16)	0,001	285,24 (4,21)	0,188	545,44 (6,79)	0,000*
40	Sim	152,63 (4,69)		283,59 (3,52)		571,61 (3,97)	

* valor desconsiderado para comparações entre médias
\+ valores a uma distribuição não normal

Na Tabela 8 pode-se observar os resultados obtidos fazendo uso da área média. Nesta tabela é possível observar que os resultados fazendo uso de CUDA não são significativos e as diferenças significativas presentes nesta tabela pioram o desempenho do modelo Eta quando realizada a integração com CUDA. Na análise dos dados gerados para área média foi possível notar também que o experimento com 6 horas de previsão, com 60 processos sem CUDA, não seguiu uma distribuição normal.

Tabela 9. Resultados dos tempos de execução da área pequena no ambiente computacional A.

		Horas de previsão					
		6		12		24	
Procs	Cuda	Média (DP)	Valor-p	Média (DP)	Valor-p	Média (DP)	Valor-p
80	Não	102,41 (3,33)	0,367	191,67 (6,55)	0,140	388,51 (3,67)	0,010
80	Sim	103,26 (2,47)		194,65 (5,93)		385,47 (3,44)	
60	Não	88,52 (1,87)	0,032*	169,56 (4,83)	0,018	334,22 (3,42)	0,556
60	Sim	90,58 (3,72)		166,31 (3,28)		333,46 (4,57)	
40	Não	73,76 (2,74)	0,001	144,49 (5,46)	0,406	279,31 (5,94)	0,000
40	Sim	77,39 (3,64)		143,15 (4,59)		289,22 (4,21)	

* valor desconsiderado para comparações entre médias

Para a área pequena, os resultados apresentados na Tabela 9 mostram uma diferença significativa em relação ao uso de CUDA nos casos de 24 horas de previsão com 80 processos e 12 horas de previsão com 60 processos. Outros resultados significativos que mostram uma piora do desempenho do modelo podem ser observados nos casos de 60 processos com seis horas, 40 processos com seis e 24 horas de previsão.

As informações citadas acima podem ser visualizadas na Figura 21 na forma de gráficos de barras, onde cada faceta é uma combinação do tamanho da área (grande, média ou pequena) e o número de horas (6, 12, 24). O eixo Y apresenta a quantidade de tempo em segundos da média das execuções e o eixo X apresenta o número de processos, finalmente, a cor indica se a tecnologia CUDA estava ou não sendo utilizada no modelo.

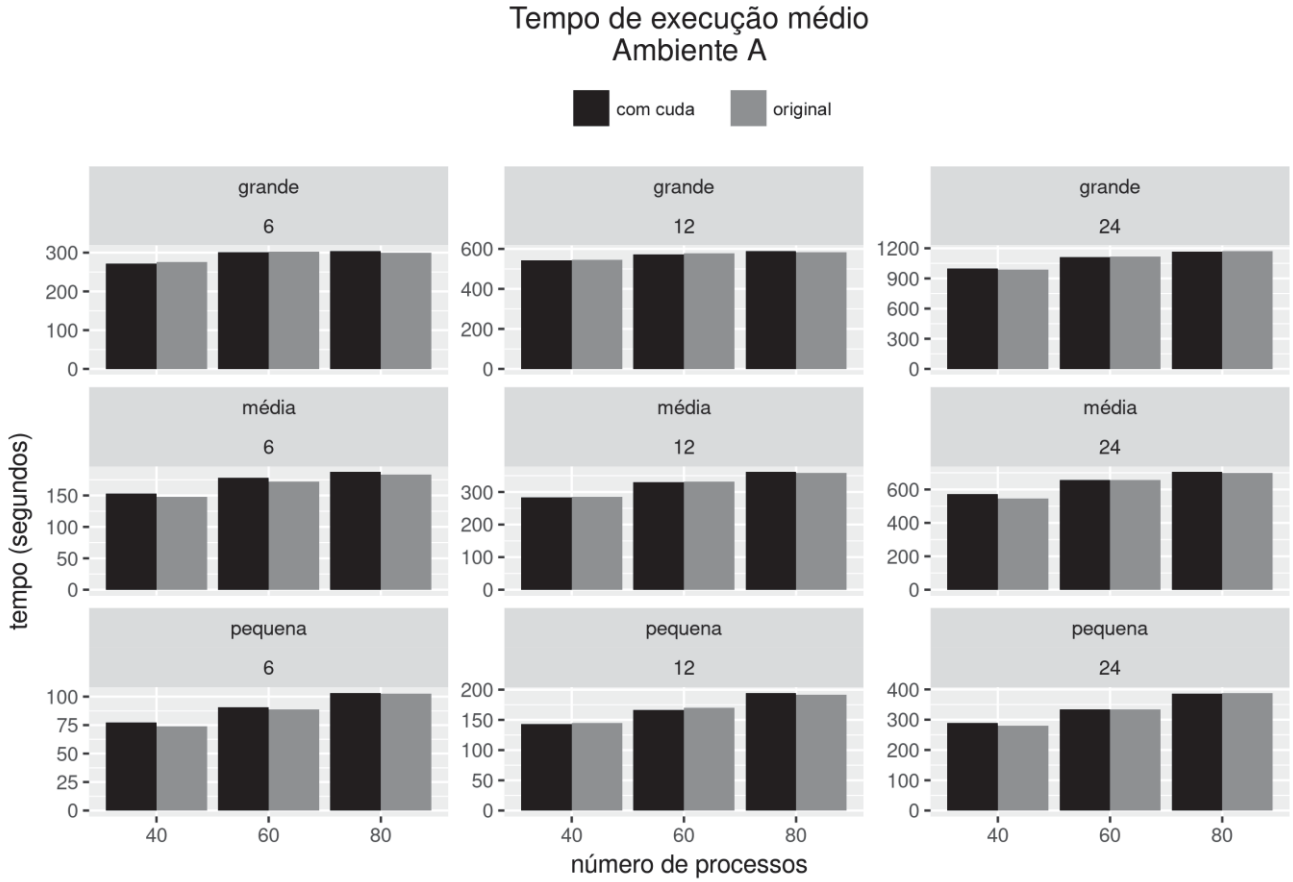

Figura 21. Gráfico de facetas representado a média dos resultados das execuções no ambiente A.

É possível notar na Figura 21 que o uso ou não da tecnologia CUDA no modelo teve pouco efeito nos tempos de execução, piorando este ou ficando abaixo de 3% de melhora (área pequena, 60 processos e 12 horas).

Um resultado inesperado obtido é que o aumento do número de processos do modelo frequentemente implica no aumento do tempo total de execução do modelo, em outras palavras, o uso de um número maior de processos é ineficiente, sendo o *overhead* dos processos maior que o ganho da paralelização. Este resultado pode estar ocorrendo devido ao número de servidores de I/O ser apenas um, onde este estaria sendo sobrecarregado, causando assim uma queda de desempenho da aplicação.

O tempo de execução foi linear em relação ao aumento do número de horas a serem processadas, o que era esperado, porém, o mesmo não pode ser dito do aumento do tempo de execução em relação ao crescimento da área. A diferença da área pequena para a área média é de cerca de quatro vezes e a diferença da área média para a grande é de cerca de duas vezes, entretanto o aumento do tempo de execução não seguiu estes fatores. Tanto da área pequena para média, quanto da área média para grande, o crescimento do tempo de execução foi de menos de duas vezes.

Um fator que pode ter influenciado nos resultados obtidos é o acesso de múltiplos processos a uma mesma GPU. Esse processo ocorre devido ao fato de que, para o ambiente computacional A, foram lançados dois, três e quatro processos por computador e todos eles realizavam acesso a uma mesma GPU, ocorrendo assim, uma serialização das tarefas a serem executadas, podendo prejudicar o tempo de execução do modelo.

Todos os dados brutos gerados por estes experimentos estão disponíveis no apêndice deste trabalho, junto às média das execuções, o desvio padrão e o coeficiente de variação dos dados.

7.3.3. Resultados no ambiente computacional B

Os testes no ambiente computacional B foram previstos para serem realizados da mesma forma que no ambiente computacional A. A expectativa era que, com máquinas de maior capacidade computacional, tanto em relação a CPU quanto em relação a GPU, os resultados em relação ao tempo de execução fossem maiores. Ademais, o menor número de computadores (apenas três), favorece um menor tempo consumido em comunicações.

Conforme já foi comentado, os resultados preliminares foram muito acima do esperado, fazendo com que os testes fossem modificados em relação ao tempo de previsão, sendo realizados apenas para 6h e 8h. Esses valores não apresentavam os resultados esperados, em relação a capacidade computacional superior do ambiente B para o A, porém os testes foram finalizados e as análises referentes à distribuição normal das repetições e diferenças significativas foram concluídas.

A suspeita está relacionada aos ativos de comunicação de rede e configuração da mesma, tarefa fora da responsabilidade e controle do autor. Os tempos de execução ficaram muito acima dos encontrados no Ambiente A, tanto com uso de CUDA, quanto na execução da implementação não modificada, a despeito do melhor hardware e menor quantidade de

computadores (Apêndice A). Dessa forma, os valores de tempo de execução não foram analisados em relação a ganho ou perda de desempenho no Ambiente computacional B.

7.4. CONSIDERAÇÕES SOBRE OS RESULTADOS

Os resultados obtidos devem ser analisados levando em consideração as especificidades dos ambientes computacionais onde foram executados. No ambiente A, foi usada uma rede que congrega aproximadamente 500 máquinas, que poderiam estar segmentadas numa rede isolada, conforme solicitado aos responsáveis, mas não foi possível. São máquinas com processador relativamente antigo e GPUs muito antigas, com baixo desempenho. Os resultados em relação ao tempo de execução de ambos os modelos não foram favoráveis em virtude da rede, recursos de GPU e por serem lançados diversos *kernels* simultâneos na mesma GPU. No ambiente B havia a expectativa de melhores resultados por ser um ambiente de melhores recursos computacionais e menor número de nós de rede. Porém, conforme descoberto a partir dos tempos de execução elevados, a instalação dos computadores compartilhou uma rede com mais de 2500 máquinas, percorrendo nas comunicações diversos ativos de rede. Dessa forma, os resultados não foram levados em consideração e não podem ser comparados. Tais problemas em relação aos ambientes de execução não ocorreriam caso os testes fossem realizados em ambientes controlados, específicos para o processamento de Alto Desempenho, tais como *clusters*, como por exemplo o Tupã [45].

O aumento do número de processos por computador de um processo para dois processos no ambiente A (dados de experimentos preliminares não apresentados na seção anterior) resultaram em uma diminuição do tempo de execução (lembrando que o número de computadores no mesmo ambiente A é sempre o mesmo). Entretanto, incrementar ainda mais o número de processos por computador (para três ou quatro no ambiente A) aumentou o tempo total de execução. Acredita-se que níveis maiores de paralelismo dentro do mesmo computador esbarram na competição por cache L2 e L3, entre outros recursos.

Outro fator que pode ter comprometido os resultados de forma mais expressiva no tempo da execução do modelo é o acesso de todos os processos a uma GPU, comentado na análise dos resultados do ambiente computacional A, causando assim uma serialização de itens a serem processados e, consequentemente, uma queda no desempenho. No ambiente B, uma GPU possui controle de acesso para múltiplos *kernels* serem executados ao mesmo tempo, porém por problemas de rede os resultados finais de tempo de execução não foram comparados.

Vale ressaltar que foram realizadas comparações dos arquivos binários de saída do modelo Eta com o objetivo de verificar a validade dos resultados obtidos após a integração da tecnologia CUDA em comparação com o software sem estas modificações.

8. CONSIDERAÇÕES FINAIS

8.1. PRODUÇÕES DECORRENTES

O presente trabalho, no decorrer do seu desenvolvimento, gerou produções científicas para os seguintes eventos:

- ERAD 2017 – Fórum de Iniciação Científica.
- ERAD 2017 – Fórum de Pós-Graduação.
- Trabalho de conclusão do aluno do curso de Ciência da Computação, Alex L. Mello.
- ERAD 2018 – Fórum de Pós-Graduação.
- ERAD 2018 – Proposta de minicurso focado em programação na linguagem Fortran utilizando MPI e OpenMP.

Além dos eventos citados acima será encaminhado um artigo demonstrando os resultados para o 17º WPerformance.

8.2. CONCLUSÕES

O presente trabalho buscou encontrar uma forma de melhoria do desempenho do PND Eta, ampliando o número de *kernels* existentes no modelo e aumentando tanto a área a ser prevista, o número de horas de previsão e os testes realizados em relação ao trabalho anterior realizado no trabalho de conclusão de curso do aluno, do curso de Ciência da Computação da UPF, Alex Mello[46].

O modelo Eta apresentou-se complexo no seu entendimento e nas modificações realizadas em seu código. A integração da tecnologia CUDA, visando delegar parte do processamento do modelo para a GPU, se deu através da escrita de quatro *kernels* de pontos em que o modelo demorava mais tempo de processamento. O resultado obtido das execuções apresenta variações significativas com a integração de CUDA no ambiente, apesar de uma piora no tempo de desempenho em alguns casos.

É necessário levar em consideração que ambos os ambientes computacionais onde os experimentos foram realizados não possuíam uma infraestrutura esperada para testes de aplicações visando alto desempenho, assim, influenciando os resultados. Tendo isso em mente,

ressalta-se a importância de novos trabalhos, seguindo a linha deste, para que seja possível obter um resultado com maior redução do tempo de exceção e com menos variáveis de instabilidade de infraestrutura que possam influenciar os resultados.

Outra conclusão obtida é que nem sempre o uso do número total de processadores de um *cluster* resulta em um tempo de processamento menor, sendo necessária a avaliação do número de horas a serem processados e o tamanho da área do experimento.

8.3. TRABALHOS FUTUROS

Seguindo a mesma ideia proposta e implementada por este trabalho, é possível explorar um ganho de desempenho na implementação nos demais laços presentes do código do modelo, aplicando, assim, a solução ao uso da tecnologia MPI e CUDA.

Visando habilitar a integração com diretivas OpenMP, seria necessária a remoção dos blocos de memória compartilhadas (*common blocks*) presentes na implementação do modelo. Enquanto o Eta fizer uso desta técnica, a implantação das diretivas se torna inviável, resultado em erros de compilação do modelo e podendo resultar na incoerência de dados pós processados. [38]

Existem diversos possíveis trabalhos futuros visando o modelo Eta, como, por exemplo, a remoção da dependência exclusiva do uso da tecnologia MPI, podendo, assim, o modelo ser executado em apenas um processador. Este processo, teoricamente, diminuiria o desempenho do modelo mas abriria espaço para uma implementação utilizando apenas a GPU para processamento, tendo, assim, uma nova alternativa de paralelismo de um nível podendo resultar em um ganho de desempenho, por vezes, maior que a implementação realizada fazendo uso de MPI.

REFERÊNCIAS

[1] DEMIR, N., MAHMUD, S. F. 2005. “Agro-climatic Conditions and Technical Inefficiencies in Agriculture.” Canadian Journal of Agricultural Economics 50: 269–80.

[2] Nation Centers for Environmental Information. Disponível em: < https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/global-forcast-system-gfs > Acesso em: Outubro. 2016.

[3] CHOU, S. C. Modelagem Numérica (Introdução a PNT). 2010. Disponível em: <https://goo.gl/tL7GPZ>. Acesso em: Outubro. 2016

[4] MOURA, R. G.; HERDIES, D. L.; et al. AVALIAÇÃO DO MODELO REGIONAL Eta UTILIZANDO AS ANÁLISES DO CPTEC E NCEP, Revista Brasileira de Meteorologia, v.25, n.1, 46 - 53, 2010.

[5] INPE. Eta Model. 1995–2006. Disponível em: <http://Etamodel.cptec.inpe.br/history/> Acesso em: Outubro. 2016.

[6] WHAT IS GPU-ACCELERATED COMPUTING. Disponível em: <http://www.nvidia.com/object/what-is-gpu-computing.html>. Acesso em: Setembro. 2017

[7] NVIDIA. Disponível em: < http://www.nvidia.com.br/object/cuda_home_new_br.html> Acesso em: Outubro. 2016.

[8] Programming Guide:: CUDA Toolkit Documentation. Disponível em: <http://docs.nvidia.com/cuda/cuda-c-programming-guide/#axzz4O2i7Qy7m > Acesso em: Outubro. 2016.

[9] LYNCH, P. (March 2008). "The origins of computer weather prediction and climate modeling" (PDF). Journal of Computational Physics. University of Miami. 227 (7): 3431–44. Bibcode: 2008 JCoPh.227.3431L. doi:10.1016/j.jcp.2007.02.034. Retrieved 2010-12-23.

[10] LYNCH, P. (2006). "Weather Prediction by Numerical Process". The Emergence of Numerical Weather Prediction. Cambridge University Press. pp. 1–27. ISBN 978-0-521-85729-1.

[11] ESPACIAL, I. N. de P. Laboratório de modelagem atmosférica (LMA). 2015. Disponível em:<http://lma.cptec.inpe.br/>. Acesso em: Março. 2017

[12] CHOU, S. C. et al. Evaluation of the Eta Simulations Nested in Three Global Climate Models. American Journal of Climate Change, Scientific Research Publishing, v. 03, n. 05, p. 438–454, dez. 2014. ISSN 2167-9495. Disponível em: <http://www.scirp.org/journal/PaperInformation.aspx?PaperID=52877&#abstract>. Acesso em: Novembro. 2016

[13] MACHADO, V. L. Análise do impacto da utilização da previsão do tempo corrigida pelo método Model Output Calibration em modelos de doenças da maçã. Universidade de Passo Fundo. 2017

[14] CHOU, S. C. MODELO REGIONAL Eta. Disponível em: <http://climanalise.cptec.inpe.br/~rclimanl/boletim/cliesp10a/27.html> Acesso em: Outubro. 2016.

[15] NVIDIA Introduces Tesla K80 With Two GK210 GPUs – World’s Fastest Accelerator With 2.9 TFlops of Compute Disponivel em: < https://wccftech.com/nvidia-introduces-

tesla-k80-gk210-gpus-worlds-fastest-accelerator-29-tflops-compute/ > Acesso em: Janeiro. 2018

[16] NICKOLLS, J. BUCK, I.; GARLAND, M.; SKADRON, K. *Scalable parallel programming with CUDA*. ACM Queue, 6 (2) (2008), pp. 40–53

[17] History of FORTRAN and FORTRAN II. Disponível em: <http://www.softwarepreservation.org/projects/FORTRAN/index.html#By_FORTRAN_project_members> Acesso em: Outubro. 2016.

[18] JORGENSEN, Ed. Introduction to Programming using Fortran 95. Las Vegas: University Of Nevada, 2014. 177 p.

[19] PGI Compilers & Tools. Disponível em: <http://www.pgroup.com/> Acesso em Outubro. 2016.

[20] Message Passing Interface (MPI). Disponível em: <https://computing.llnl.gov/tutorials/mpi/#What> Acesso em: Outubro. 2016.

[21] HERMANNS, M.; Parallel Programming in Fortran 95 using OpenMP. School of Aeronautical Engineering. Universidad Politécnica de Madrid. Spain: April. 2002.

[22] GABRIEL, E.; FAGG, G. E.; et al. *Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation In Proceedings*, 11th European PVM/MPI Users' Group Meeting (September 2004), pp. 97-104.

[23] J. Marshall, A. Adcroft, C. Hill, L. Perelman, and C. Heisey, "*A Finite-Volume Incompressible Navier-Stokes Model for Studies of Ocean on Parallel Computers*," *Journal of Geophysical Research*, vol. 102, no. C3, pp. 5753–5766, 1997

[24] SCHEPKE, C.; MAILLARD, N. *Exploring Multi-level Parallelism in Atmospheric Applications. 2012 13th Symposium on Computer Systems.* 41 - 47.

[25] ASUNCIÓN, M.; MANTAS, J.; CASTRO, M.; FERNANDEZ-NIETO, E.; *An MPI-CUDA implementation of an improved Roe method for two-layer shallow water systems. Journal of Parallel and Distributed Computing. Volume 72, Issue 9, September 2012.* 1065-1072.

[26] DIAZ, J.; MUNOZ-CARO, C.; NINO, A. *A Survey of Parallel Programming Models and Tools in the Multi and Many-Core Era. IEEE Transactions on Parallel and Distributed Systems (Volume: 23, Issue: 8, Aug. 2012).* 1369 - 1386.

[27] OYARZUN, G.; BORRELL, R.; GOROBETS, A.; OLIVA, A. *MPI-CUDA sparse matrix–vector multiplication for the conjugate gradient method with an approximate inverse preconditioner. Computers & Fluids* 92 (2014) 244–252

[28] HUANG, H.; WANG, L.; LEE, E.; CHEN, P. *An MPI-CUDA Implementation and Optimization for Parallel Sparse Equations and Least Squares (LSQR). Procedia Computer Science* 9 (2012) 76 – 85.

[29] KIJSIPONGSE, E.; U-RUEKOLAN, S.; NGAMPHIW, C.; TONGSIMA, S. *Efficient large Pearson correlation matrix computing using hybrid MPI/CUDA. Computer Science and Software Engineering (JCSSE), 2011 Eighth International Joint Conference on.* 237 – 241.

[30] APOSTAL, D.; FOERSTER, K.; CHATTERJEE, A.; DESELL, T. *Password Recovery Using MPI and CUDA. High Performance Computing (HiPC), 2012 19th International Conference on*. 1-9.

[31] PENNYCOOK, S.; HAMMOND, S.; JARVIS, S.; *Performance Analysis of a Hybrid MPI/CUDA. Implementation of the NASLU Benchmark. ACM SIGMETRICS Performance Evaluation Review - Special issue on the 1st international workshop on*

performance modeling, benchmarking and simulation of high performance *computing systems (PMBS 10) archive. Volume 38 Issue 4, March 2011*. 23-29.

[32] BUENO, J.; PLANAS, J.; DURAN, A.; BADIA, R.; MARTORELL, X.; AYGUADÉ, E.; LABARTA, J. *Productive Programming of GPU Clusters with OmpSs. Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*. 557- 568.

[33] J. M. Perez, R. M. Badia, and J. Labarta, "*A dependencyaware task-based programming environment for multi-core architectures*," IEEE Int. *Conference on Cluster Computing*, pp. 142–151, September 2008.

[34] HUISMANN, I.; STILLER, J.; FRÖHLICH, J. *Two-level parallelization of a fluid mechanics algorithm exploiting hardware heterogeneity. Computers & Fluids* 117 (2015) 114–124.

[35] RABENSEIFNER, R., HAGER, G., JOST, G. 2009. Communication Characteristics and Hybrid MPI/OpenMP Parallel Programming on Clusters of Multi-core SMP Nodes. Proceedings of the Cray Users Group Conference 2009 (CUG 2009), Atlanta, GA, USA, May 4-7, 2009.

[36] KRAUS, J.; MESSMER, P.; MULTI GPU PROGRAMMING WITH MPI. GPU Technology Conference. Disponivel em: <http://on-demand.gputechconf.com/gtc/2014/presentations/S4236-multi-gpu-programming-mpi.pdf> Acesso em: Outubro. 2016.

[37] OpenACC 2.0 Spec. Disponvel em: <www.openacc.org> Acesso em: Outubro. 2016.

[38] Issue with common block in OpenMP parallel programming - StackOverflow. Disponível em: <https://stackoverflow.com/questions/32446564/issue-with-common-block-in-openmp-parallel-programming>. Acesso em: Fevereiro. 2017.

[39] NVIDIA GeForce GT 630| Nvidia. Disponível em: <http://www.nvidia.com.br/object/geforce-gt-630-br.html#pdpContent=2> Acesso em: Setembro. 2017

[40] Placa de vídeo GTX 770 com GPU Boost 2.0| GeForce | NVIDIA. Disponível em: <http://www.nvidia.com.br/object/geforce-gtx-770-br.html#pdpContent=2>Acesso em: Setembro. 2017

[41] GeForce GTX TITAN X Graphics card | GeForce | Nvidia Disponível em: <http://www.nvidia.com.br/object/geforce-gtx-titan-x-br.html#pdpContent=2> Acesso em: Setembro. 2017

[42] Tesla K40 GPU Active Accelerator. Disponível em: <https://www.nvidia.com/content/PDF/kepler/Tesla-K40-Active-Board-Spec-BD-06949-001_v03.pdf> Acesso em: Setembro. 2017

[43] Placa de vídeo GeForce GTX 660 com Tecnologia Kepler | Nvidia. Disponível em: <http://www.nvidia.com.br/object/geforce-gtx-660-br.html> Acesso em: Setembro. 2017

[44] Developer Zone - CUDA Toolkit Documentation. Disponivel em: <http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#compute-capabilities> Acesso em: Setembro. 2017

[45] Supercomputador - CPTEC/INPE. Disponível em:<http://www.cptec.inpe.br/supercomputador/> Acesso em: Fevereiro. 2018

[46] MELLO, A.; REBONATTO, M. T.; Alternativa à paralelização do modelo Eta: um estudo de caso com CUDA. Universidade de Passo Fundo. Dezembro, 2016.

APÊNDICE A - RESULTADOS BRUTOS DAS EXECUÇÕES DOS EXPERIMENTOS

Legenda Ambiente A			
Área	Tempo de previsão	Processos	Cuda
1 - Grande	1 - 24 h	1 – 80	1 – Sem
2 - Média	2 - 12h	2 - 60	2 - Com
3 - Pequena	3 - 6h	3 - 40	

Ordem	**Área**	**Tempo de previsão**	**Processos**	**Cuda**	**Tempo de execução**
1	1	1	1	1	1167,30
2	1	1	1	1	1179,87
3	1	1	1	1	1175,27
4	1	1	1	1	1173,91
5	1	1	1	1	1166,02
6	1	1	1	1	1170,14
7	1	1	1	1	1176,66
8	1	1	1	1	1182,81
9	1	1	1	1	1167,44
10	1	1	1	1	1174,45
11	1	1	1	1	1168,37
12	1	1	1	1	1179,64
13	1	1	1	1	1159,46
14	1	1	1	1	1176,57
15	1	1	1	1	1154,30
16	1	1	1	1	1157,54
17	1	1	1	1	1173,28
18	1	1	1	1	1160,72
19	1	1	1	1	1158,49
20	1	1	1	1	1180,97
21	1	1	1	2	1157,75
22	1	1	1	2	1171,59
23	1	1	1	2	1162,95
24	1	1	1	2	1169,47
25	1	1	1	2	1170,39
26	1	1	1	2	1167,70
27	1	1	1	2	1176,79

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
28	1	1	1	2	1172,56
29	1	1	1	2	1162,94
30	1	1	1	2	1166,31
31	1	1	1	2	1170,97
32	1	1	1	2	1164,23
33	1	1	1	2	1171,44
34	1	1	1	2	1160,43
35	1	1	1	2	1151,47
36	1	1	1	2	1164,34
37	1	1	1	2	1171,29
38	1	1	1	2	1156,09
39	1	1	1	2	1169,36
40	1	1	1	2	1149,13
41	1	1	2	1	1131,53
42	1	1	2	1	1119,81
43	1	1	2	1	1118,35
44	1	1	2	1	1121,25
45	1	1	2	1	1131,19
46	1	1	2	1	1108,64
47	1	1	2	1	1113,76
48	1	1	2	1	1118,55
49	1	1	2	1	1117,81
50	1	1	2	1	1123,15
51	1	1	2	1	1123,89
52	1	1	2	1	1118,69
53	1	1	2	1	1105,26
54	1	1	2	1	1116,61
55	1	1	2	1	1113,27
56	1	1	2	1	1121,62
57	1	1	2	1	1126,21
58	1	1	2	1	1108,18
59	1	1	2	1	1117,21
60	1	1	2	1	1115,96
61	1	1	2	2	1119,05
62	1	1	2	2	1100,97
63	1	1	2	2	1100,62
64	1	1	2	2	1118,07
65	1	1	2	2	1109,79
66	1	1	2	2	1114,41
67	1	1	2	2	1088,84
68	1	1	2	2	1118,19

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
69	1	1	2	2	1112,20
70	1	1	2	2	1108,55
71	1	1	2	2	1121,76
72	1	1	2	2	1113,21
73	1	1	2	2	1110,75
74	1	1	2	2	1131,13
75	1	1	2	2	1112,40
76	1	1	2	2	1103,53
77	1	1	2	2	1109,96
78	1	1	2	2	1105,61
79	1	1	2	2	1121,21
80	1	1	2	2	1117,50
81	1	1	3	1	981,79
82	1	1	3	1	990,55
83	1	1	3	1	984,49
84	1	1	3	1	981,09
85	1	1	3	1	982,85
86	1	1	3	1	990,39
87	1	1	3	1	979,15
88	1	1	3	1	987,62
89	1	1	3	1	982,94
90	1	1	3	1	996,11
91	1	1	3	1	981,54
92	1	1	3	1	980,41
93	1	1	3	1	989,53
94	1	1	3	1	983,85
95	1	1	3	1	987,86
96	1	1	3	1	975,78
97	1	1	3	1	992,31
98	1	1	3	1	992,52
99	1	1	3	1	986,77
100	1	1	3	1	978,94
101	1	1	3	2	1008,78
102	1	1	3	2	1002,49
103	1	1	3	2	999,08
104	1	1	3	2	997,19
105	1	1	3	2	999,87
106	1	1	3	2	1000,90
107	1	1	3	2	996,51
108	1	1	3	2	1002,56
109	1	1	3	2	996,62

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
110	1	1	3	2	996,94
111	1	1	3	2	994,07
112	1	1	3	2	1004,22
113	1	1	3	2	983,00
114	1	1	3	2	994,23
115	1	1	3	2	1002,91
116	1	1	3	2	1002,21
117	1	1	3	2	1004,84
118	1	1	3	2	1001,19
119	1	1	3	2	997,28
120	1	1	3	2	986,95
121	1	2	1	1	586,64
122	1	2	1	1	587,05
123	1	2	1	1	575,06
124	1	2	1	1	584,34
125	1	2	1	1	577,41
126	1	2	1	1	586,31
127	1	2	1	1	585,12
128	1	2	1	1	587,15
129	1	2	1	1	581,17
130	1	2	1	1	580,21
131	1	2	1	1	579,86
132	1	2	1	1	579,44
133	1	2	1	1	585,06
134	1	2	1	1	578,95
135	1	2	1	1	578,66
136	1	2	1	1	585,6
137	1	2	1	1	581,71
138	1	2	1	1	586,19
139	1	2	1	1	585,07
140	1	2	1	1	586,39
141	1	2	1	2	581,52
142	1	2	1	2	585,43
143	1	2	1	2	585,92
144	1	2	1	2	593,84
145	1	2	1	2	602,83
146	1	2	1	2	594,08
147	1	2	1	2	584,91
148	1	2	1	2	588,69
149	1	2	1	2	597,69
150	1	2	1	2	588,73

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
151	1	2	1	2	587,32
152	1	2	1	2	585,08
153	1	2	1	2	589,21
154	1	2	1	2	587,99
155	1	2	1	2	583,95
156	1	2	1	2	585,01
157	1	2	1	2	587,06
158	1	2	1	2	585,75
159	1	2	1	2	585,36
160	1	2	1	2	580,53
161	1	2	2	1	580,36
162	1	2	2	1	578,03
163	1	2	2	1	574,73
164	1	2	2	1	576,39
165	1	2	2	1	577,31
166	1	2	2	1	572,14
167	1	2	2	1	574,96
168	1	2	2	1	572,94
169	1	2	2	1	577,71
170	1	2	2	1	578,16
171	1	2	2	1	572,44
172	1	2	2	1	579,96
173	1	2	2	1	585,63
174	1	2	2	1	577,94
175	1	2	2	1	578,39
176	1	2	2	1	575,45
177	1	2	2	1	574,9
178	1	2	2	1	571,48
179	1	2	2	1	575,75
180	1	2	2	1	587,12
181	1	2	2	2	579,95
182	1	2	2	2	574,58
183	1	2	2	2	567,83
184	1	2	2	2	572,89
185	1	2	2	2	566,67
186	1	2	2	2	566,88
187	1	2	2	2	579,8
188	1	2	2	2	567,78
189	1	2	2	2	577,57
190	1	2	2	2	579,42
191	1	2	2	2	581,18

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
192	1	2	2	2	576,7
193	1	2	2	2	575,32
194	1	2	2	2	577,56
195	1	2	2	2	566,74
196	1	2	2	2	585,35
197	1	2	2	2	573,09
198	1	2	2	2	565,36
199	1	2	2	2	565,27
200	1	2	2	2	567,08
201	1	2	3	1	546,54
202	1	2	3	1	552,96
203	1	2	3	1	544,61
204	1	2	3	1	538,82
205	1	2	3	1	542,71
206	1	2	3	1	542,92
207	1	2	3	1	545,39
208	1	2	3	1	540,05
209	1	2	3	1	539,31
210	1	2	3	1	551,1
211	1	2	3	1	542,04
212	1	2	3	1	541,33
213	1	2	3	1	548,34
214	1	2	3	1	536,13
215	1	2	3	1	543,15
216	1	2	3	1	538,94
217	1	2	3	1	543,35
218	1	2	3	1	541,06
219	1	2	3	1	558,75
220	1	2	3	1	543,28
221	1	2	3	2	548,23
222	1	2	3	2	546,22
223	1	2	3	2	548,82
224	1	2	3	2	545,29
225	1	2	3	2	550,91
226	1	2	3	2	534,91
227	1	2	3	2	538,42
228	1	2	3	2	535,12
229	1	2	3	2	546,48
230	1	2	3	2	551,04
231	1	2	3	2	549,74
232	1	2	3	2	549,69

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
233	1	2	3	2	539,54
234	1	2	3	2	535,3
235	1	2	3	2	539,74
236	1	2	3	2	541,04
237	1	2	3	2	542,55
238	1	2	3	2	541,24
239	1	2	3	2	537,12
240	1	2	3	2	543,11
241	1	3	1	1	291,38
242	1	3	1	1	305,7
243	1	3	1	1	295,82
244	1	3	1	1	298,37
245	1	3	1	1	292,73
246	1	3	1	1	298,79
247	1	3	1	1	296,55
248	1	3	1	1	312,39
249	1	3	1	1	296,01
250	1	3	1	1	296,21
251	1	3	1	1	304,97
252	1	3	1	1	292,21
253	1	3	1	1	298,57
254	1	3	1	1	309,22
255	1	3	1	1	291,32
256	1	3	1	1	306,46
257	1	3	1	1	301,4
258	1	3	1	1	293,51
259	1	3	1	1	305,3
260	1	3	1	1	297,17
261	1	3	1	2	298,04
262	1	3	1	2	297,87
263	1	3	1	2	310,41
264	1	3	1	2	315,41
265	1	3	1	2	307,78
266	1	3	1	2	308,38
267	1	3	1	2	302,93
268	1	3	1	2	296,01
269	1	3	1	2	305,54
270	1	3	1	2	298,41
271	1	3	1	2	299,96
272	1	3	1	2	309,54
273	1	3	1	2	296,8

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
274	1	3	1	2	303,05
275	1	3	1	2	308,64
276	1	3	1	2	295,53
277	1	3	1	2	310,41
278	1	3	1	2	300,25
279	1	3	1	2	307,06
280	1	3	1	2	298,74
281	1	3	2	1	293,12
282	1	3	2	1	305,36
283	1	3	2	1	308,74
284	1	3	2	1	305,03
285	1	3	2	1	302,98
286	1	3	2	1	305,19
287	1	3	2	1	302,4
288	1	3	2	1	302,3
289	1	3	2	1	304,67
290	1	3	2	1	301,28
291	1	3	2	1	308,48
292	1	3	2	1	299,59
293	1	3	2	1	301,32
294	1	3	2	1	298,74
295	1	3	2	1	304,45
296	1	3	2	1	299,29
297	1	3	2	1	301,27
298	1	3	2	1	308,54
299	1	3	2	1	300,8
300	1	3	2	1	300,28
301	1	3	2	2	300,53
302	1	3	2	2	295,47
303	1	3	2	2	296,84
304	1	3	2	2	297,65
305	1	3	2	2	307,54
306	1	3	2	2	299,87
307	1	3	2	2	292,83
308	1	3	2	2	301,39
309	1	3	2	2	295,11
310	1	3	2	2	310,05
311	1	3	2	2	309,66
312	1	3	2	2	302,47
313	1	3	2	2	301,99
314	1	3	2	2	303,36

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
315	1	3	2	2	296,42
316	1	3	2	2	303,34
317	1	3	2	2	298,9
318	1	3	2	2	298,47
319	1	3	2	2	299,87
320	1	3	2	2	299,05
321	1	3	3	1	270,23
322	1	3	3	1	270,78
323	1	3	3	1	279,18
324	1	3	3	1	278,94
325	1	3	3	1	271,07
326	1	3	3	1	277,8
327	1	3	3	1	271,44
328	1	3	3	1	278,19
329	1	3	3	1	276,24
330	1	3	3	1	272,95
331	1	3	3	1	284,42
332	1	3	3	1	270,16
333	1	3	3	1	279,02
334	1	3	3	1	265,54
335	1	3	3	1	276,45
336	1	3	3	1	281,46
337	1	3	3	1	272,27
338	1	3	3	1	278,32
339	1	3	3	1	276,48
340	1	3	3	1	276,3
341	1	3	3	2	264,37
342	1	3	3	2	266,52
343	1	3	3	2	279,73
344	1	3	3	2	267,34
345	1	3	3	2	270,87
346	1	3	3	2	271,19
347	1	3	3	2	276,96
348	1	3	3	2	277,83
349	1	3	3	2	268,89
350	1	3	3	2	276,19
351	1	3	3	2	273,4
352	1	3	3	2	270,5
353	1	3	3	2	270,37
354	1	3	3	2	271,45
355	1	3	3	2	273,43

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
356	1	3	3	2	271,09
357	1	3	3	2	269,45
358	1	3	3	2	268,24
359	1	3	3	2	269,53
360	1	3	3	2	271,85
361	2	1	1	1	695,96
362	2	1	1	1	694,95
363	2	1	1	1	702,11
364	2	1	1	1	692,42
365	2	1	1	1	700,21
366	2	1	1	1	698,74
367	2	1	1	1	705,77
368	2	1	1	1	696,62
369	2	1	1	1	702,87
370	2	1	1	1	696,05
371	2	1	1	1	702,35
372	2	1	1	1	700,68
373	2	1	1	1	701,23
374	2	1	1	1	706,74
375	2	1	1	1	709,08
376	2	1	1	1	704,9
377	2	1	1	1	696,26
378	2	1	1	1	702,48
379	2	1	1	1	695,66
380	2	1	1	1	697,69
381	2	1	1	2	712,48
382	2	1	1	2	697,79
383	2	1	1	2	700,28
384	2	1	1	2	708,37
385	2	1	1	2	705,7
386	2	1	1	2	701,17
387	2	1	1	2	699,44
388	2	1	1	2	714,47
389	2	1	1	2	709,33
390	2	1	1	2	692,09
391	2	1	1	2	700,58
392	2	1	1	2	715,18
393	2	1	1	2	707,04
394	2	1	1	2	695,18
395	2	1	1	2	694,81
396	2	1	1	2	706,84

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
397	2	1	1	2	708,92
398	2	1	1	2	704,23
399	2	1	1	2	706,15
400	2	1	1	2	701,1
401	2	1	2	1	657,63
402	2	1	2	1	644,89
403	2	1	2	1	646,67
404	2	1	2	1	650,98
405	2	1	2	1	662,27
406	2	1	2	1	653,86
407	2	1	2	1	657,26
408	2	1	2	1	654,06
409	2	1	2	1	651,66
410	2	1	2	1	657,61
411	2	1	2	1	653,26
412	2	1	2	1	665,53
413	2	1	2	1	642,73
414	2	1	2	1	667,1
415	2	1	2	1	662,37
416	2	1	2	1	669,62
417	2	1	2	1	657,1
418	2	1	2	1	652,97
419	2	1	2	1	662,29
420	2	1	2	1	655,01
421	2	1	2	2	656,92
422	2	1	2	2	654,34
423	2	1	2	2	650,54
424	2	1	2	2	654,75
425	2	1	2	2	650,08
426	2	1	2	2	658,86
427	2	1	2	2	650,13
428	2	1	2	2	655,32
429	2	1	2	2	670,56
430	2	1	2	2	655,09
431	2	1	2	2	661,14
432	2	1	2	2	648,86
433	2	1	2	2	655,48
434	2	1	2	2	647,27
435	2	1	2	2	661,88
436	2	1	2	2	651,58
437	2	1	2	2	652,1

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
438	2	1	2	2	654,76
439	2	1	2	2	652,87
440	2	1	2	2	661,09
441	2	1	3	1	533,56
442	2	1	3	1	553,31
443	2	1	3	1	553,27
444	2	1	3	1	547,38
445	2	1	3	1	548,32
446	2	1	3	1	546,62
447	2	1	3	1	539,79
448	2	1	3	1	537,61
449	2	1	3	1	537,88
450	2	1	3	1	561,3
451	2	1	3	1	540,01
452	2	1	3	1	549,11
453	2	1	3	1	542,06
454	2	1	3	1	544,06
455	2	1	3	1	545,83
456	2	1	3	1	549,01
457	2	1	3	1	546,36
458	2	1	3	1	540,85
459	2	1	3	1	539,04
460	2	1	3	1	553,51
461	2	1	3	2	572,56
462	2	1	3	2	571,41
463	2	1	3	2	563,75
464	2	1	3	2	572,46
465	2	1	3	2	571,75
466	2	1	3	2	573,83
467	2	1	3	2	571,95
468	2	1	3	2	575,17
469	2	1	3	2	571,39
470	2	1	3	2	569,54
471	2	1	3	2	572,19
472	2	1	3	2	563,68
473	2	1	3	2	570,08
474	2	1	3	2	567,2
475	2	1	3	2	578,14
476	2	1	3	2	574,48
477	2	1	3	2	569,84
478	2	1	3	2	571,81

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
479	2	1	3	2	580,26
480	2	1	3	2	570,78
481	2	2	1	1	362,41
482	2	2	1	1	357,97
483	2	2	1	1	355,32
484	2	2	1	1	358,28
485	2	2	1	1	355,79
486	2	2	1	1	358,9
487	2	2	1	1	352,39
488	2	2	1	1	359,09
489	2	2	1	1	354,94
490	2	2	1	1	359,1
491	2	2	1	1	359,33
492	2	2	1	1	356,17
493	2	2	1	1	356,83
494	2	2	1	1	354,08
495	2	2	1	1	357,82
496	2	2	1	1	355,42
497	2	2	1	1	357,24
498	2	2	1	1	358,33
499	2	2	1	1	360,59
500	2	2	1	1	363,38
501	2	2	1	2	357,53
502	2	2	1	2	358,87
503	2	2	1	2	362,44
504	2	2	1	2	364,9
505	2	2	1	2	361,64
506	2	2	1	2	362,51
507	2	2	1	2	355,25
508	2	2	1	2	353,04
509	2	2	1	2	364,58
510	2	2	1	2	362,23
511	2	2	1	2	359,67
512	2	2	1	2	360,73
513	2	2	1	2	365,89
514	2	2	1	2	360,5
515	2	2	1	2	357,65
516	2	2	1	2	356,28
517	2	2	1	2	369,31
518	2	2	1	2	358,55
519	2	2	1	2	365,82

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
520	2	2	1	2	363,91
521	2	2	2	1	335,38
522	2	2	2	1	334,44
523	2	2	2	1	328,21
524	2	2	2	1	333,07
525	2	2	2	1	336,2
526	2	2	2	1	331,64
527	2	2	2	1	332,03
528	2	2	2	1	329,71
529	2	2	2	1	335,42
530	2	2	2	1	331,4
531	2	2	2	1	330,55
532	2	2	2	1	328,43
533	2	2	2	1	332,28
534	2	2	2	1	332,53
535	2	2	2	1	333,12
536	2	2	2	1	338,2
537	2	2	2	1	338,71
538	2	2	2	1	327,31
539	2	2	2	1	326,45
540	2	2	2	1	328,9
541	2	2	2	2	333,41
542	2	2	2	2	336,49
543	2	2	2	2	332,1
544	2	2	2	2	332,55
545	2	2	2	2	337,71
546	2	2	2	2	328,68
547	2	2	2	2	327,68
548	2	2	2	2	321,28
549	2	2	2	2	330,14
550	2	2	2	2	332,82
551	2	2	2	2	321,02
552	2	2	2	2	337,86
553	2	2	2	2	333,67
554	2	2	2	2	329,74
555	2	2	2	2	330,72
556	2	2	2	2	333,89
557	2	2	2	2	330,92
558	2	2	2	2	329,3
559	2	2	2	2	327,16
560	2	2	2	2	328,32

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
561	2	2	3	1	288,68
562	2	2	3	1	282,21
563	2	2	3	1	280,51
564	2	2	3	1	288,77
565	2	2	3	1	291,61
566	2	2	3	1	282,39
567	2	2	3	1	289,98
568	2	2	3	1	282,5
569	2	2	3	1	276,89
570	2	2	3	1	288,38
571	2	2	3	1	287,49
572	2	2	3	1	278,94
573	2	2	3	1	284,47
574	2	2	3	1	283,16
575	2	2	3	1	286,19
576	2	2	3	1	287,14
577	2	2	3	1	281,39
578	2	2	3	1	283,67
579	2	2	3	1	289,27
580	2	2	3	1	291,22
581	2	2	3	2	279,71
582	2	2	3	2	288,59
583	2	2	3	2	284,64
584	2	2	3	2	281,59
585	2	2	3	2	276,87
586	2	2	3	2	281,67
587	2	2	3	2	282,99
588	2	2	3	2	280,03
589	2	2	3	2	285,11
590	2	2	3	2	285,09
591	2	2	3	2	283,96
592	2	2	3	2	284,12
593	2	2	3	2	279,62
594	2	2	3	2	284,81
595	2	2	3	2	282,98
596	2	2	3	2	285,02
597	2	2	3	2	289,03
598	2	2	3	2	288,68
599	2	2	3	2	288,63
600	2	2	3	2	278,8
601	2	3	1	1	184,05

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
602	2	3	1	1	177,23
603	2	3	1	1	181,06
604	2	3	1	1	181,77
605	2	3	1	1	176,84
606	2	3	1	1	189,72
607	2	3	1	1	186,95
608	2	3	1	1	186,87
609	2	3	1	1	183,48
610	2	3	1	1	181,39
611	2	3	1	1	181,64
612	2	3	1	1	187,78
613	2	3	1	1	183,42
614	2	3	1	1	182,43
615	2	3	1	1	183,31
616	2	3	1	1	179,86
617	2	3	1	1	182,38
618	2	3	1	1	183,69
619	2	3	1	1	189,24
620	2	3	1	1	188,62
621	2	3	1	2	192,51
622	2	3	1	2	184,58
623	2	3	1	2	192,96
624	2	3	1	2	185,25
625	2	3	1	2	190,78
626	2	3	1	2	184,81
627	2	3	1	2	189,48
628	2	3	1	2	183,77
629	2	3	1	2	188,83
630	2	3	1	2	186,87
631	2	3	1	2	190,27
632	2	3	1	2	187,47
633	2	3	1	2	186,41
634	2	3	1	2	183,38
635	2	3	1	2	186,14
636	2	3	1	2	187,59
637	2	3	1	2	182,97
638	2	3	1	2	187,84
639	2	3	1	2	186,53
640	2	3	1	2	186,33
641	2	3	2	1	172,16
642	2	3	2	1	167,87

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
643	2	3	2	1	171,95
644	2	3	2	1	172,03
645	2	3	2	1	170,6
646	2	3	2	1	176,72
647	2	3	2	1	170,4
648	2	3	2	1	162,19
649	2	3	2	1	165,29
650	2	3	2	1	179,23
651	2	3	2	1	179,67
652	2	3	2	1	171,58
653	2	3	2	1	170,31
654	2	3	2	1	169,35
655	2	3	2	1	182,23
656	2	3	2	1	172,22
657	2	3	2	1	171,39
658	2	3	2	1	176,39
659	2	3	2	1	178,37
660	2	3	2	1	166,52
661	2	3	2	2	178,72
662	2	3	2	2	177,3
663	2	3	2	2	175,95
664	2	3	2	2	179,75
665	2	3	2	2	176,87
666	2	3	2	2	175,78
667	2	3	2	2	176,22
668	2	3	2	2	181,16
669	2	3	2	2	177,22
670	2	3	2	2	173,64
671	2	3	2	2	181,33
672	2	3	2	2	179,34
673	2	3	2	2	177,13
674	2	3	2	2	178,89
675	2	3	2	2	177,61
676	2	3	2	2	177,43
677	2	3	2	2	174,11
678	2	3	2	2	180,94
679	2	3	2	2	181,65
680	2	3	2	2	185,22
681	2	3	3	1	145,38
682	2	3	3	1	142,11
683	2	3	3	1	140,23

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
684	2	3	3	1	150,44
685	2	3	3	1	148,18
686	2	3	3	1	147,93
687	2	3	3	1	150,89
688	2	3	3	1	149,87
689	2	3	3	1	150,03
690	2	3	3	1	145,26
691	2	3	3	1	153,01
692	2	3	3	1	149,28
693	2	3	3	1	150,41
694	2	3	3	1	152,37
695	2	3	3	1	146,85
696	2	3	3	1	148,34
697	2	3	3	1	147,3
698	2	3	3	1	149,56
699	2	3	3	1	146,84
700	2	3	3	1	146,98
701	2	3	3	2	155,39
702	2	3	3	2	148,78
703	2	3	3	2	152,61
704	2	3	3	2	150,47
705	2	3	3	2	150,86
706	2	3	3	2	141,01
707	2	3	3	2	154,63
708	2	3	3	2	146,12
709	2	3	3	2	151,26
710	2	3	3	2	163,7
711	2	3	3	2	150,55
712	2	3	3	2	156,58
713	2	3	3	2	158,74
714	2	3	3	2	153,54
715	2	3	3	2	155,57
716	2	3	3	2	153,33
717	2	3	3	2	149,67
718	2	3	3	2	153,39
719	2	3	3	2	151,56
720	2	3	3	2	154,81
721	3	1	1	1	384,76
722	3	1	1	1	389,81
723	3	1	1	1	389,71
724	3	1	1	1	386,67

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
725	3	1	1	1	382,75
726	3	1	1	1	391,5
727	3	1	1	1	389,49
728	3	1	1	1	386,87
729	3	1	1	1	388,07
730	3	1	1	1	386,58
731	3	1	1	1	382,48
732	3	1	1	1	386,99
733	3	1	1	1	387,6
734	3	1	1	1	391,36
735	3	1	1	1	393,81
736	3	1	1	1	391,1
737	3	1	1	1	395,61
738	3	1	1	1	392,99
739	3	1	1	1	389,19
740	3	1	1	1	382,8
741	3	1	1	2	386,43
742	3	1	1	2	379,18
743	3	1	1	2	384,17
744	3	1	1	2	380,69
745	3	1	1	2	386,68
746	3	1	1	2	381,78
747	3	1	1	2	385,8
748	3	1	1	2	382,96
749	3	1	1	2	381,68
750	3	1	1	2	389,31
751	3	1	1	2	381,2
752	3	1	1	2	384,8
753	3	1	1	2	387,25
754	3	1	1	2	386,2
755	3	1	1	2	384,2
756	3	1	1	2	387,6
757	3	1	1	2	390,87
758	3	1	1	2	389,75
759	3	1	1	2	390,48
760	3	1	1	2	388,32
761	3	1	2	1	331,88
762	3	1	2	1	330,69
763	3	1	2	1	332,38
764	3	1	2	1	329,5
765	3	1	2	1	335,87

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
766	3	1	2	1	335,3
767	3	1	2	1	339,38
768	3	1	2	1	339,86
769	3	1	2	1	335,72
770	3	1	2	1	328,45
771	3	1	2	1	330,15
772	3	1	2	1	341,98
773	3	1	2	1	334,47
774	3	1	2	1	333,16
775	3	1	2	1	335,29
776	3	1	2	1	334,17
777	3	1	2	1	333,28
778	3	1	2	1	334,86
779	3	1	2	1	333,5
780	3	1	2	1	334,52
781	3	1	2	2	334,59
782	3	1	2	2	330,63
783	3	1	2	2	326,83
784	3	1	2	2	334,36
785	3	1	2	2	340,95
786	3	1	2	2	338,27
787	3	1	2	2	336,47
788	3	1	2	2	334,69
789	3	1	2	2	332,56
790	3	1	2	2	332,07
791	3	1	2	2	340,05
792	3	1	2	2	335,33
793	3	1	2	2	330,72
794	3	1	2	2	328,48
795	3	1	2	2	329,97
796	3	1	2	2	336,45
797	3	1	2	2	333,45
798	3	1	2	2	336,04
799	3	1	2	2	335,64
800	3	1	2	2	321,71
801	3	1	3	1	279,95
802	3	1	3	1	274,37
803	3	1	3	1	278,13
804	3	1	3	1	282,05
805	3	1	3	1	286,05
806	3	1	3	1	288,54

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
807	3	1	3	1	279,3
808	3	1	3	1	281,54
809	3	1	3	1	274,91
810	3	1	3	1	291,27
811	3	1	3	1	288,63
812	3	1	3	1	280,45
813	3	1	3	1	267,51
814	3	1	3	1	273,09
815	3	1	3	1	272,84
816	3	1	3	1	276,93
817	3	1	3	1	275,34
818	3	1	3	1	279,73
819	3	1	3	1	276,31
820	3	1	3	1	279,23
821	3	1	3	2	289,42
822	3	1	3	2	283,63
823	3	1	3	2	284,47
824	3	1	3	2	293,12
825	3	1	3	2	290,01
826	3	1	3	2	292,5
827	3	1	3	2	287,91
828	3	1	3	2	289,58
829	3	1	3	2	294,81
830	3	1	3	2	282,64
831	3	1	3	2	291,69
832	3	1	3	2	294,95
833	3	1	3	2	286,18
834	3	1	3	2	293,73
835	3	1	3	2	295,68
836	3	1	3	2	284,62
837	3	1	3	2	285,62
838	3	1	3	2	287,57
839	3	1	3	2	284,24
840	3	1	3	2	292,11
841	3	2	1	1	196,53
842	3	2	1	1	197,09
843	3	2	1	1	192,7
844	3	2	1	1	197,4
845	3	2	1	1	192,07
846	3	2	1	1	200,86
847	3	2	1	1	192,77

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
848	3	2	1	1	182,03
849	3	2	1	1	192,84
850	3	2	1	1	197,33
851	3	2	1	1	189,05
852	3	2	1	1	182,69
853	3	2	1	1	195,76
854	3	2	1	1	176,26
855	3	2	1	1	200,96
856	3	2	1	1	190,36
857	3	2	1	1	192,74
858	3	2	1	1	192,98
859	3	2	1	1	187,55
860	3	2	1	1	183,48
861	3	2	1	2	191,45
862	3	2	1	2	197,53
863	3	2	1	2	192,09
864	3	2	1	2	189,92
865	3	2	1	2	184,47
866	3	2	1	2	193,71
867	3	2	1	2	197,41
868	3	2	1	2	198,42
869	3	2	1	2	201,33
870	3	2	1	2	202,99
871	3	2	1	2	192,2
872	3	2	1	2	201,31
873	3	2	1	2	185,29
874	3	2	1	2	182,38
875	3	2	1	2	195,22
876	3	2	1	2	199,1
877	3	2	1	2	201,75
878	3	2	1	2	193,44
879	3	2	1	2	193,89
880	3	2	1	2	199,17
881	3	2	2	1	167,05
882	3	2	2	1	164,88
883	3	2	2	1	180,04
884	3	2	2	1	167,79
885	3	2	2	1	173,87
886	3	2	2	1	173,28
887	3	2	2	1	168,03
888	3	2	2	1	167,56

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
889	3	2	2	1	170,55
890	3	2	2	1	160,06
891	3	2	2	1	174,62
892	3	2	2	1	166,64
893	3	2	2	1	174,39
894	3	2	2	1	175,13
895	3	2	2	1	170,53
896	3	2	2	1	169,73
897	3	2	2	1	160,79
898	3	2	2	1	167,97
899	3	2	2	1	170,52
900	3	2	2	1	167,8
901	3	2	2	2	169,09
902	3	2	2	2	167,94
903	3	2	2	2	164,96
904	3	2	2	2	167,14
905	3	2	2	2	167,46
906	3	2	2	2	166,56
907	3	2	2	2	173,59
908	3	2	2	2	162,87
909	3	2	2	2	170,54
910	3	2	2	2	169,81
911	3	2	2	2	162,59
912	3	2	2	2	161,52
913	3	2	2	2	167,1
914	3	2	2	2	169,83
915	3	2	2	2	161,23
916	3	2	2	2	164,9
917	3	2	2	2	164,93
918	3	2	2	2	163,31
919	3	2	2	2	167,13
920	3	2	2	2	163,89
921	3	2	3	1	147,88
922	3	2	3	1	147,94
923	3	2	3	1	146,73
924	3	2	3	1	144,01
925	3	2	3	1	140,57
926	3	2	3	1	131,64
927	3	2	3	1	153,46
928	3	2	3	1	146,43
929	3	2	3	1	141,54

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
930	3	2	3	1	148,03
931	3	2	3	1	149,89
932	3	2	3	1	146,87
933	3	2	3	1	146,08
934	3	2	3	1	139,18
935	3	2	3	1	149,85
936	3	2	3	1	143,83
937	3	2	3	1	136,4
938	3	2	3	1	138,08
939	3	2	3	1	140,81
940	3	2	3	1	150,63
941	3	2	3	2	147,71
942	3	2	3	2	144,35
943	3	2	3	2	145,26
944	3	2	3	2	141,31
945	3	2	3	2	151,98
946	3	2	3	2	137,86
947	3	2	3	2	146,38
948	3	2	3	2	140,1
949	3	2	3	2	143,53
950	3	2	3	2	147,75
951	3	2	3	2	146,3
952	3	2	3	2	142,26
953	3	2	3	2	142,01
954	3	2	3	2	137,09
955	3	2	3	2	143,88
956	3	2	3	2	137,42
957	3	2	3	2	144,11
958	3	2	3	2	143,41
959	3	2	3	2	132,3
960	3	2	3	2	148
961	3	3	1	1	101,05
962	3	3	1	1	102,94
963	3	3	1	1	98,85
964	3	3	1	1	105,6
965	3	3	1	1	99,38
966	3	3	1	1	107,21
967	3	3	1	1	102,07
968	3	3	1	1	103,71
969	3	3	1	1	105,86
970	3	3	1	1	108,11

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
971	3	3	1	1	101,38
972	3	3	1	1	97,11
973	3	3	1	1	103,18
974	3	3	1	1	98,7
975	3	3	1	1	108,61
976	3	3	1	1	101,39
977	3	3	1	1	100,7
978	3	3	1	1	98,37
979	3	3	1	1	103,45
980	3	3	1	1	100,51
981	3	3	1	2	103,09
982	3	3	1	2	100,22
983	3	3	1	2	102,63
984	3	3	1	2	102,95
985	3	3	1	2	103,54
986	3	3	1	2	98,58
987	3	3	1	2	103,36
988	3	3	1	2	107,86
989	3	3	1	2	102,95
990	3	3	1	2	100,4
991	3	3	1	2	98,57
992	3	3	1	2	103,9
993	3	3	1	2	104,52
994	3	3	1	2	106,3
995	3	3	1	2	103,39
996	3	3	1	2	105,99
997	3	3	1	2	101,92
998	3	3	1	2	103,41
999	3	3	1	2	105,6
1000	3	3	1	2	105,93
1001	3	3	2	1	90,04
1002	3	3	2	1	84,92
1003	3	3	2	1	90,72
1004	3	3	2	1	87,05
1005	3	3	2	1	90,41
1006	3	3	2	1	88,39
1007	3	3	2	1	87,72
1008	3	3	2	1	89,94
1009	3	3	2	1	91,39
1010	3	3	2	1	86,39
1011	3	3	2	1	90,04

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
1012	3	3	2	1	85,24
1013	3	3	2	1	88,35
1014	3	3	2	1	88,85
1015	3	3	2	1	87,12
1016	3	3	2	1	87,15
1017	3	3	2	1	91
1018	3	3	2	1	88,65
1019	3	3	2	1	87,54
1020	3	3	2	1	89,51
1021	3	3	2	2	85,31
1022	3	3	2	2	92,85
1023	3	3	2	2	90,75
1024	3	3	2	2	97,15
1025	3	3	2	2	90,71
1026	3	3	2	2	84,2
1027	3	3	2	2	89,98
1028	3	3	2	2	93,47
1029	3	3	2	2	88,74
1030	3	3	2	2	93,76
1031	3	3	2	2	89,02
1032	3	3	2	2	89,25
1033	3	3	2	2	95,38
1034	3	3	2	2	87,7
1035	3	3	2	2	82,83
1036	3	3	2	2	89,15
1037	3	3	2	2	91,73
1038	3	3	2	2	92,99
1039	3	3	2	2	94,55
1040	3	3	2	2	92,42
1041	3	3	3	1	81,65
1042	3	3	3	1	74,64
1043	3	3	3	1	71,86
1044	3	3	3	1	70,87
1045	3	3	3	1	71,82
1046	3	3	3	1	73,51
1047	3	3	3	1	74,14
1048	3	3	3	1	73,01
1049	3	3	3	1	72,8
1050	3	3	3	1	76,09
1051	3	3	3	1	75,21
1052	3	3	3	1	71,07

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
1053	3	3	3	1	69,54
1054	3	3	3	1	72,99
1055	3	3	3	1	74,24
1056	3	3	3	1	71,61
1057	3	3	3	1	72,66
1058	3	3	3	1	76,1
1059	3	3	3	1	77,97
1060	3	3	3	1	73,52
1061	3	3	3	2	76,8
1062	3	3	3	2	76,83
1063	3	3	3	2	82,08
1064	3	3	3	2	79,65
1065	3	3	3	2	80,83
1066	3	3	3	2	76,94
1067	3	3	3	2	85,09
1068	3	3	3	2	76,3
1069	3	3	3	2	72,1
1070	3	3	3	2	75,47
1071	3	3	3	2	75,58
1072	3	3	3	2	72,52
1073	3	3	3	2	71,16
1074	3	3	3	2	74,65
1075	3	3	3	2	82,88
1076	3	3	3	2	75,53
1077	3	3	3	2	77,9
1078	3	3	3	2	78,2
1079	3	3	3	2	76,6
1080	3	3	3	2	80,79

Legenda Ambiente B			
Área	**Tempo de previsão**	**Processos**	**Cuda**
1 - Grande	1 - 8 h	1 – 30	1 – Sem
2 - Média	2 - 6h	2 – 15	2 - Com
3 - Pequena		3 - 9	

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
1	1	1	1	1	4232,96
2	1	1	1	1	4327,84
3	1	1	1	1	4370,59
4	1	1	1	1	4375,73
5	1	1	1	1	4327,12

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
6	1	1	1	1	4295,63
7	1	1	1	1	4378,21
8	1	1	1	1	4266,44
9	1	1	1	1	4361,30
10	1	1	1	1	4324,24
11	1	1	1	1	4300,44
12	1	1	1	1	4313,82
13	1	1	1	1	4306,51
14	1	1	1	1	4255,60
15	1	1	1	1	4297,60
16	1	1	1	1	4324,68
17	1	1	1	1	4380,86
18	1	1	1	1	4266,71
19	1	1	1	1	4365,29
20	1	1	1	1	4373,79
21	1	1	1	2	4194,78
22	1	1	1	2	4294,56
23	1	1	1	2	4271,84
24	1	1	1	2	4247,65
25	1	1	1	2	4173,49
26	1	1	1	2	4120,36
27	1	1	1	2	4184,19
28	1	1	1	2	4182,98
29	1	1	1	2	4221,57
30	1	1	1	2	4217,66
31	1	1	1	2	4184,86
32	1	1	1	2	4211,50
33	1	1	1	2	4158,96
34	1	1	1	2	4200,47
35	1	1	1	2	4217,19
36	1	1	1	2	4235,09
37	1	1	1	2	4242,30
38	1	1	1	2	4210,48
39	1	1	1	2	4196,45
40	1	1	1	2	4234,03
41	1	1	2	1	3806,30
42	1	1	2	1	3786,97
43	1	1	2	1	3855,37
44	1	1	2	1	3861,21
45	1	1	2	1	3761,26
46	1	1	2	1	3794,66

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
47	1	1	2	1	3790,67
48	1	1	2	1	3824,63
49	1	1	2	1	3831,99
50	1	1	2	1	3821,08
51	1	1	2	1	3815,34
52	1	1	2	1	3888,73
53	1	1	2	1	3789,55
54	1	1	2	1	3767,54
55	1	1	2	1	3735,17
56	1	1	2	1	3734,70
57	1	1	2	1	3847,49
58	1	1	2	1	3800,43
59	1	1	2	1	3834,39
60	1	1	2	1	3800,08
61	1	1	2	2	3700,92
62	1	1	2	2	3722,35
63	1	1	2	2	3721,19
64	1	1	2	2	3723,27
65	1	1	2	2	3690,83
66	1	1	2	2	3715,34
67	1	1	2	2	3594,01
68	1	1	2	2	3697,54
69	1	1	2	2	3749,03
70	1	1	2	2	3780,71
71	1	1	2	2	3701,78
72	1	1	2	2	3745,64
73	1	1	2	2	3772,80
74	1	1	2	2	3719,05
75	1	1	2	2	3699,80
76	1	1	2	2	3768,25
77	1	1	2	2	3738,51
78	1	1	2	2	3690,30
79	1	1	2	2	3769,84
80	1	1	2	2	3669,30
81	1	1	3	1	2635,45
82	1	1	3	1	2642,99
83	1	1	3	1	2662,76
84	1	1	3	1	2631,34
85	1	1	3	1	2681,25
86	1	1	3	1	2673,04
87	1	1	3	1	2645,56

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
88	1	1	3	1	2619,72
89	1	1	3	1	2657,73
90	1	1	3	1	2647,82
91	1	1	3	1	2627,92
92	1	1	3	1	2644,62
93	1	1	3	1	2633,82
94	1	1	3	1	2619,45
95	1	1	3	1	2667,95
96	1	1	3	1	2647,40
97	1	1	3	1	2634,95
98	1	1	3	1	2670,84
99	1	1	3	1	2616,99
100	1	1	3	1	2634,62
101	1	1	3	2	2618,84
102	1	1	3	2	2650,18
103	1	1	3	2	2669,37
104	1	1	3	2	2691,76
105	1	1	3	2	2617,91
106	1	1	3	2	2651,92
107	1	1	3	2	2735,94
108	1	1	3	2	2691,24
109	1	1	3	2	2695,10
110	1	1	3	2	2614,99
111	1	1	3	2	2659,96
112	1	1	3	2	2682,74
113	1	1	3	2	2651,97
114	1	1	3	2	2638,96
115	1	1	3	2	2637,73
116	1	1	3	2	2671,38
117	1	1	3	2	2572,71
118	1	1	3	2	2680,11
119	1	1	3	2	2639,35
120	1	1	3	2	2663,90
121	1	2	1	1	3251,61
122	1	2	1	1	3304,72
123	1	2	1	1	3358,74
124	1	2	1	1	3350,97
125	1	2	1	1	3318,01
126	1	2	1	1	3312,24
127	1	2	1	1	3349,72
128	1	2	1	1	3316,06

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
129	1	2	1	1	3329,44
130	1	2	1	1	3332,90
131	1	2	1	1	3359,67
132	1	2	1	1	3322,45
133	1	2	1	1	3360,14
134	1	2	1	1	3323,12
135	1	2	1	1	3324,18
136	1	2	1	1	3284,91
137	1	2	1	1	3327,07
138	1	2	1	1	3298,00
139	1	2	1	1	3285,40
140	1	2	1	1	3314,23
141	1	2	1	2	3184,09
142	1	2	1	2	3154,54
143	1	2	1	2	3259,68
144	1	2	1	2	3249,97
145	1	2	1	2	3246,19
146	1	2	1	2	3158,27
147	1	2	1	2	3147,31
148	1	2	1	2	3195,62
149	1	2	1	2	3223,17
150	1	2	1	2	3135,55
151	1	2	1	2	3181,66
152	1	2	1	2	3199,84
153	1	2	1	2	3220,05
154	1	2	1	2	3236,50
155	1	2	1	2	3230,67
156	1	2	1	2	3231,82
157	1	2	1	2	3185,63
158	1	2	1	2	3095,35
159	1	2	1	2	3197,93
160	1	2	1	2	3260,69
161	1	2	2	1	2771,43
162	1	2	2	1	2832,38
163	1	2	2	1	2845,04
164	1	2	2	1	2866,00
165	1	2	2	1	2803,85
166	1	2	2	1	2788,51
167	1	2	2	1	2831,06
168	1	2	2	1	2861,24
169	1	2	2	1	2835,28

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
170	1	2	2	1	2861,93
171	1	2	2	1	2823,07
172	1	2	2	1	2805,21
173	1	2	2	1	2830,94
174	1	2	2	1	2797,79
175	1	2	2	1	2848,74
176	1	2	2	1	2848,64
177	1	2	2	1	2857,72
178	1	2	2	1	2839,00
179	1	2	2	1	2832,25
180	1	2	2	1	2881,35
181	1	2	2	2	2808,11
182	1	2	2	2	2851,10
183	1	2	2	2	2811,97
184	1	2	2	2	2794,18
185	1	2	2	2	2740,24
186	1	2	2	2	2795,48
187	1	2	2	2	2805,43
188	1	2	2	2	2800,31
189	1	2	2	2	2791,36
190	1	2	2	2	2790,59
191	1	2	2	2	2772,02
192	1	2	2	2	2802,07
193	1	2	2	2	2812,80
194	1	2	2	2	2873,89
195	1	2	2	2	2801,31
196	1	2	2	2	2801,89
197	1	2	2	2	2786,31
198	1	2	2	2	2798,10
199	1	2	2	2	2780,09
200	1	2	2	2	2783,81
201	1	2	3	1	2651,35
202	1	2	3	1	2628,92
203	1	2	3	1	2626,48
204	1	2	3	1	2649,59
205	1	2	3	1	2637,19
206	1	2	3	1	2666,50
207	1	2	3	1	2679,93
208	1	2	3	1	2683,56
209	1	2	3	1	2661,09
210	1	2	3	1	2615,12

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
211	1	2	3	1	2617,48
212	1	2	3	1	2607,44
213	1	2	3	1	2612,38
214	1	2	3	1	2656,23
215	1	2	3	1	2666,25
216	1	2	3	1	2673,09
217	1	2	3	1	2648,15
218	1	2	3	1	2630,00
219	1	2	3	1	2612,14
220	1	2	3	1	2673,37
221	1	2	3	2	1962,03
222	1	2	3	2	1982,29
223	1	2	3	2	1966,99
224	1	2	3	2	1980,58
225	1	2	3	2	1987,26
226	1	2	3	2	2054,48
227	1	2	3	2	2002,81
228	1	2	3	2	2043,42
229	1	2	3	2	1959,66
230	1	2	3	2	1985,61
231	1	2	3	2	1990,89
232	1	2	3	2	2006,77
233	1	2	3	2	2028,92
234	1	2	3	2	2007,40
235	1	2	3	2	1978,56
236	1	2	3	2	1975,85
237	1	2	3	2	1973,70
238	1	2	3	2	2053,88
239	1	2	3	2	1963,52
240	1	2	3	2	1994,49
241	2	1	1	1	2613,09
242	2	1	1	1	2579,51
243	2	1	1	1	2582,34
244	2	1	1	1	2593,35
245	2	1	1	1	2565,93
246	2	1	1	1	2605,40
247	2	1	1	1	2627,64
248	2	1	1	1	2649,67
249	2	1	1	1	2518,16
250	2	1	1	1	2622,18
251	2	1	1	1	2581,34

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
252	2	1	1	1	2585,51
253	2	1	1	1	2515,00
254	2	1	1	1	2565,88
255	2	1	1	1	2550,18
256	2	1	1	1	2621,82
257	2	1	1	1	2629,78
258	2	1	1	1	2546,77
259	2	1	1	1	2600,27
260	2	1	1	1	2612,60
261	2	1	1	2	2563,23
262	2	1	1	2	2587,16
263	2	1	1	2	2597,23
264	2	1	1	2	2564,18
265	2	1	1	2	2566,72
266	2	1	1	2	2564,23
267	2	1	1	2	2579,02
268	2	1	1	2	2548,96
269	2	1	1	2	2582,47
270	2	1	1	2	2585,36
271	2	1	1	2	2607,25
272	2	1	1	2	2567,53
273	2	1	1	2	2591,00
274	2	1	1	2	2601,72
275	2	1	1	2	2578,92
276	2	1	1	2	2521,40
277	2	1	1	2	2598,01
278	2	1	1	2	2565,03
279	2	1	1	2	2569,82
280	2	1	1	2	2602,14
281	2	1	2	1	2216,21
282	2	1	2	1	2190,15
283	2	1	2	1	2156,80
284	2	1	2	1	2236,35
285	2	1	2	1	2170,10
286	2	1	2	1	2200,66
287	2	1	2	1	2160,01
288	2	1	2	1	2252,04
289	2	1	2	1	2141,60
290	2	1	2	1	2199,38
291	2	1	2	1	2194,40
292	2	1	2	1	2216,45

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
293	2	1	2	1	2245,47
294	2	1	2	1	2217,21
295	2	1	2	1	2179,46
296	2	1	2	1	2199,81
297	2	1	2	1	2236,20
298	2	1	2	1	2237,91
299	2	1	2	1	2222,48
300	2	1	2	1	2214,86
301	2	1	2	2	2419,99
302	2	1	2	2	2423,62
303	2	1	2	2	2317,10
304	2	1	2	2	2405,19
305	2	1	2	2	2379,54
306	2	1	2	2	2348,18
307	2	1	2	2	2356,26
308	2	1	2	2	2371,99
309	2	1	2	2	2443,35
310	2	1	2	2	2373,06
311	2	1	2	2	2393,86
312	2	1	2	2	2387,06
313	2	1	2	2	2401,73
314	2	1	2	2	2441,73
315	2	1	2	2	2431,91
316	2	1	2	2	2370,18
317	2	1	2	2	2401,53
318	2	1	2	2	2393,72
319	2	1	2	2	2392,13
320	2	1	2	2	2391,45
321	2	1	3	1	1441,67
322	2	1	3	1	1432,50
323	2	1	3	1	1419,90
324	2	1	3	1	1385,27
325	2	1	3	1	1433,01
326	2	1	3	1	1434,90
327	2	1	3	1	1419,98
328	2	1	3	1	1424,21
329	2	1	3	1	1410,24
330	2	1	3	1	1426,81
331	2	1	3	1	1404,74
332	2	1	3	1	1426,88
333	2	1	3	1	1416,16

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
334	2	1	3	1	1397,72
335	2	1	3	1	1405,31
336	2	1	3	1	1403,77
337	2	1	3	1	1392,01
338	2	1	3	1	1399,68
339	2	1	3	1	1428,17
340	2	1	3	1	1414,42
341	2	1	3	2	1514,72
342	2	1	3	2	1587,18
343	2	1	3	2	1579,47
344	2	1	3	2	1565,37
345	2	1	3	2	1579,95
346	2	1	3	2	1567,51
347	2	1	3	2	1602,75
348	2	1	3	2	1529,04
349	2	1	3	2	1571,03
350	2	1	3	2	1558,83
351	2	1	3	2	1565,33
352	2	1	3	2	1465,80
353	2	1	3	2	1530,04
354	2	1	3	2	1565,08
355	2	1	3	2	1531,66
356	2	1	3	2	1577,90
357	2	1	3	2	1589,35
358	2	1	3	2	1593,02
359	2	1	3	2	1601,99
360	2	1	3	2	1581,95
361	2	2	1	1	1951,78
362	2	2	1	1	1936,99
363	2	2	1	1	1930,42
364	2	2	1	1	1962,67
365	2	2	1	1	1926,47
366	2	2	1	1	1918,87
367	2	2	1	1	1901,32
368	2	2	1	1	1907,74
369	2	2	1	1	1906,51
370	2	2	1	1	1970,33
371	2	2	1	1	1982,04
372	2	2	1	1	1958,10
373	2	2	1	1	1938,19
374	2	2	1	1	1933,63

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
375	2	2	1	1	1922,20
376	2	2	1	1	1948,36
377	2	2	1	1	1942,28
378	2	2	1	1	1923,17
379	2	2	1	1	1933,07
380	2	2	1	1	1982,56
381	2	2	1	2	1952,62
382	2	2	1	2	1950,16
383	2	2	1	2	1911,40
384	2	2	1	2	1965,68
385	2	2	1	2	1955,32
386	2	2	1	2	1935,96
387	2	2	1	2	1887,77
388	2	2	1	2	1923,49
389	2	2	1	2	1957,60
390	2	2	1	2	1924,63
391	2	2	1	2	1858,38
392	2	2	1	2	1928,06
393	2	2	1	2	1969,06
394	2	2	1	2	1958,73
395	2	2	1	2	1954,46
396	2	2	1	2	1896,19
397	2	2	1	2	1936,12
398	2	2	1	2	1936,73
399	2	2	1	2	1948,68
400	2	2	1	2	1877,86
401	2	2	2	1	1669,88
402	2	2	2	1	1648,52
403	2	2	2	1	1636,04
404	2	2	2	1	1654,58
405	2	2	2	1	1666,60
406	2	2	2	1	1655,43
407	2	2	2	1	1641,81
408	2	2	2	1	1640,49
409	2	2	2	1	1611,14
410	2	2	2	1	1630,45
411	2	2	2	1	1637,57
412	2	2	2	1	1674,57
413	2	2	2	1	1597,36
414	2	2	2	1	1664,34
415	2	2	2	1	1629,45

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
416	2	2	2	1	1640,29
417	2	2	2	1	1638,00
418	2	2	2	1	1673,32
419	2	2	2	1	1639,01
420	2	2	2	1	1672,35
421	2	2	2	2	1644,51
422	2	2	2	2	1647,50
423	2	2	2	2	1620,76
424	2	2	2	2	1627,60
425	2	2	2	2	1650,42
426	2	2	2	2	1615,19
427	2	2	2	2	1663,33
428	2	2	2	2	1636,16
429	2	2	2	2	1625,51
430	2	2	2	2	1631,96
431	2	2	2	2	1657,75
432	2	2	2	2	1686,01
433	2	2	2	2	1678,79
434	2	2	2	2	1715,25
435	2	2	2	2	1642,17
436	2	2	2	2	1640,84
437	2	2	2	2	1637,57
438	2	2	2	2	1652,23
439	2	2	2	2	1671,14
440	2	2	2	2	1660,99
441	2	2	3	1	1093,43
442	2	2	3	1	1071,38
443	2	2	3	1	1035,55
444	2	2	3	1	1055,86
445	2	2	3	1	1033,99
446	2	2	3	1	1071,19
447	2	2	3	1	1048,30
448	2	2	3	1	1082,12
449	2	2	3	1	1052,13
450	2	2	3	1	1083,80
451	2	2	3	1	1058,23
452	2	2	3	1	1068,11
453	2	2	3	1	1056,77
454	2	2	3	1	1092,80
455	2	2	3	1	1064,45
456	2	2	3	1	1063,58

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
457	2	2	3	1	1055,27
458	2	2	3	1	1087,12
459	2	2	3	1	1057,75
460	2	2	3	1	1071,63
461	2	2	3	2	1060,20
462	2	2	3	2	1098,53
463	2	2	3	2	1087,59
464	2	2	3	2	1079,24
465	2	2	3	2	1059,60
466	2	2	3	2	1058,66
467	2	2	3	2	1095,05
468	2	2	3	2	1051,61
469	2	2	3	2	1078,57
470	2	2	3	2	1076,51
471	2	2	3	2	1072,25
472	2	2	3	2	1044,82
473	2	2	3	2	1073,53
474	2	2	3	2	1072,52
475	2	2	3	2	1036,18
476	2	2	3	2	1082,65
477	2	2	3	2	1069,70
478	2	2	3	2	1081,09
479	2	2	3	2	1043,96
480	2	2	3	2	1079,82
481	3	1	1	1	1411,70
482	3	1	1	1	1422,53
483	3	1	1	1	1407,41
484	3	1	1	1	1439,34
485	3	1	1	1	1434,39
486	3	1	1	1	1426,02
487	3	1	1	1	1437,79
488	3	1	1	1	1414,84
489	3	1	1	1	1420,84
490	3	1	1	1	1419,41
491	3	1	1	1	1418,30
492	3	1	1	1	1427,11
493	3	1	1	1	1442,66
494	3	1	1	1	1451,07
495	3	1	1	1	1411,13
496	3	1	1	1	1451,75
497	3	1	1	1	1422,46

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
498	3	1	1	1	1442,59
499	3	1	1	1	1411,19
500	3	1	1	1	1438,36
501	3	1	1	2	1325,85
502	3	1	1	2	1334,28
503	3	1	1	2	1349,22
504	3	1	1	2	1328,36
505	3	1	1	2	1323,39
506	3	1	1	2	1326,92
507	3	1	1	2	1332,29
508	3	1	1	2	1338,36
509	3	1	1	2	1334,33
510	3	1	1	2	1375,39
511	3	1	1	2	1320,38
512	3	1	1	2	1315,06
513	3	1	1	2	1314,26
514	3	1	1	2	1320,66
515	3	1	1	2	1327,09
516	3	1	1	2	1327,62
517	3	1	1	2	1361,69
518	3	1	1	2	1329,92
519	3	1	1	2	1317,01
520	3	1	1	2	1321,29
521	3	1	2	1	1004,26
522	3	1	2	1	993,09
523	3	1	2	1	998,42
524	3	1	2	1	1000,21
525	3	1	2	1	1010,55
526	3	1	2	1	1000,72
527	3	1	2	1	964,69
528	3	1	2	1	1011,95
529	3	1	2	1	982,64
530	3	1	2	1	1021,17
531	3	1	2	1	1010,13
532	3	1	2	1	996,01
533	3	1	2	1	984,97
534	3	1	2	1	1018,61
535	3	1	2	1	997,15
536	3	1	2	1	977,16
537	3	1	2	1	989,29
538	3	1	2	1	1014,21

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
539	3	1	2	1	1021,09
540	3	1	2	1	1004,15
541	3	1	2	2	1037,75
542	3	1	2	2	984,50
543	3	1	2	2	1012,94
544	3	1	2	2	1037,08
545	3	1	2	2	982,91
546	3	1	2	2	1000,28
547	3	1	2	2	985,31
548	3	1	2	2	989,83
549	3	1	2	2	980,11
550	3	1	2	2	988,49
551	3	1	2	2	1036,44
552	3	1	2	2	987,27
553	3	1	2	2	1020,13
554	3	1	2	2	1009,23
555	3	1	2	2	1018,11
556	3	1	2	2	1007,94
557	3	1	2	2	983,98
558	3	1	2	2	1009,73
559	3	1	2	2	1000,00
560	3	1	2	2	1011,34
561	3	1	3	1	579,19
562	3	1	3	1	583,16
563	3	1	3	1	569,62
564	3	1	3	1	565,95
565	3	1	3	1	584,43
566	3	1	3	1	580,85
567	3	1	3	1	570,54
568	3	1	3	1	572,72
569	3	1	3	1	567,92
570	3	1	3	1	572,63
571	3	1	3	1	580,33
572	3	1	3	1	586,27
573	3	1	3	1	580,02
574	3	1	3	1	580,29
575	3	1	3	1	574,94
576	3	1	3	1	578,15
577	3	1	3	1	581,63
578	3	1	3	1	576,58
579	3	1	3	1	577,60

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
580	3	1	3	1	583,22
581	3	1	3	2	573,40
582	3	1	3	2	588,48
583	3	1	3	2	576,88
584	3	1	3	2	576,60
585	3	1	3	2	571,94
586	3	1	3	2	590,76
587	3	1	3	2	588,90
588	3	1	3	2	580,57
589	3	1	3	2	592,16
590	3	1	3	2	579,97
591	3	1	3	2	588,53
592	3	1	3	2	580,38
593	3	1	3	2	590,90
594	3	1	3	2	583,48
595	3	1	3	2	590,77
596	3	1	3	2	597,83
597	3	1	3	2	564,14
598	3	1	3	2	579,91
599	3	1	3	2	573,26
600	3	1	3	2	578,17
601	3	2	1	1	989,52
602	3	2	1	1	984,04
603	3	2	1	1	980,95
604	3	2	1	1	991,47
605	3	2	1	1	980,06
606	3	2	1	1	973,01
607	3	2	1	1	1009,06
608	3	2	1	1	989,34
609	3	2	1	1	976,55
610	3	2	1	1	986,25
611	3	2	1	1	1001,36
612	3	2	1	1	984,65
613	3	2	1	1	1005,23
614	3	2	1	1	1010,34
615	3	2	1	1	981,97
616	3	2	1	1	1001,84
617	3	2	1	1	1012,43
618	3	2	1	1	989,11
619	3	2	1	1	972,00
620	3	2	1	1	979,31

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
621	3	2	1	2	1003,17
622	3	2	1	2	972,30
623	3	2	1	2	992,31
624	3	2	1	2	968,89
625	3	2	1	2	1019,78
626	3	2	1	2	993,46
627	3	2	1	2	971,46
628	3	2	1	2	969,28
629	3	2	1	2	1008,46
630	3	2	1	2	990,06
631	3	2	1	2	961,76
632	3	2	1	2	1001,43
633	3	2	1	2	989,95
634	3	2	1	2	977,38
635	3	2	1	2	965,71
636	3	2	1	2	994,84
637	3	2	1	2	989,98
638	3	2	1	2	997,48
639	3	2	1	2	1002,69
640	3	2	1	2	941,77
641	3	2	2	1	750,29
642	3	2	2	1	766,84
643	3	2	2	1	760,28
644	3	2	2	1	728,39
645	3	2	2	1	758,30
646	3	2	2	1	766,07
647	3	2	2	1	740,05
648	3	2	2	1	753,67
649	3	2	2	1	737,56
650	3	2	2	1	774,54
651	3	2	2	1	755,07
652	3	2	2	1	777,04
653	3	2	2	1	751,75
654	3	2	2	1	760,52
655	3	2	2	1	740,45
656	3	2	2	1	751,21
657	3	2	2	1	789,21
658	3	2	2	1	777,19
659	3	2	2	1	751,44
660	3	2	2	1	750,45
661	3	2	2	2	761,38

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
662	3	2	2	2	747,14
663	3	2	2	2	746,15
664	3	2	2	2	748,91
665	3	2	2	2	746,30
666	3	2	2	2	749,78
667	3	2	2	2	767,38
668	3	2	2	2	764,06
669	3	2	2	2	749,52
670	3	2	2	2	756,23
671	3	2	2	2	759,48
672	3	2	2	2	752,75
673	3	2	2	2	754,46
674	3	2	2	2	770,48
675	3	2	2	2	745,91
676	3	2	2	2	751,52
677	3	2	2	2	744,91
678	3	2	2	2	750,04
679	3	2	2	2	751,74
680	3	2	2	2	742,54
681	3	2	3	1	450,82
682	3	2	3	1	439,19
683	3	2	3	1	433,60
684	3	2	3	1	437,34
685	3	2	3	1	428,15
686	3	2	3	1	450,35
687	3	2	3	1	433,17
688	3	2	3	1	439,69
689	3	2	3	1	442,98
690	3	2	3	1	447,31
691	3	2	3	1	420,26
692	3	2	3	1	432,50
693	3	2	3	1	429,83
694	3	2	3	1	435,42
695	3	2	3	1	440,73
696	3	2	3	1	442,17
697	3	2	3	1	428,90
698	3	2	3	1	444,32
699	3	2	3	1	433,76
700	3	2	3	1	441,14
701	3	2	3	2	433,58
702	3	2	3	2	438,38

Ordem	Área	Tempo de previsão	Processos	Cuda	Tempo de execução
703	3	2	3	2	435,76
704	3	2	3	2	443,86
705	3	2	3	2	438,36
706	3	2	3	2	428,05
707	3	2	3	2	441,37
708	3	2	3	2	428,03
709	3	2	3	2	426,31
710	3	2	3	2	419,50
711	3	2	3	2	445,33
712	3	2	3	2	441,99
713	3	2	3	2	437,89
714	3	2	3	2	433,97
715	3	2	3	2	431,52
716	3	2	3	2	434,79
717	3	2	3	2	437,55
718	3	2	3	2	431,40
719	3	2	3	2	435,82
720	3	2	3	2	445,54

APÊNDICE B - CÓDIGOS ORIGINAIS E MODIFICADOS DO MODELO ETA

Código Original A – Hyper

```
916      DO 450 J=MYJS2,MYJE2
917          DO 450 I=MYIS,MYIE
918              TQ2B(I,J)=Q2ST(I,J,1)*ETADT(I,J,1)*F4Q2(1)
919      450 END DO
```

Kernel A CUDA Criado – Hyper

```
 3 attributes(global) subroutine hyper(d_a,d_b, d_c, d_d, MYIS, MYIE, MYJS2, MYJE2)
 4     implicit none
 5     integer, value :: MYIS,MYIE,MYJS2,MYJE2
 6     real,device :: d_a(:,:), d_b(:,:,:), d_c(:,:,:), d_d(:)
 7     integer :: i,j
 8
 9     i = ((blockIdx%x - 1) * blockDim%x) + threadIdx%x
10     j = ((blockIdx%y - 1) * blockDim%y) + threadIdx%y
11
12     if(i<1 .OR. i>(MYIE-MYIS +1) .OR. j<1 .OR. J>(MYJE2-MYJS2+1)) return
13     d_a(i,j) = d_b(i,j,1) * d_c(i,j,1) * d_d(1)
14     end subroutine hyper
```

Chamada do Kernel A – Hyper

```
923     ALLOCATE(d_a(MYIS:MYIE,MYJS2:MYJE2))
924     ALLOCATE(d_b(MYIS:MYIE,MYJS2:MYJE2,1))
925     ALLOCATE(d_c(MYIS:MYIE,MYJS2:MYJE2,1))
926     ALLOCATE(d_d(1))
927     d_a(MYIS:MYIE,MYJS2:MYJE2) = TQ2B(MYIS:MYIE,MYJS2:MYJE2)
928     d_b(:,:,1) = Q2ST(MYIS:MYIE,MYJS2:MYJE2,1)
929     d_c(:,:,1) = ETADT(MYIS:MYIE,MYJS2:MYJE2,1)
930     d_d(1) = F4Q2(1)
931
932     call hyper<<<grid, tBlock>>>(d_a, d_b, d_c, d_d, MYIS, MYIE, MYJS2, MYJE2)
933
934     TQ2B(MYIS:MYIE,MYJS2:MYJE2) = d_a(MYIS:MYIE,MYJS2:MYJE2)
935
936     DEALLOCATE (d_a)
937     DEALLOCATE (d_b)
938     DEALLOCATE (d_c)
939     DEALLOCATE (d_d)
```

Código Original B – Hyper 2

```
1125     DO 1200 L=1,LM1
1126         DO 1200 J=MYJS2,MYJE2
1127             DO 1200 I=MYIS,MYIE
1128                 SAMU(I,J,L)=SAMU(I,J,L)*VTM(I,J,L+1)
1129                 SAMV(I,J,L)=SAMV(I,J,L)*VTM(I,J,L+1)
1130     1200 END DO
```

Kernel B CUDA Criado – Hyper2

```
16 attributes(global) subroutine hyper2(d_b,d_c,d_x,MYIS, MYIE, MYJS2, MYJE2,LM1)
17     implicit none
18     integer, value :: MYIS, MYIE, MYJS2, MYJE2,LM1
19     real,device :: d_b(:,:,:), d_c(:,:,:),d_x(:,:,:)
20     integer :: i,j,k
21
22     i = ((blockIdx%x - 1) * blockDim%x) + threadIdx%x
23     j = ((blockIdx%y - 1) * blockDim%y) + threadIdx%y
24     k = ((blockIdx%z - 1) * blockDim%z) + threadIdx%z
25     !d_a(i,j) = (i* -1000) -j
26     if(i<1 .OR. i>(MYIE-MYIS +1) .OR. j<1 .OR. j>(MYJE2-MYJS2 +1) .OR. K<1 .OR. K>LM1) return
27     d_c(i,j,k) =d_c(i,j,k) * d_x(i,j,k +1)
28     d_b(i,j,k) =d_b(i,j,k) * d_x(i,j,k+1)
29     end subroutine hyper2
```

Chamada do Kernel B – Hyper2

```
1134     ALLOCATE(d_b(MYIS:MYIE,MYJS2:MYJE2,1:LM1))
1135     ALLOCATE(d_c(MYIS:MYIE,MYJS2:MYJE2,1:LM1))
1136     ALLOCATE(d_x(MYIS:MYIE,MYJS2:MYJE2,1:(LM1+1)))
1137     d_b(:,:,:) = SAMU(MYIS:MYIE,MYJS2:MYJE2,1:LM1)
1138     d_c(:,:,:) = SAMV(MYIS:MYIE,MYJS2:MYJE2,1:LM1)
1139     d_x(:,:,:) = VTM(MYIS:MYIE,MYJS2:MYJE2,1:(LM1+1))
1140
1141     call hyper2<<<grid2, tBlock2>>>(d_b,d_c,d_x,MYIS, MYIE, MYJS2, MYJE2,LM1)
1142
1143     SAMU(MYIS:MYIE,MYJS2:MYJE2,1:LM1) = d_b(MYIS:MYIE,MYJS2:MYJE2,1:LM1)
1144     SAMV(MYIS:MYIE,MYJS2:MYJE2,1:LM1) = d_c(MYIS:MYIE,MYJS2:MYJE2,1:LM1)
1145
1146     DEALLOCATE(d_b)
1147     DEALLOCATE(d_c)
1148     DEALLOCATE(d_x)
```

Código Original C – Hyper3

```
567          DO 230 L=2,LM1
568              DO 230 J=MYJS2,MYJE2
569                  DO 230 I=MYIS,MYIE
570                      ARRAY1(I,J,L)=WFA(L-1)*(1.-SAM(I,J,L-1))+WFB(L-1)
571                      ARRAY2(I,J,L)=WFA(L)+WFB(L)*(1.-SAM(I,J,L+1))
572          230 END DO
```

Kernel C CUDA Criado – Hyper3

```
31 attributes(global) subroutine hyper3(d_array1, d_array2, d_wfa, d_sam, d_wfb, MYIS, MYIE, MYJS2, MYJE2,LM1)
32     implicit none
33     integer, value :: MYIS, MYIE, MYJS2, MYJE2,LM1
34     real,device :: d_array1(:,:,:), d_array2(:,:,:),d_sam(:,:,:), d_wfa(:), d_wfb(:)
35     integer :: i,j,k
36
37     i = ((blockIdx%x - 1) * blockDim%x) + threadIdx%x
38     j = ((blockIdx%y - 1) * blockDim%y) + threadIdx%y
39     k = ((blockIdx%z - 1) * blockDim%z) + threadIdx%z
40     if(i<1 .OR. i>(MYIE-MYIS +1) .OR. j<1 .OR. j>(MYJE2-MYJS2 +1) .OR. K<2 .OR. K>LM1) return
41
42         d_array1(i,j,k) = d_wfa(k-1)*(1.-d_sam(i,j,k-1)) + d_wfb(k-1)
43         d_array2(i,j,k) = d_wfa(k)+d_wfb(k)*(1.-d_sam(i,j,k+1))
```

Chamada do Kernel C – Hyper3

```
542         call cpu_time(comeco)
543
544         Allocate(d_wfa(1:LM1))
545         allocate(d_sam(MYIS:MYIE,MYJS2:MYJE2,1:(LM1+1)))
546         allocate(d_wfb(1:LM1))
547         allocate(d_array1(MYIS:MYIE,MYJS2:MYJE2,1:(LM1+1)))
548         allocate(d_array2(MYIS:MYIE,MYJS2:MYJE2,1:(LM1+1)))
549
550         d_wfa(:) = WFA(1:(LM1+1))
551         d_sam(:,:,:) = SAM(MYIS:MYIE,MYJS2:MYJE2,1:(LM1+1))
552         d_wfb(:) = WFB(1:(LM1+1))
553
554         call hyper3<<<grid2, tBlock2>>>(d_array1, d_array2, d_wfa, d_sam, d_wfb, MYIS, MYIE, MYJS2, MYJE2, LM1)
555
556         ARRAY1(MYIS:MYIE,MYJS2:MYJE2,2:LM1)=d_array1(MYIS:MYIE,MYJS2:MYJE2,2:LM1)
557         ARRAY2(MYIS:MYIE,MYJS2:MYJE2,2:LM1)=d_array2(MYIS:MYIE,MYJS2:MYJE2,2:LM1)
558
559         DEALLOCATE (d_wfa)
560         DEALLOCATE (d_sam)
561         DEALLOCATE (d_wfb)
562         DEALLOCATE (d_array1)
563         DEALLOCATE (d_array2)
```