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

sid.inpe.br/mtc-m21c/2019/03.25.12.58-TDI

PARALELIZAÇÃO E OTIMIZAÇÃO DO DESEMPENHO COMPUTACIONAL DO MODELO BRASIL-SR

Jefferson Gonçalves de Souza

Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Celso Luiz Mendes, e Rodrigo Santos Costa, aprovada em 26 de fevereiro de 2019.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34R/3T295ES>

INPE
São José dos Campos
2019

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GBDIR)
Serviço de Informação e Documentação (SESID)
CEP 12.227-010
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/7348
E-mail: pubtc@inpe.br

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

sid.inpe.br/mtc-m21c/2019/03.25.12.58-TDI

PARALELIZAÇÃO E OTIMIZAÇÃO DO DESEMPENHO COMPUTACIONAL DO MODELO BRASIL-SR

Jefferson Gonçalves de Souza

Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Celso Luiz Mendes, e Rodrigo Santos Costa, aprovada em 26 de fevereiro de 2019.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34R/3T295ES>

INPE
São José dos Campos
2019

Dados Internacionais de Catalogação na Publicação (CIP)

Souza, Jefferson Gonçalves de.

So89p Paralelização e otimização do desempenho computacional do modelo BRASIL-SR / Jefferson Gonçalves de Souza. – São José dos Campos : INPE, 2019.

xxvi + 101 p. ; (sid.inpe.br/mtc-m21c/2019/03.25.12.58-TDI)

Dissertação (Mestrado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2019.

Orientadores : Drs. Celso Luiz Mendes, e Rodrigo Santos Costa.

1. Paralelização. 2. OpenMP. 3. Radiação solar. I.Título.

CDU 681.324

Aluno (a): *Jefferson Gonçalves de Souza*

Título: "PARALELIZAÇÃO E OTIMIZAÇÃO DO DESEMPENHO COMPUTACIONAL DO MODELO BRASIL-SR"

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de *Mestre* em *Computação Aplicada*

Dr. Stephan Stephany

Presidente / INPE / SJCampos - SP

( ) Participação por Video - Conferência

(X) Aprovado *( ) Reprovado*

Dr. Celso Luiz Mendes

Orientador(a) / INPE / São José dos Campos - SP

( ) Participação por Video - Conferência

(X) Aprovado *( ) Reprovado*

Dr. Rodrigo Santos Costa

Orientador(a) / INPE / São José dos Campos - SP

( ) Participação por Video - Conferência

(X) Aprovado *( ) Reprovado*

Dr. Reinaldo Roberto Rosa

Membro da Banca / INPE / SJCampos - SP

( ) Participação por Video - Conferência

(X) Aprovado *( ) Reprovado*

Dr. Fernando Ramos Martins

Convidado(a) / UNIFESP / Santos - SP

( ) Participação por Video - Conferência

(X) Aprovado *( ) Reprovado*

Este trabalho foi aprovado por:

( ) maioria simples

(X) unanimidade

São José dos Campos, 26 de fevereiro de 2019

“Uma mente que se abre a uma nova ideia jamais voltará ao seu tamanho original”.

Albert Einstein

AGRADECIMENTOS

Gostaria de agradecer, primeiramente, a meus pais, Tarciso e Celeste, e à minha irmã, Cinthya e minha sobrinha (afilhada) Alicía, pelo amor incondicional e apoio constante, pela compreensão de minha ausência e, principalmente, pelos esforços empreendidos com o objetivo de me ajudar a superar as dificuldades encontradas durante o Mestrado.

Agradeço sinceramente ao Dr. Celso Luiz Mendes e ao Dr. Rodrigo Santos Costa meus orientadores, por todo apoio e orientação necessária para o desenvolvimento desta dissertação de Mestrado.

Agradeço também o apoio dos colegas do Laboratório de Modelagem e Estudos de Recursos Renováveis de Energia (LABREN), em especial ao Francisco, Marcelo, André, Eduardo, Lucía Chipponelli, Maria Francisca, Silvia, Guilherme, Fabiana, Madeleine, Simone e Enio, pela companhia diária e pela ajuda no desenvolvimento deste estudo.

A minha namorada Thais, que esteve do meu lado em todos os momentos, me apoiando e proporcionando bons momentos.

Ao Instituto Nacional de Pesquisas Espaciais (INPE), ao Laboratório Associado de Computação e Matemática Aplicada (LAC) e a todos os professores, colaboradores e funcionários que fizeram parte dessa jornada.

Ao Laboratório Nacional de Computação Científica (LNCC/MCTI), por fornecer recursos de HPC do supercomputador SDumont, que contribuíram para os resultados desta pesquisa.

À Coordenação de Aperfeiçoamento Pessoal de Nível Superior (CAPES), pela oportunidade de estudo e disponibilização de suporte financeiro.

Dedico esse trabalho aos meus avós Juventino e Clicie, e José Francisco e Adel (*in memorian*), com todo amor e gratidão a vocês que foram exemplo de caráter e dignidade.

RESUMO

O modelo de transferência radiativa BRASIL-SR foi desenvolvido pelo LABREN/CCST/INPE e utiliza imagens de satélite em conjunto com dados climatológicos médios mensais para o cálculo de estimativas de irradiação solar incidente, permitindo mapear os locais propícios para o aproveitamento do recurso solar. As estimativas fornecidas pelo modelo têm foco nos resultados mensais e ainda na transmitância atmosférica calculada apenas para médias horárias no centro do intervalo mensal, devido ao alto custo computacional. Apesar de exaustivamente validado e de apresentar erros na mesma ordem de outras ferramentas computacionais da literatura, as extrapolações realizadas implicam em maiores valores de viés nas estimativas de irradiação horaria e diária. Este trabalho teve por objetivo otimizar o desempenho computacional do modelo BRASIL-SR em sistemas com múltiplos núcleos, através da utilização de diretivas de paralelização do OpenMP com escalonamento dos laços e alteração nos dados de entrada e saída para o formato em NetCDF. O tempo de processamento do modelo BRASIL-SR foi reduzido de 27 horas para 1 hora e 30 minutos, utilizando 24 *threads*. Outras técnicas tradicionais de otimização, como blocagem e vetorização, também foram investigadas, mas não apresentaram uma melhoria no desempenho devido à estrutura do código do modelo. Com o resultado inicial desta pesquisa, foi possível alterar o modo de calculo da transmitância, que antes era calculado para o centro do intervalo mensal, para o calculo diário da transmitância. Um script foi desenvolvimento para a execução do pré-processamento e processamento do modelo, obtendo um tempo total de pré-processamento e processamento de 1 hora e 40 minutos, uma melhora de 3700%. A utilização do modelo paralelizado com OpenMP, juntamente com os *scripts* Python, desenvolvidos para a operacionalização do modelo, será determinante para o desenvolvimento de modelos de previsão de irradiação solar de curto e curtíssimo prazo, sendo de extrema relevancia para tomada de decisão de diversos atores, como por exemplo o Operador Nacional do Sistema Elétrico.

Palavras-chave: Paralelização. OpenMP. Radiação Solar.

PARALLELIZATION AND OPTIMIZATION OF THE COMPUTATIONAL PERFORMANCE OF THE MODEL BRASIL-SR

ABSTRACT

The radiative transfer model BRASIL-SR was developed by LABREN/CCST/INPE. This model uses satellite images data with monthly average climatological for the estimates of incident solar irradiation, allowing mapping of the sites suitable for the use of the solar resource. The estimates provided by the model focus on the monthly results and on the atmospheric transmittance calculated only for hourly means in the center of the monthly range, due to the high computational cost. In spite of being extensively validated and presenting errors in the same order as other computational tools in the literature, the extrapolations carried out imply higher bias values in the hourly and daily irradiation estimates. This work aimed to optimize the computational performance of the BRASIL-SR model in systems with multi-core, using parallelization policies OpenMP with escalation of the loops and change in the input and output data to the format in NetCDF. The processing time of the model BRASIL-SR was reduced from 27 hours to 1 hour and 30 minutes using 24 threads. Other traditional optimization techniques, such as blocking and vectoring, were also investigated but did not show an improvement in performance due to the model code structure. With the initial result of this research, it was possible to change the mode of transmittance calculation, which was previously calculated to the center of the monthly interval, for the daily transmittance calculation. A script was development for the execution of the pre-processing and processing of the model, obtaining a total time of pre-processing and processing of 1 hour and 40 minutes, an improvement of 3700%. The use of the parallelized model with OpenMP, together with the Python scripts, developed for the operation of the model, will be determinant for the development of models of prediction of solar irradiation of short and very short term, being of extreme relevance for the decision making of several actors , such as the National Electric System Operator.

Keywords: Parallelization. OpenMP. Solar Radiation.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

GLOSSÁRIO

ASCII	Código binário que codifica um conjunto de sinais utilizando apenas 7 bits de representatividade;
Big Endian	Modo de escrita onde os bytes de maior ordem de um número serão armazenados na memória nos menores endereços;
Binário	Arquivos que não estão em formato texto onde cada byte pode ser utilizado para representar um valor;
Cache	Memória de acesso rápido interno a um sistema;
Chip	Pastilha onde estão localizados os núcleos e threads.
Clock	Número de ações que o processador consegue executar por segundo;
Flag	Mecanismo lógico que pode ativar ou desativar uma determinada característica de um código ou compilador;
Fortran	Linguagem de programação muito utilizada em análise numérica;
GOES	Satélite geoestacionário operado pelo Serviço de Informações Ambientais Americano;
Little Endian	Modo de escrita onde os bytes de menor ordem de um número serão armazenados na memória nos menores endereços;
Hardware	Equipamentos embarcados em computadores que necessitam de algum tipo de processamento, parte física do computador;
Multi-Núcleos /Multicore	Processador com mais de um núcleo;
Multithread	Núcleo com mais de uma thread;
Matlab	Software interativo de alta performance voltado para o cálculo numérico;
Núcleo	Pastilha de silício contendo transistores onde cada núcleo funciona como um processador independente;
Overhead	Qualquer processamento ou armazenamento em excesso, seja de tempo de computação, de memória, de largura de banda ou gasto para executar uma determinada tarefa;
Pipeline	Técnica de hardware que permite que o processador realize busca de uma ou mais instruções além da próxima a ser executada;
Processador Risc	Linha de arquitetura de computadores que favorece um conjunto simples e pequeno de instruções que levam a mesma quantidade

	de tempo para serem executadas;
Processador Cisc	Linha de arquitetura de computadores capaz de executar centenas de instruções complexas diferentes sendo, assim, mais versátil;
PThread	Padrão que define uma API para criar e manipular threads;
Python	Linguagem de programação de alto nível, interpretada;
Rede Infiniband	Padrão de comunicação de dados de alto desempenho;
Script	Conjunto de instruções para serem executadas em determinado sistema;
Software	Sequência de instruções a serem seguidas e/ou executadas, na manipulação de um dado e interpretado por um processador, parte lógica do computador;
Shell	Interface para acessar os serviços de um sistema operacional;
Speedup	Relação entre o tempo gasto para executar uma tarefa pelo número de threads utilizado, resultando o ganho de desempenho do código;
Thread	Pequeno programa que trabalha como um subsistema, sendo uma forma de um processo se autodividir em duas ou mais tarefas;
Workstation	Computador com capacidade de processamento superior aos computadores comuns.

LISTA DE SIGLAS E ABREVIATURAS

ANEEL	Agência Nacional de Energia Elétrica
ARB	Architeture Review Board
CAPES	Coordenação de Aperfeiçoamento Pessoal de Nível Superior
CCST	Centro Ciência do Sistema Terrestre
CDO	Climate Data Operator
CFD	Computational Fluid Dynamics
COP21	Conference of the Parties
CPTEC	Centro de Previsão e Estudos Climáticos
CQNUAC	Convenção-Quadro das Nações unidades sobre Mudanças do Clima
DAACs	Distributed Active Archive Centers
DRAM	Dynamic Random Access Memory
EPE	Empresa de Pesquisa Energética
FORTRAN	Formula Translation
GFS	Global Forecast System
GL	Ground Level
GOES	Geostationary Operational Environmental Satellite
GNU	Gnu's Not Unix
GPROF	GNU Profiler
INMET	Instituto Nacional de Meteorologia
INPE	Instituto Nacional de Pesquisas Espaciais
LABREN	Laboratório de Modelagem e Estudos de Recursos Renováveis de Energia
LAC	Laboratório Associado de Computação e Matemática Aplicada
NCEP	National Centers for Environmental Prediction
NetCDF	Network Common Data Form
MATLAB	Matrix Laboratory
MIMD	Multiple Instruction, Multiple Data
MISD	Multiple Instriction, Single Data
MIT	Massachusetts Institute of Technology

MPI	Message Passing Interface
ONS	Operador Nacional do Sistema Elétrico
OpenMP	Open Multi-Processing
PAPI	Performance Application Programming Interface
PROINFA	Programa de Incentivo às Fontes Alternativas
SONDA	Sistema de Organização Nacional de Dados Ambientais
SIN	Sistema Interligado Nacional
SPG	Serviço de Pós-Graduação
SIMD	Single Instruction, Multiple Data
SISD	Single Instruction, Single Data
UFSC	Universidade Federal de Santa Catarina
WRF	Weather Research Forecast Model

LISTA DE SÍMBOLOS

Φ_g	Radiação Solar Global
Φ_0	Radiação no Topo da Atmosfera
τ_{clear}	Transmitância de Céu Claro
τ_{cloud}	Transmitância de Céu Completamente Nublado
$\tau_{atm-dir}$	Transmitância de Céu Claro para Radiação Direta
$\tau_{cloud-dir}$	Transmitância da Nuvem
C_{ceff}	Cobertura Efetiva de Nuvens
DNI	Radiação Direta
E	Eficiênca
E/S	Entrada e Sáida
Ghz	Giga-Hertz
L_{clear}	Mínima Cobertura de Nuvens
L_{cloud}	Máxima Cobertura de Nuvens
L	Pixel da Imagem de Satélite
km	Quilometro
kWh	Quilowatt-hora
Mflops/s	Megaflops por segundo
GW	Gigawatt
n	Número de Processadores
MB	Megabyte
MW	Megawatt
PByte	Petabyte
TByte	Terabyte
T_p	Tempo de Execução em Paralelo
T_s	Tempo de Execução Serial
TWh	Terawatt-hora
s	Segundos

S	Speedup
Sf	Fração serial do Código
W	Oeste

SUMÁRIO

Pág.

1 INTRODUÇÃO E MOTIVAÇÃO

A principal preocupação ambiental nos dias atuais diz respeito ao aquecimento global e mudanças climáticas decorrentes do aumento de temperatura no planeta. A principal hipótese está relacionada com a atividade antrópica, em especial com o aumento das emissões de gases de efeito estufa - em decorrência do elevado consumo de combustíveis fósseis - e com a mudança da cobertura e do uso do solo. Os países desenvolvidos possuem grande responsabilidade no incremento destas emissões, em função do seu padrão de consumo; entretanto, há uma importante parcela de emissões proveniente dos países em desenvolvimento.

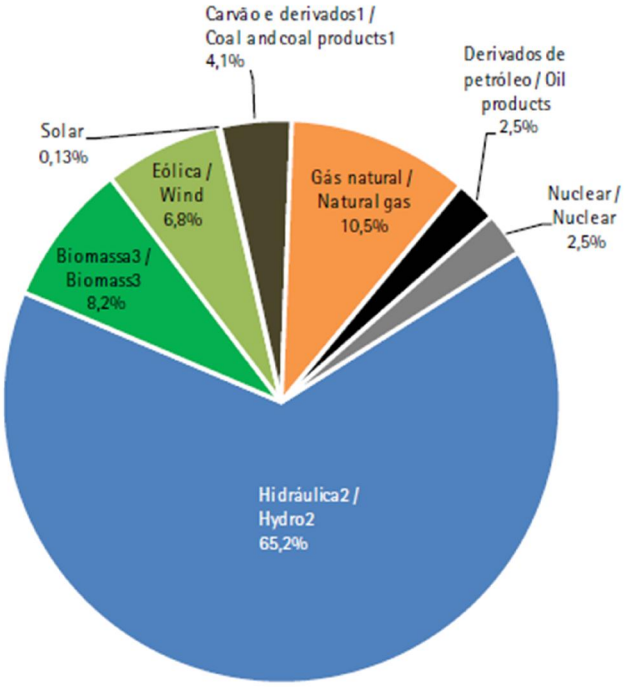
A Conferência das Nações Unidas sobre Meio Ambiente e Desenvolvimento ocorrida no Rio de Janeiro em 1992 (Eco-92 ou Rio-92) foi um marco em relação às questões ambientais e de mudanças no clima. A partir dessa conferência foi elaborado um tratado e criado a Convenção-Quadro das Nações Unidas sobre Mudanças do Clima (CQNUAC). Este tratado foi assinado por diversos países do mundo, dentre eles o Brasil, com o objetivo de estabilizar a concentração de gases de efeito estufa na atmosfera em um nível que impeça uma interferência antrópica perigosa no sistema climático (CQNUAC, 1992). A partir dessas conferências foi criado o Protocolo de Kyoto em 1992, no qual foram definidas metas de redução de emissões para todos os países que mais emitem gases. Poucos países não se comprometeram em reduzir as emissões como os Estados Unidos e Taiwan. O Brasil ratificou o documento em 2002, tendo sua aprovação interna por meio do Decreto Legislativo nº 144 de 2002.

Em 2007, um comunicado da comissão ao conselho e ao parlamento europeu definiu como meta que a União Europeia tenha, até 2020, 20% de fontes renováveis em sua matriz energética (MARTINOT, 2013).

Já a energia solar vem sendo uma das mais promissoras opções energéticas, uma vez que a maior parte do território brasileiro está localizado na região

intertropical e recebe elevada irradiação solar durante todo o ano (PEREIRA et al. 2017). Estudos indicam que a irradiação solar no país é aproximadamente o dobro da média mundial, porém tal fonte energética ainda é pouco explorada. A participação da fonte solar na matriz elétrica brasileira em 2016 era de apenas 0,02%, com uma capacidade instalada de 29,61MW. Este valor aumentou em 2018, chegando a 174,5MW, o que representou 0,13% na matriz elétrica brasileira. Na Figura 1.1 pode ser observada a participação das diversas fontes de geração de energia elétrica (BEN, 2018). O Plano Decenal de Energia 2027 (EPE, 2018), coloca a fonte solar com uma capacidade instalada de 8,6 GW – mas podendo chegar até 12 GW se considerada a geração distribuída.

Figura 1.1 - Oferta Interna de energia elétrica por fonte.



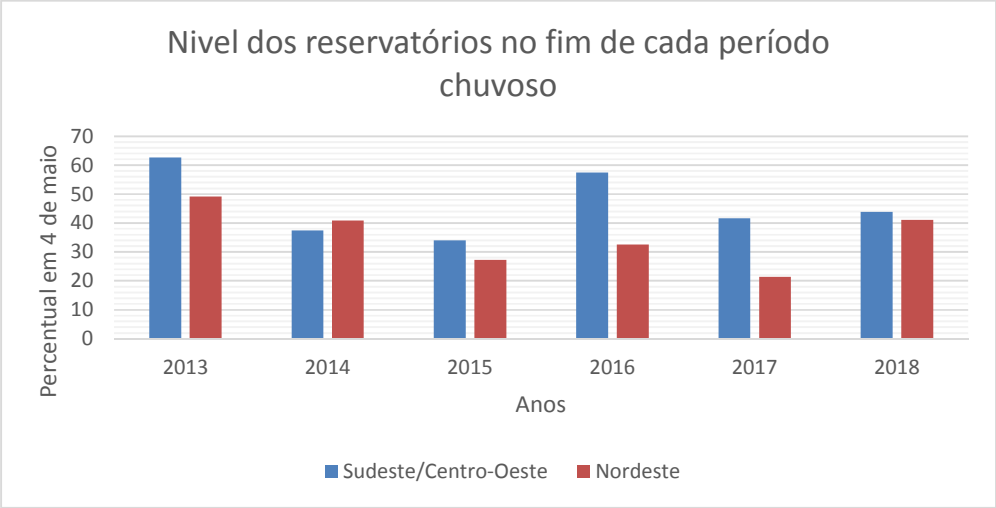
Fonte: BEN (2018).

O Brasil dispõe de uma matriz elétrica de origem predominantemente renovável, com destaque para a fonte hidríca que responde por 65,2% da

oferta interna. As fontes renováveis representam 80,4% da oferta interna de eletricidade no Brasil, que é a resultante da soma dos montantes referentes à produção nacional mais as importações, que são essencialmente de origem renovável (BEN, 2018).

Apesar da grande participação de fontes renováveis na matriz elétrica, a forte dependência das fontes hídricas nos leva a outro problema: a segurança energética. Nos últimos anos, o Brasil tem se deparado com um déficit hídrico, causado pela variabilidade climática; quando o País experimenta condições de estiagem, os reservatórios podem vir a atingir níveis muito baixos, ameaçando o fornecimento de energia. Isso também levanta questões relativas ao conflito do uso da água, que também é utilizada no abastecimento e na agricultura. De acordo com o Operador Nacional do Sistema Elétrico (ONS), a variabilidade nos níveis dos reservatórios no Sudeste/Centro-Oeste e Nordeste alterou muito de 2013 a 2018, sendo que em 2018 os valores mostraram que haverá uma menor necessidade de acionamento das usinas termoelétricas. Mas, mesmo com o ano de 2018 se mostrando um ano um pouco mais estável, no ano de 2015 os níveis desses reservatórios ficaram abaixo de 35%, como pode ser observado na Figura 1.2. Para suprir as questões energéticas, o governo brasileiro tem recorrido emergencialmente às termoelétricas, que utilizam a queima de carvão, óleo, biomassa e gás, porém, com maior emissão de gases e maior custo.

Figura 1.2 – Níveis dos reservatórios Sudeste/CentroOeste e Nordeste.



Fonte: Adaptada da ONS (2018).

Diante deste contexto, há uma necessidade iminente da diversificação da matriz elétrica brasileira, o que minimizaria as questões de Segurança Hídrica, Energética e Alimentar. É importante que esta diversificação também ocorra no sentido da inserção de fontes renováveis, em função das informações já apresentadas. A intermitência das fontes solar e eólica pode ser minimizada com o uso combinado com a hidráulica, no que se conhece como complementariedade hidro-eólica, hidro-solar e mesmo hidro-eólica-solar.

Antes da implantação de empreendimentos eólicos ou solares, é necessário conduzir estudos de verificação de viabilidade e disponibilidade do recurso. Com o mercado de energia em alta, a obtenção de dados confiáveis de irradiação solar acaba sendo um problema, já que são poucas as estações de medição com a qualidade e com as séries históricas disponíveis para este fim. Esta é uma grande dificuldade no uso de dados medidos para o mapeamento, já que a interpolação dos valores fica comprometida pela baixa cobertura espacial, em países como o Brasil. Surge então a necessidade do uso de modelos computacionais, que simulem o comportamento da atmosfera e

possam inferir, em seu modo diagnóstico, estes processos. Os dados medidos, por outro lado, possuem papel fundamental, mas apenas na validação destas ferramentas.

No Brasil, a rede do Sistema de Organização Nacional de Dados Ambientais (SONDA), realiza a implementação de infraestrutura física e de recursos humanos destinada a levantar e melhorar a base de dados dos recursos de energia solar e eólica no território nacional. Atualmente, existem 17 estações realizando medições de dados de radiação solar no Brasil (INPE.SONDA, 2017), que auxiliam na validação destas estimativas.

Estes modelos físicos possuem um elevado custo computacional, já que precisam considerar uma grande quantidade de processos físicos e a sua interação com muitos níveis verticais e com muitos pontos de grade. Este aspecto representa um grande problema de ordem computacional, que segue sendo objeto de estudo na Computação de Alto Desempenho, através de diversas técnicas de processamento paralelo.

A constante melhoria no desempenho computacional é uma demanda de modelos meteorológicos de mesoescala, como o WRF (*Weather Research Forecast Model*), aplicado em estudos meteorológicos e em previsão de tempo e clima. A demanda de redução de tempo de processamento - para que haja um aumento na resolução espacial e maior detalhamento dos processos físicos parametrizados - só é possível com o processamento de alto desempenho (MICHALAKES et al. 1999). Trabalhos como o de PINTO (2008), que realizou testes no WRF em ambientes de cluster, mostrando uma redução de cerca de 30% no tempo de execução.

O Centro de Ciência do Sistema Terrestre (CCST) do Instituto Nacional de Pesquisas Espaciais (INPE) realiza pesquisas direcionadas à quantificação do recurso solar e eólico através do Laboratório de Modelagem e Estudos de Recursos Renováveis de Energia (LABREN). Uma das suas principais ferramentas computacionais é o BRASIL-SR, um modelo físico que é utilizado

na obtenção de estimativas de irradiação solar incidente na superfície. Ele combina a utilização da aproximação de Dois-Fluxos na solução da Equação de Transferência Radiativa (ETR) com o uso de parâmetros de imagens de satélites e dados climatológicos. Um dos seus principais problemas é exatamente o elevado tempo de processamento computacional. Hoje, o modelo BRASIL-SR leva aproximadamente 27 horas para realizar um mês de estimativas de irradiação solar, considerando um cálculo de transmitância atmosférica centrado no intervalo simulado. Com a demanda do setor de energia não apenas por diagnósticos, mas também por prognósticos de curto e curtíssimo prazo, é importante que o modelo seja não só otimizado em termos de tempo de processamento, mas também tenha o seu código adaptado para o cálculo das transmitâncias diárias, melhorando a qualidade das estimativas horárias.

2 OBJETIVOS

Este trabalho tem como objetivo buscar formas para o aumento de desempenho computacional do modelo BRASIL-SR, que gera estimativas de irradiação solar focadas em valores médios mensais utilizando a transmitância calculada para o centro do intervalo mensal, e realizar alterações em seu código para calcular estimativas de forma diárias alterando para o calculo diário da transmitância.

Numa primeira fase deste estudo, foi realizada uma análise do desempenho computacional da versão atual do modelo, executada em modo sequencial. O principal objetivo desta fase é indicar quais módulos são os mais custosos computacionalmente e a melhor forma de realizar a leitura dos arquivos de entrada e saída (E/S). A partir destes resultados, foram exploradas técnicas de paralelização do código, de modo a otimizar suas seções mais intensivas sob o ponto de vista computacional. Foi utilizado a paralelização através do uso de múltiplos *threads*, em um ambiente paralelo de memória compartilhada. A implementação deste tipo de paralelização foi realizada com diretivas do padrão OpenMP, as quais permitem a criação e o gerenciamento automático de *threads*. A efetividade deste tipo de paralelização pode ser medida através da observação de como varia o desempenho total do modelo à medida que mais *threads* são empregados em sua execução. Esta abordagem, onde cada *thread* é executado em um dos núcleos disponíveis no sistema, visa explorar, de forma eficaz, o crescente número de núcleos dos processadores multi-núcleo modernos.

A segunda fase deste estudo visou realizar alterações no modelo de forma que a radiação solar incidente horária, que era calculada mensalmente com a transmitância calculada para o dia central do mês, passasse a ser calculada diariamente com a transmitância calculada para cada dia. Foi realizado o desenvolvimento de um script para realizar o pré e processamento do modelo,

auxiliando para que o modelo se torne operacional e reduzindo a interferência humana. Para tal, foram utilizadas a linguagem de programação Python e Fortran 90, na conversão de scripts já existentes em *matlab*.

3 FUNDAMENTAÇÃO TEÓRICA

3.1. **Energia solar**

O Sol, fonte de energia para a vida na terra e responsável pela fotossíntese e fenômenos climatológicos, já foi reverenciado por civilizações antigas e também motivo de debates científicos e religiosos, quando Galileu Galilei afirmou no século XVII que o Sol estava no centro do universo revolucionando a física, astronomia e todas as ciências com o chamado Sistema Heliocêntrico.

A energia emitida pelo Sol é aproveitada desde muitos anos atrás, seja para o aquecimento de residências ou mesmo para uso bélico, como quando Archimedes, que desenvolveu um sistema para incendiar navios inimigos utilizando espelhos, concentrando a radiação solar em um único ponto.

Ao longo dos anos, muitas foram as formas que se utilizavam a energia solar. Até que em 1839, Edmond Becquerel verificou que placas metálicas - de platina ou prata - mergulhadas num eletrólito, produziam uma pequena diferença de potencial quando exposto à luz (BECQUEREL, 1839). Em 1877, Willian G. Adams e Richard E. Day utilizaram as propriedades fotocondutoras do selênio para desenvolver o primeiro dispositivo sólido de produção de eletricidade por exposição à luz (GREEN, 2005)

No Brasil, em 2018, são aproximadamente 37 mil sistemas conectados à rede com uma potência que ultrapassa os 350 MW de geração distribuída, que são tipicamente urbanas e integradas em telhados e cobertura de edificações. Em 2016, a geração distribuída tinha quase que o dobro de potência instalada do que a geração centralizada, que são as usinas fotovoltaicas. No ano de 2017, mesmo a geração distribuída aumentando de 60,2 MW para 177,4 MW, um aumento de quase 300%, a geração centralizada passou a liderar o mercado solar fotovoltaico brasileiro. Com uma potência instalada de 27,8 MW em 2016, passou para 935,3 MW em 2017, justamente ano em que as usinas

fotovoltaicas garantidas pelos leilões do Ministério de Minas e Energia (MME), começaram a entrar em operação. (ABSOLAR, 2018).

3.1.1. Radiação solar

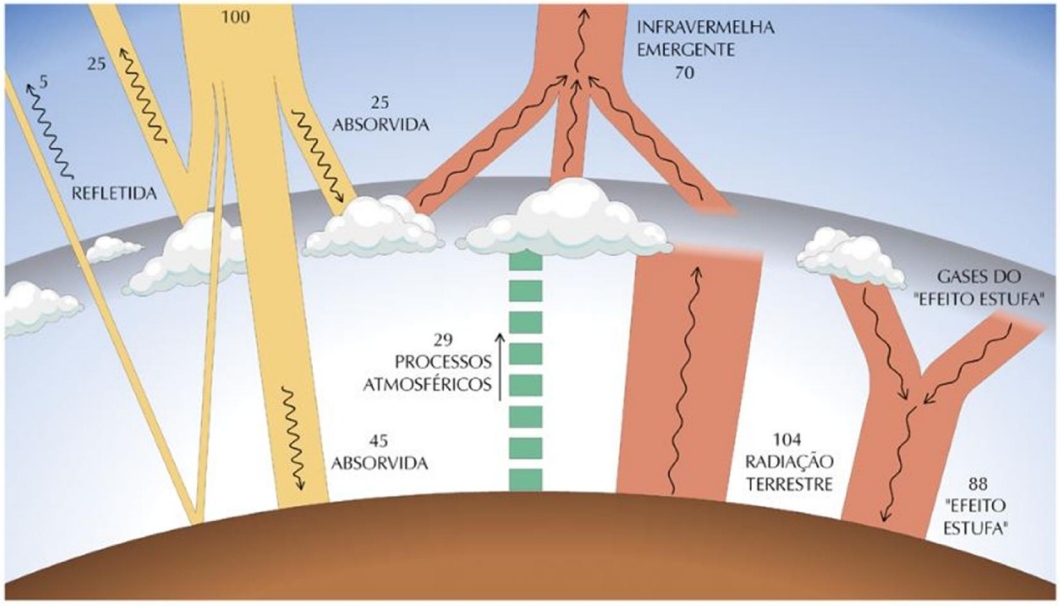
Uma fonte de energia primária pode ser considerada renovável quando as condições naturais permitem sua reposição em um curto horizonte de tempo como as energias de hidráulica, maremotriz, geotermal, biomassa, eólica e solar (GOLDEMBERG et al., 2008).

A energia irradiada pelo Sol cobre uma ampla faixa do espectro eletromagnético e cerca de 81% dessa energia chega ao Sistema Terra/Atmosfera no comprimento de onda que vai do visível ao infravermelho próximo (PEREIRA, 2017).

A energia solar na superfície da Terra é suficiente para atender 10.000 vezes o consumo de energia do mundo. Somente a luz do sol pode produzir uma média de 1.700kWh de energia elétrica por ano para cada metro quadrado de área (BRAKMANN et al., 2003). Muito além das questões energéticas, a radiação solar desempenha papel importante em diversas atividades econômicas ou mesmo naquelas relacionadas com a saúde.

Esta energia, ao atravessar a atmosfera, sofre complexas interações com os constituintes atmosféricos através de processos de absorção e espalhamento da radiação incidente (LIMA, 2015). Esses processos, mostrados na Figura 3.1, dependem do comprimento de onda da radiação e do tamanho e natureza do gás ou particulado atmosférico que interage com a radiação solar (LIOU, 2002).

Figura 3.1 – Balanço de energia global.



Fonte: Pereira et al. (2006).

As nuvens, que cobrem em média de 40% a 60% da superfície da Terra, desempenham um papel fundamental no balanço de energia do planeta, (ECHER et al., 2001). Partículas de aerossóis em suspensão na atmosfera influênciam fortemente o balanço radiativo da atmosfera e o clima, a química da atmosfera e a visibilidade, desde a escala local até as escalas regional e global (ARTAXO et al., 2006). Em regiões com incidência de muitas nuvens ou com muitos aerossóis o potencial solar pode vir a ser baixo.

As irradiações que chegam na superfície terrestre depois de sofrer os processos de absorção e espalhamento são:

- Irradiação global: é a taxa de energia total por unidade de área incidente numa superfície horizontal.

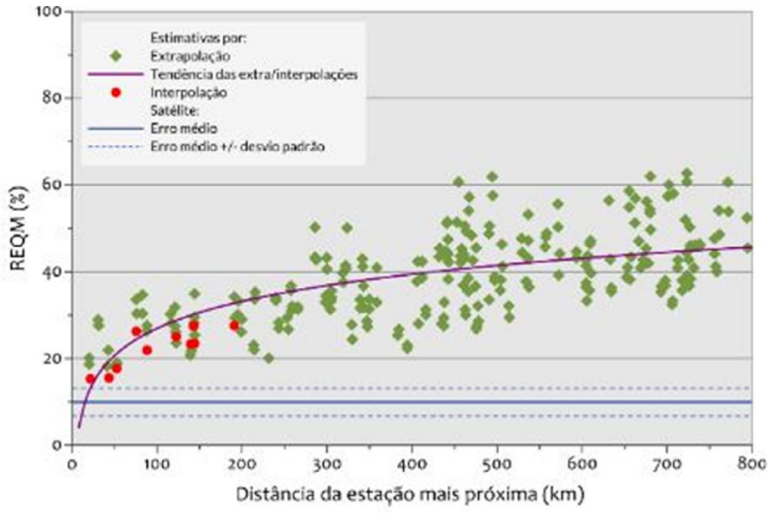
- Irradiação direta: é a taxa de energia por unidade de área proveniente diretamente do Sol que incide perpendicularmente à superfície.

- Irradiação difusa: é a taxa de energia incidente sobre uma superfície horizontal por unidade de área, decorre do espalhamento do feixe solar direto pelos constituintes atmosféricos (moléculas, material particulado, nuvens, etc.) (PEREIRA, 2017).

Para realizar a medição da irradiação que chega na superfície terrestre é necessária a instalação de sensores como os pireliômetros - que medem a irrâdiancia direta, os piranômetros que medem a irradiância global e piranômetros sombreados para a irradiância difusa. Entretanto, estas medições são pontuais e para uma melhor cobertura espacial é necessário a utilização de modelos físicos para se obter dados confiáveis. Estas informações podem ser utilizadas na identificação de áreas com potencial para a instalação de empreendimentos solares.

Utilizar dados meteorológicos é a fonte mais segura para se obter o conhecimento do potencial local de energia solar. Porém, pesquisas indicam que os erros de interpolação de dados observados em estações de superfície afastadas em mais de 30km entre si, são superiores aos erros de estimativas produzidas por modelos de transferência radiativa, como pode ser observado na Figura 3.2 (MARTINS, 2011).

Figura 3.2 – Comparação entre dados interpolados e de modelos.



Fonte: Pereira et al. (2017).

Com a extensão territorial brasileira, seriam necessárias aproximadamente 280 mil estações de coleta de dados em superfície para criar uma malha de 30km de resolução espacial, o que é totalmente inviável.

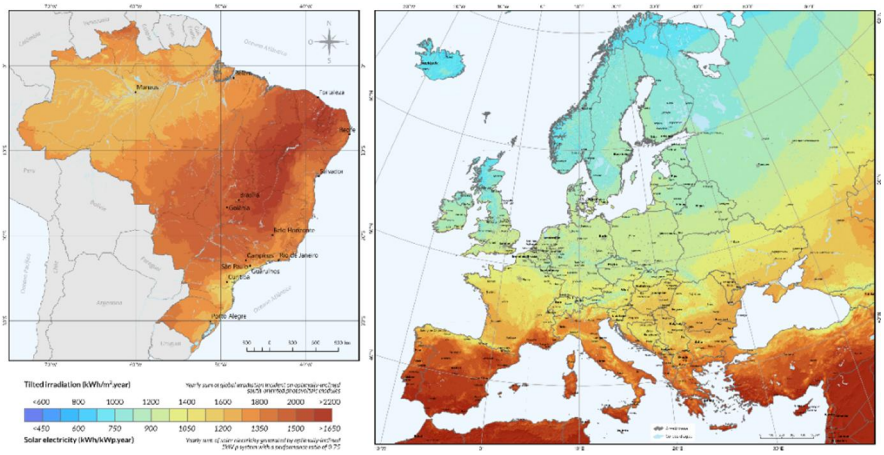
3.1.2. Radiação solar como fonte de energia

A radiação solar em fonte de energia vem se mostrando como um importante mecanismo para a diminuição da emissão dos gases do efeito estufa e para a segurança energética de países como o Brasil, que tem mais de 60% de fonte hidráulica na sua matriz elétrica.

O potencial de energia solar no Brasil é alto e se comparado com o potencial de países da Europa, que já possuem longo histórico no aproveitamento deste recurso. Se considerarmos por exemplo, a Alemanha (um dos países pioneiros no aproveitamento solar), veremos que é possível gerar mais energia durante um ano no local com menor incidência de sol no Brasil do que o com maior incidência de sol em território alemão, como mostra a Figura 3.3.

Ainda tomando a Alemanha como exemplo, tem-se este país como referência no uso do recurso solar por possuir uma politica de incentivos extremamente eficiente - a cada ano se consolida mais com grandes investimentos voltados as energias renováveis. Batizada de *Energiewende*, a Alemanha promete fazer uma transição profunda em suas fontes de energia, retirando as fontes fósseis e nucleares e inserindo 80% de fontes renováveis. Em 2016 31% da energia gerada na Alemanha já era de fonte renovável, sendo que 6% - ou 38 TWh - é de fonte solar (IEA, 2018).

Figura 3.3 – Mapas de irradiação e de produtividade.



Fonte: Adaptado de Tiepolo et al. (2017).

No Brasil, o PROINFA (Programa de Incentivo às Fontes Alternativas) foi criado em 2002 com o objetivo de aumentar a participação de fontes alternativas renováveis na produção de energia elétrica e privilegiando empreendedores que não tenham vinculo societários com concessionárias de geração, transmissão ou distribuição (ANEEL, 2018). Porém, quando criado, não houve a inclusão da fonte solar, inserida apenas em 2014, quando foi realizado o primeiro leilão que contemplava a geração fotovoltaica.

A geração de energia elétrica por fontes solares pode ser feita por diversas formas, como através de coletores solares parabólicos que concentram os raios solares sobre um ponto aumentando a densidade de fluxo de irradiância. As usinas heliotérmicas são compostas de uma torre central, que recebendo a radiação dos concentradores aquecem um sistema de armazenamento com óleos e sais fundidos e permitem a geração de eletricidade, mesmo durante períodos de nebulosidade ou noturnos. Já geradores solares fotovoltaicos, que podem ser usinas de grande porte ou instalações em solo como em telhados residenciais e coberturas de empresas, são painéis de silício cristalino e ou arsenieto de gálio, instalados em estruturas metálicas inclinadas fixas que converte a energia do sol em energia elétrica.

3.2. **Modelos de transferência radiativa**

Ao longo dos anos, vários modelos foram desenvolvidos para solucionar a equação de transferência radiativa de forma precisa e assim obter estimativas da irradiação solar na superfície. Estudos utilizando métodos que utilizam harmônicos esféricos (DAVE; CANOSA, 1974), ordenadas discretas (LIOU, 1976), Monte Carlo e diferenças finitas (LENOBLE, 1985) têm um custo computacional muito alto inviabilizando de ser executado, também, diariamente.

Um dos primeiros modelos a utilizar o método de dois fluxos foi desenvolvido por Kerschgens et al. (1978). Ele utilizava como entrada os coeficientes de absorção e espalhamento de diversos constituintes atmosféricos e aerossóis com grande detalhamento espectral. Os coeficientes de absorção e espalhamento por nuvens foram determinados com o uso de modelos desenvolvidos por Yamamoto (1962) e Deirmendjian (1969). Gautier et al. (1980) também utilizou o método de dois fluxos para estimar o fluxo de radiação solar na superfície a partir de medidas de radiância obtidas pelo satélite GOES.

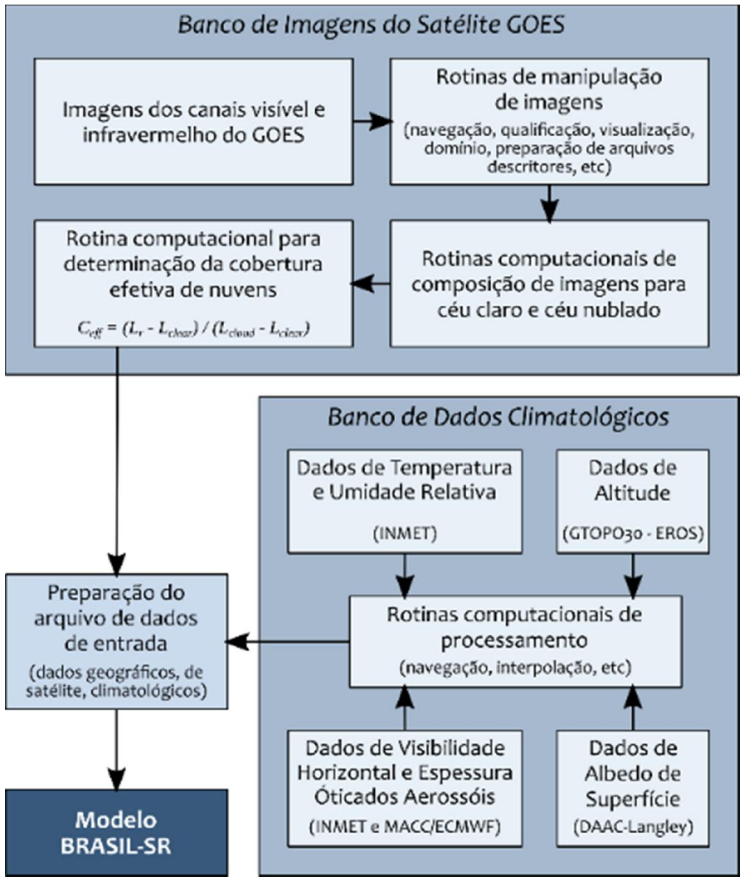
No Brasil, a utilização de modelos físicos ou estatísticos que utilizam imagens de satélite para obtenção de estimativas de radiação solar incidente na superfície é muito bem aceita na comunidade cientifica, desde que estes possuam resultados validados. Esta é uma alternativa mais interessante para a obtenção com precisão e com resolução espacial aplicável aos estudos de potencial solar. No INPE, dois modelos físicos vêm sendo desenvolvidos desde os anos 90: o modelo GL e o modelo BRASIL-SR. O modelo GL foi desenvolvido pelo Centro de Previsão do Tempo e Estudos Climáticos (CPTEC) e divide a radiação solar em três faixas espectrais; utiliza valores de radiância medidos pelo satélite geoestacionário e adota um conjunto de parâmetros atmosféricos como água precipitável, ozônio, dióxido de carbono, refletância da superfície e das nuvens. (MARTINS, 2001). O modelo BRASIL-

SR combina a aproximação de dois fluxos (MEADOR; WEAVER,1980; PEREIRA, 2017) na solução da equação de transferência radiativa com o uso de parâmetros determinados de forma estatística a partir de imagens de satélite (Pereira et al. 2017).

3.3. **Modelo BRASIL-SR**

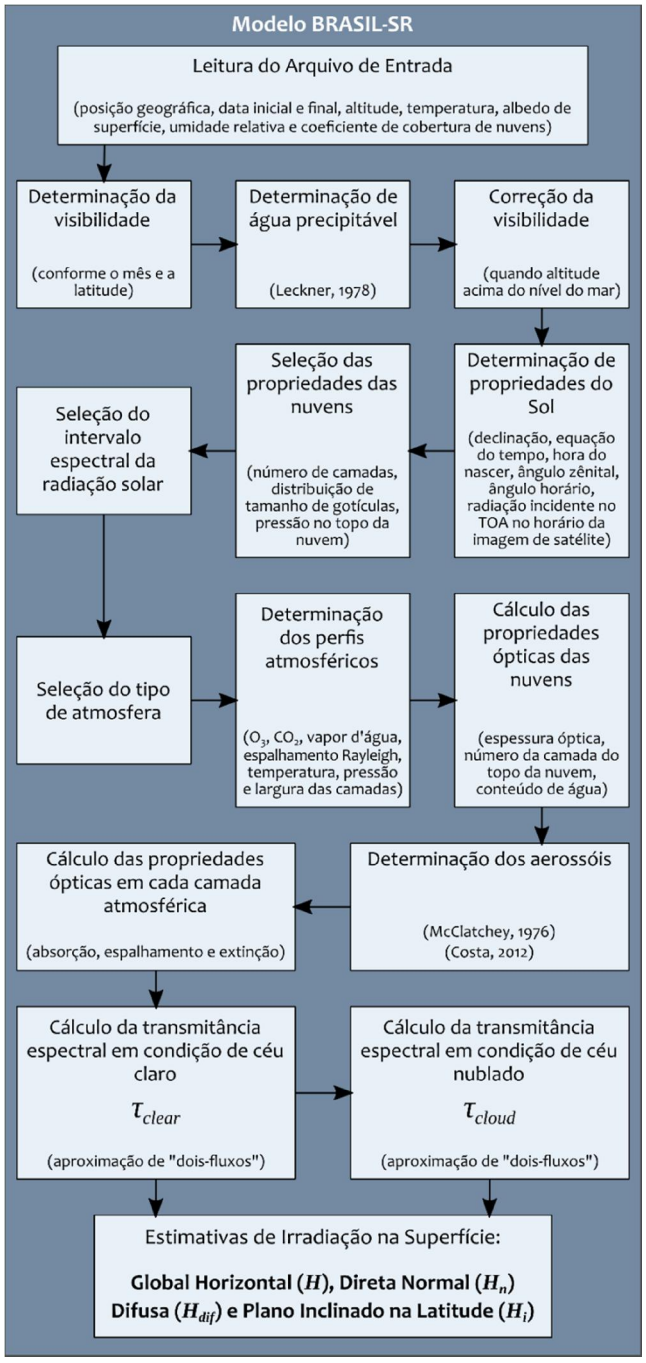
O BRASIL-SR é um modelo físico para obtenção de estimativas da radiação solar incidente na superfície. Ele combina a utilização da aproximação de Dois-Fluxos na solução da equação de transferência radiativa com o uso de informações climatológicas e parâmetros determinados a partir de imagens de satélite (MARTINS et al., 2001). Foi desenvolvido com base no modelo alemão IGMK Forschungszentrum e adaptado para as condições brasileiras pelo LABREN/CCST/INPE e pela Universidade Federal de Santa Catarina – UFSC (PEREIRA et al., 2006), sendo o principal recurso deste grupo a quantificação do recurso energético solar. O código do modelo contém mais de 8300 linhas e 68 funções, divididas em 28 arquivos, sendo escrito em linguagem Fortran 90. A Figura 3.4 mostra o pré-processamento dos dados de entrada do modelo e a Figura 3.5 o seu fluxograma dos procedimentos.

Figura 3.4 – Fluxograma do preprocessamento dos dados de entrada do modelo.



Fonte: Pereira et al. (2017).

Figura 3.5 – Fluxograma do processamento do modelo BRASIL-SR.



Fonte: Pereira *et al*. (2017).

O BRASIL-SR utiliza imagens de satélite geoestacionário para a identificação das condições de nebulosidade e foi adaptado para obter estas informações a partir do satélite GOES, que monitora o continente Americano com resolução temporal de 30 minutos, estando sobre o equador em 75º W de longitude (NASA, 2017). Os dados de temperatura, umidade e visibilidade são médias climatológicas de estações operadas pelo Instituto Nacional de Meteorologia (INMET) e foram interpoladas, utilizando o metódo de *Krigagem* linear*,* para terem a mesma resolução das imagens de satélite. Os dados de albedo foram gerados a partir das grades mensais disponibilizadas pelo Distributed Active Archive Centers (DAACs) e os dados de altitude foram obtidos pela base de dados do TOPODATA, com resolução espacial de 60 metros.

A cobertura de nuvens é o principal fator de modulação da transmitância atmosférica, de modo que o BRASIL-SR utiliza valores climatológicos das variáveis atmosféricas para estimar as demais propriedades óticas. O fluxo de radiação solar no topo da atmosfera está linearmente distribuído entre as duas condições extremas: céu claro e céu completamente encoberto (PEREIRA et al. 2006). Dessa forma, pode-se calcular o fluxo de radiação solar global incidente na superfície Φ_g, a partir da Equação 3.1:

$$\Phi_g = \Phi_0\{(\tau_{clear} - \tau_{cloud})(1 - C_{ceff}) + \tau_{cloud}\} \quad (3.1)$$

onde Φ_0 é a radiação incidente no topo da atmosfera, τ_{clear} e τ_{cloud} são as transmitâncias atmosféricas em condições de céu claro e completamente nublado e C_{ceff} é a cobertura efetiva de nuvens dada pela Equação 3.2:

$$C_{ceff} = \frac{L - L_{clear}}{L_{cloud} - L_{clear}} \quad (3.2)$$

onde L é um pixel da imagem de satélite e L_{cloud} *e* L_{clear} são as leituras em condições de céu encoberto e céu claro para o mesmo pixel da imagem.

A irradiação direta horizontal é estimada assumindo que a absorção da radiação solar pelas nuvens não é significativa e que a contribuição do

espalhamento da radiação solar causada pelas nuvens pode ser adicionada à transmitância atmosférica em condições de céu claro. É estimada pela Equação 3.3:

$$DNI = \Phi_0 . \tau_{atm-dir} . \tau_{cloud-dir} \quad (3.3)$$

onde $\tau_{cloud-dir}$ representa a transmitância das nuvens e $\tau_{atm-dir}$ é a transmitância de céu claro para a radiação direta. A transmitância $\tau_{cloud-dir}$ é estimada a partir do coeficiente de cobertura efetiva de nuvens utilizando a aproximação como mostrado na Equação 3.4. (STUHLMANN et al. 1990)

$$\tau_{cloud-dir} = \frac{(1-\tau_c)}{(\beta - \tau_c)} \quad (3.4)$$

$$\text{Onde} \begin{cases} \tau_c = (CCI + 0{,}05) & if\ CCI < 0{,}95 \\ \tau_c = 1{,}0 & if\ CCI \geq 0{,}95 \end{cases}$$

O modelo é executado através de um script que executa seus sete módulos de forma automática, cada um deles com diferentes funções na estimativa final da irradiação. São eles:

1. Cálculo da transmitância em céu claro;
2. Cálculo da transmitância em céu parcialmente nublado;
3. Cálculo das irradiações Global, Difusa e Direta;
4. Cálculo da irradiação Par;
5. Cálculo da integral diária da irradiação Par;
6. Cálculo das integrais diárias das irradiações Global, Difusa e Direta e o Plano Inclinado;

7. Cálculo das médias mensais das irradiações Global, Difusa, Direta, Par e o Plano Inclinado.

O cálculo da transmitância atmosférica é realizado para o meio do intervalo da simulação, assim, com um intervalo de 30 dias (um mês), a transmitância será calculada apenas para o 15º dia e posteriormente será extrapolada para todos os horários das imagens de satélite disponíveis para o período. As rotinas para o cálculo das transmitâncias para cada horário, são extremamente custosas do ponto de vista computacional, sendo executadas de forma serial em aproximadamente 25 horas, o que acaba inviabilizando o cálculo de estimativas diárias. Os maiores tomadores de tempo são os dois primeiros módulos do modelo, responsáveis por calcular as transmitâncias de céu claro e céu encoberto.

Os dados produzidos pelo modelo BRASIL-SR quando comparado os valores observados com os dados modelados para as médias mensais dos totais diários de irradiação global o coeficiente de correlação está na faixa de 0,81 a 0,98 e a raiz do erro quadrático médio entre 395 e 467 Wh/m^2. Em virtude destes bons resultados, os dados produzidos por este modelo acabam sendo muito utilizados pela comunidade científica, prova disso é o Atlas Brasileiro de Energia Solar lançado em 2017 e o Atlas de Energia Solar do estado do Paraná lançado em 2018, e embora os atlas e os dados do modelo, são da área de energia solar, eles acabam atendendo áreas como meteorologia, climatologia, agricultura, hidrologia e arquitetura, além de colaborar com informações climatológicas para novos empreendimentos solares.

3.4. **Processamento de alto desempenho**

Desde o surgimento dos primeiros computadores há a necessidade de se ter uma máquina capaz de realizar uma grande quantidade de cálculos em menor tempo. Em cada época, algumas arquiteturas ganhavam destaques por

oferecer velocidades de processamento extremamente mais altas que a maioria das outras máquinas, mesmo que às custas de alguma limitação nas suas funcionalidades. Assim, a busca por mais desempenho otimizando a velocidade do processador e a utilização de mais processadores em cada máquina ficou conhecida como processamento paralelo.

Seymour Cray projetou o primeiro supercomputador chamado de Cray-1 com performance de 250Mflops/s. A busca por melhor desempenho de processamento começou a se tornar fundamental para um supercomputador sendo necessário o desenvolvimento e criação de ferramentas de *hardware* e *software* para ajudar a melhorar o desempenho dos computadores, como o pipeline, processadores Cisc e Risc, vetorização, otimização de memória e paralelismo. (FONSECA, 2007)

Um importante passo foi dado em 2003, quando os fabricantes de hardware observaram que aumentar a frequência de *clock* dos processadores era complicado pois a dissipação de calor e o consumo de energia eram muito altos. A solução foi incluir várias unidades de processamento em cada *chip*, resultando em um aumento de desempenho devido à quantidade de unidades de processamento, conhecidas como núcleos. Estes novos processadores foram chamados de *multicore*, e passaram a tornar-se o componente básico de computadores paralelos.

A utilização de processamento de alto desempenho vem a cada dia se tornando mais importante para toda a sociedade. Com o desenvolvimento de novos processadores em conjunto de supercomputadores, os problemas como previsão do tempo, dinâmica celular, simulações de terremotos, dinâmica de fluidos, aplicações militares, internet e Ecommerce, cosmologia e astrofísica entre muitos outros, ficam a cada dia mais fáceis de se resolver.

Semestralmente é divulgada uma lista com os 500 supercomputadores mais rápidos do mundo, onde os Estados Unidos estão dominando as duas primeiras colocações. O *Summit* é o supercomputador mais rápido do mundo

desde junho/2018, ele possui mais de dois milhões de núcleos e uma performance de 143 PFlops/s (TOP500, 2018).

O Brasil aparece na posição 331 do ranking em dezembro/2018, com o *BC1*, com 38400 núcleos com processadores Intel Xeon e utilizado pela indústria. O Brasil teve um outro representante na posição 472 do ranking em junho/2017, o Santos Dumont, que é utilizado na área acadêmica e científica (TOP500, 2018). O Santos Dumont, fica locado no estado do Rio de Janeiro, em Petrópolis e possui um total de 18.144 núcleos distribuídos em 756 nós computacionais e conta com processadores Intel Xeon, Intel Xeon Phi e GPU (Unidade de Processamento Gráfico) da Nvidia, tornando-o hibrido. Este supercomputador conta com uma rede infiniband e armazenamento de 1,7 PBytes e um sistema de arquivos secundários com capacidade bruta de 640TBytes com 64 GigaBytes de memória *RAM* (Memória de Acesso Aleatório) (LNCC, 2017).

Cada nó computacional contém dois núcleos e cada núcleo contém sua memória cache L1, L2 e L3. Nestes núcleos a cache tem tamanho de 32kbytes, 256kbytes e 30720kbytes respectivamente. A cache L1 é dividida em L1i (memória de instrução) e L1d (memória para dados) auxiliando o núcleo de processamento a ir direto na memória de instrução para buscar uma instrução, ou indo direto para à memória de dados para realizar a busca por um dado, melhorando assim o tempo de busca da informação pelo núcleo de processamento.

Este supercomputador está acessível à comunidade de pesquisa e desenvolvimento, desde julho de 2016. Ele está sendo muito utilizado para pesquisas e desenvolvimento nas áreas de saúde, ciências biológicas, bioinformática, química, engenharia, meteorologia, linguística, geociências, astronomia e computação cientifica. Em todas as áreas citadas o supercomputador é imprescindível, pois com computadores pessoais não seria possível, ou demorariam alguns anos, para identificar mutações somáticas e

germinativas associadas ao câncer, verificar estruturas de galáxias em grande escala ou realizar simulações de alta performance de processos deposicionais devido a correntes gravitacionais.

O Santos Dumont, mesmo estando locado no Rio de Janeiro, é utilizado por vários outros estados do Brasil, mostrando sua importância na pesquisa nacional, sendo que ele tem o maior número de núcleos em um nó computacional. De acordo com o LNCC, desde agosto de 2016 até fevereiro de 2019, foram mais de 230 mil tarefas executados no supercomputador.

Os supercomputadores necessitam de uma rede de interconexão que tenha uma boa taxa de transferência de arquivos. Em redes do tipo Ethernet, sua topologia é classificada como um barramento, utilizando um roteador que fornece acesso compartilhado a todos os nós computacionais. A rede Ethernet tem uma taxa de transferência da ordem de *megabyte* por segundo e um baixo custo na sua implementação.

Outra rede de interconexão é a *InfiniBand* que suporta o protocolo RDMA (Acesso Direto a Memória Remota), que permite a uma área de memória ter acesso direto à memória de outro nó, sem passar pelos gargalos de E/S e do sistema operacional. Assim, é possível alcançar elevadas taxas de transferência de dados com baixa latência. O Santos Dumont utiliza a InfiniBand com capacidade de 58 GBytes/s com uma banda total de 112.752GBytes/s e tempo de transferência de mensagens por porta de 132 milhões de mensagens por segundo (INFINIBAND, 2019).

3.5. **Arquitetura paralela**

A arquitetura paralela é caracterizada pela utilização de vários processadores com vários núcleos (cores), é capaz de reduzir o tempo de processamento de um programa. Ao longo dos anos, muitos tipos de computadores paralelos já

foram propostos e construídos, sendo assim criada uma taxonomia para categorizá-los. (FLYNN,1972), classificou os computadores paralelos baseando-se em dois conceitos – fluxo de instruções e fluxo de dados.

Um fluxo de instruções corresponde a um contador de programa, um sistema com *n* processadores tem *n* contadores de programa e assim, *n* fluxos de instruções. O fluxo de dados consiste em um conjunto de operandos para vários fluxos de dados (TANENBAUM, 2007).

Segundo a taxonomia de Flynn (1972), Tabela 3.1, as arquiteturas paralelas podem ser divididas em quatro tipos, SISD (*single instruction, single data*), SIMD (*single instruction, multiple data*), MISD (*multiple instruction, single data*) e MIMD (*multiple instruction, multiple data*).

Tabela 3.1 – Taxonomia de Flynn para computadores paralelos.

	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIND

Fonte: Flynn (1972).

No esquema de SISD, um único fluxo trabalha sobre um único fluxo de dados, correspondendo ao processamento sequencial da máquina de Von Neumann que representa os computadores convencionais. O SIMD corresponde a mesma instrução aplicada simultaneamente a diversos dados e neste esquema se enquadram os processadores vetoriais e matriciais. O MISD representa múltiplas instruções operando simultaneamente sobre o mesmo fluxo de dados. Já o MIMD representa o processamento de múltiplos dados por múltiplas instruções, neste esquema podemos citar as arquiteturas massivamente paralelas.

O conceito de computadores paralelos foi definido por Almasi e Gottlieb (1994) como sendo uma coleção de elementos de processamento que cooperam e se comunicam para resolver grandes problemas de forma rápida. Algumas máquinas foram criadas baseando-se na taxonomia de Flynn, como Von Neumann que tem sua arquitetura, hoje considerada clássica, de computadores digitais utilizando o esquema SISD. O CM2 (Connection Machine 2) utilizando o esquema SIMD e os Multicomputadores com Multiprocessadores e Multicore que utilizam esquema MIMD.

O CM2 foi desenvolvido no MIT (Massachusetts Institute of Technology) na década de 1980. Foi planejado para processamento de Inteligência Artificial com configuração de 512 MB de DRAM e configurável para até 65.536 processadores organizados em um hipercubo, tendo que cada vértice equivale a um nó computacional. (HILLIS, 1986)

Multicomputadores são sistemas distribuídos que contém dois ou mais computadores interconectados por uma rede comunicando-se por troca de mensagem, com cada computador podendo realizar uma tarefa distinta. Os Multiprocessadores são um conjunto de processadores que não estão no mesmo chip e que utilizam memória compartilhada. Ambas tecnologias foram criadas a partir da década de 1970 para tentar suprir a necessidade crescente de processamento, porém, esbarraram na dificuldade de programação e pelo seu alto custo.

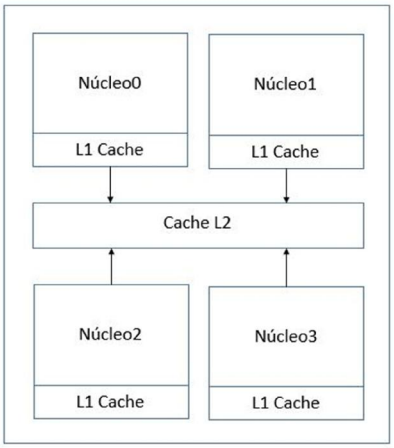
Os computadores *Multicore* foram criados em meados de 2006 e foi uma saída das fabricantes de processadores para aumentar o poder de processamento sem esbarrar no problema de superaquecimento dos chips. *Multicore* são os processadores que contém mais de um núcleo em cada *chip*, garantindo assim, o processamento simultâneo de mais de uma tarefa.

Os sistemas operacionais tratam cada um dos núcleos de forma independente e na maioria dos casos, cada núcleo contém sua própria memória *cache*; em

algumas situações a realização de acesso direto e independente à memória principal.

Quando se está utilizando dois ou mais núcleos de processamento não é possível somar as capacidades. Por exemplo, um processador com quatro núcleos com *clock* de 2 Ghz não equivale a um processador de um núcleo funcionando com *clock* de 8 Ghz, e sim quatro núcleos de 2 Ghz operando em paralelo. Abaixo, na Figura 3.6 podemos observar como ficam dispostos os núcleos e suas memórias cache.

Figura 3.6 – Exemplo de uma arquitetura *multicore*.



3.5.1. Tipos de memória compartilhada e distribuida

A taxonomia de Flynn para arquiteturas do tipo MIMD pode ser classificada de acordo com o tipo de memória do sistema, sendo de memória compartilhada ou de memória distribuída.

Memória distribuída utiliza múltiplos nós de processamento de forma independente com módulos de memória que são conectados por uma rede de comunicação.

A memória compartilhada é a memória global que pode ser acessada simultaneamente por todos os processadores, sendo diferenciada das memórias locais como a cache e registradores. Assim, os processadores realizam tarefas de forma independentes, mas cada processador faz acesso a um endereço de memória compartilhada por vez. Está arquitetura de memória tem uma vantagem em relação a facilidade no uso e rapidez na transferência de dados entre processadores, porém, o número de processadores fica limitado pelo acesso simultâneo à memória, ou seja, ele tem uma baixa escalabilidade.

A arquitetura de Memória Distribuída é aquela em que cada processador possui sua própria memória, ficando invisível a todos os outros processadores, os quais estão conectados por uma rede de comunicação. Por não possuir acesso simultâneo à memória, já que cada processador acessa sua própria memória, a sua vantagem é a maior escalabilidade. Porém, as comunicações entre os processadores têm um custo muito alto o que ocasiona um *overhead* de comunicação.

3.6. **OpenMP**

O *Open Multi-Processing* (OpenMP) é uma interface de programação multi processo de memória compartilhada em múltiplas plataformas. Foi desenvolvido pelo grupo *OpenMP Architecture Review Board* (ARB) - que é formado pelos maiores fabricantes de *software* e *hardware* do mundo. O OpenMP trabalha com a arquitetura *multicore* e *multithread*, conseguindo assim executar simultaneamente duas ou mais tarefas utilizando *threads*. Cada *thread* possui sua própria pilha de execução que compartilha o mesmo espaço de memória com outros *threads* do mesmo processo (LLNL, 2016). OpenMP utiliza variáveis de ambiente, bibliotecas de serviço e diretivas de compilação chamadas de sentinela. A diretiva de compilação é utilizada para identificar as áreas do programa que serão paralelizadas. Por exemplo, o início de toda área

paralelizada começa com uma diretiva do tipo *!$omp*, como observado abaixo (CHAPMAN et al.; 2008):

```
!$omp parallel
    write(*,*) “Hello”
!$omp end parallel
```

Quando a região que será paralelizada é determinada, é iniciada a inserção das diretivas. A diretiva *omp parallel do* determina a região paralela e ao mesmo tempo distribui as iterações do laço na região, por entre os *threads* de um grupo. Cada *thread* calcula sua parte das interações da seguinte maneira, supondo um laço com 1000 iterações, se 10 *threads* estão em uso, então, cada *thread* calcula 100 iterações do laço.

A diretiva *!$omp private()* é importante quando se necessita especificar uma variável que terá seu uso especifico em cada *thread*. Assim, durante a execução a variável, caso seja um índice de um laço, terá um valor em cada *thread*, garantindo que o *thread* não utilize uma mesma posição que está sendo utilizada em outro *thread*.

Quando no código existe uma região onde há a necessidade de ser executada por um *thread* por vez, podemos utilizar a diretiva *!$omp critical*, assim quando um *thread* estiver executando a região critica os outros *threads* irão bloquear a execução quando alcançarem essa região, e cada um executará a região de acordo com a ordem de chegada.

A forma de escalonamento descreve como as iterações do laço serão divididas entre os *threads*. O escalonamento (*schedule)* em OpenMP pode ser do tipo *static, dynamic, guided* e *runtime,* com suas diferenças mostradas abaixo:

- *Schedule static*: o laço será dividido em pedaços de tamanho definido fixo e depois será distribuído estaticamente para cada *thread*;

- *Schedule dynamic*: o laço será dividido em pedaços de tamanho variável e depois será distribuído dinamicamente para cada *thread*;

- *Schedule guided*: o número de iterações para cada *thread* irá variar, começando com um valor grande e sendo reduzido exponencialmente, à medida que o laço é executado por cada *thread*;

- *Schedule runtime:* realiza a decisão de agendamento durante a execução do programa, com a definição da variável *omp_schedule*. Neste caso o tamanho não é definido na diretiva (CENAPAD, 2018).

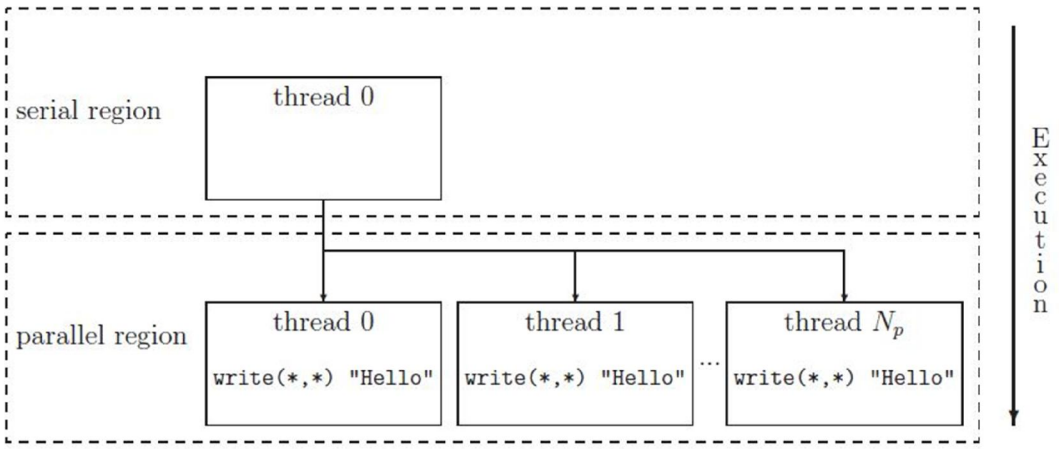
O escalonamento faz com que as iterações dos laços, sejam divididas entre os *threads*, sendo que a distribuição é definida pelo valor passado a uma variável, aqui definida de *chunk,* e para o código, Figura 3.7. Assim, se o valor passado para o *chunk* for igual a 1, significa que o OpenMP vai dividir o problema em colunas e passar para cada *thread* apenas 1 coluna, ajudando a evitar o desbalanceamento de carga no código.

Figura 3.7 – Trecho do primeiro módulo utilizando escalonamento

```
use omp_lib
Chunck = 1
!$omp parallel do private(I,J)  schedule(static, Chunk)
DO J=1,1180
  DO I=1,1784
     CALL Transmit (ndia, lon(I), lat(J), tem(I,J), umid(I,J), alt(I,J), siw(I,J), o, tclear(I,J))
  END DO
END DO
!$omp end parallel do
```

Na Figura 3.8 é possível observar como é o fluxo de trabalho da região paralela e da região serial. Cada *thread* recebe a função de escrever e imprime algo na tela. A quantidade de impressão vai depender da quantidade de *threads* que são instanciados.

Figura 3.8 – Fluxo de trabalho da região paralela.



Fonte – Hermanns (2002).

Dentre as vantagens de se utilizar OpenMP na paralelização, podemos citar: necessidade de poucas alterações no código serial existente; há uma boa estrutura para suporte à programação paralela e ajuste dinâmico do número de *threads* que será usado na aplicação; a paralelização pode ser feita incrementalmente, com partes do programa original sendo tratadas progressivamente.

3.7. **Técnica de blocagem e vetorização**

A técnica de blocagem se baseia em agrupar os elementos adjacentes da matriz, os quais serão acessados repetidamente ao longo da execução. Esse método é eficiente desde que a quantidade de dados em cada bloco seja capaz de ficar armazenada na memória *cache*. Assim, ao criar um bloco, deve se garantir que o bloco seja o maior possível, minimizando o *overhead* dos novos laços, e seja pequeno o bastante para ser armazenada na *cache*.

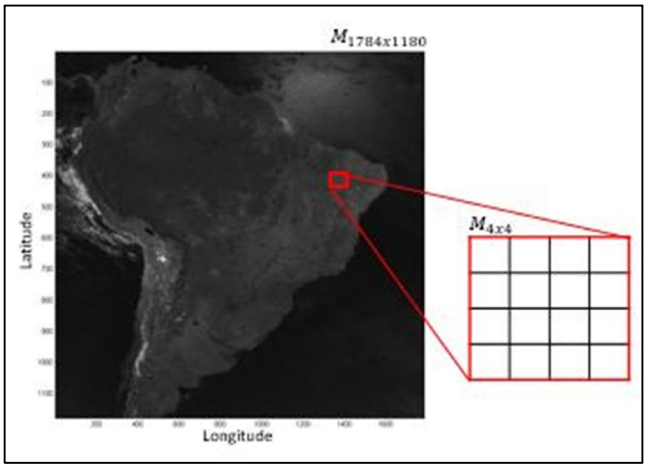
Na técnica de blocagem com matrizes bidimensionais, os *strideI* e *strideJ* são os responsáveis pelo tamanho do bloco a ser obtido e que será armazenada na *cache.*

Para entender melhor como a técnica de blocagem é realizada, na Figura 3.9, foi utilizado um trecho do código do primeiro módulo do modelo BRASIL-SR. Podemos observar que foi definido o *stride* (passo) igual a 4, como observado na Figura 3.10. Isso significa que serão criados blocos de tamanho 4x4, tentando assim, garantir que o bloco tenha um tamanho que possa ser armazenado na memória cache, na tentativa de melhorar o tempo de acesso aos dados armazenados na memória e assim melhorar o tempo de processamento.

Figura 3.9 - Trecho do primeiro módulo do modelo com blocagem.

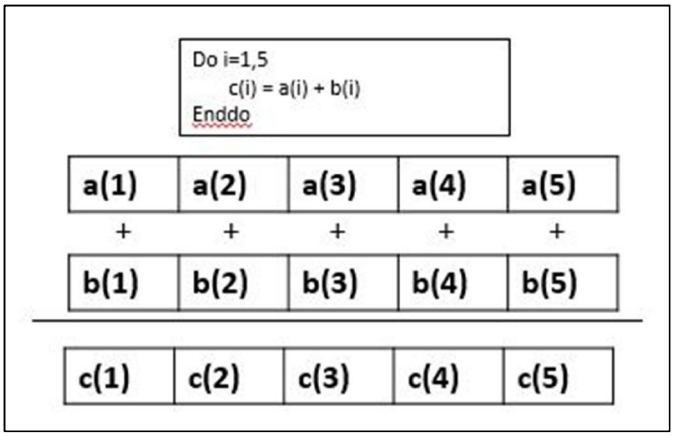
```
ni = 1784,   nj = 1180
stridei = 4, stridej = 4
do kj=1,nj,stridej
  do ki=1,ni,stridei
    do J=kj,min(nj,kj+stridej-1)
      do I=ki,min(ni,ki+stridei-1)
     CALL Transmit(ndia, lon(I), lat(J), tem(I,J), umid(I,J), alt(I,J), siw(I,J), 0, tclear(I,J))
     CALL Transmit(ndia, lon(I), lat(J), tem(I,J), umid(I,J), alt(I,J), siw(I,J), 1, tcloud(I,J)
      enddo
    enddo
  enddo
enddo
```

Figura 3.10 – Técnica de blocagem.



Outra técnica investigada para a redução do tempo de processamento foi a vetorização. Ela consiste na capacidade do processador de realizar uma operação em dois ou mais elementos ao mesmo tempo, tendo assim, um ganho de desempenho. Esse ganho acontece da seguinte maneira: por exemplo, se realizar a soma de dois vetores de cinco elementos cada, de forma escalar, o processador irá executar uma instrução com uma operação de cada vez; utilizando a vetorização, o processador realizaria uma instrução com cinco operações, como pode ser observado na Figura 3.11. Para realizar a vetorização o cálculo de cada resultado não pode depender dos resultados anteriores, pois quando for realizar as iterações do laço, cada iteração deve computar um dos cinco elementos de forma independente. A vetorização é realizada através de *flag* de vetorização e com a *flag* que não permite a vetorização (-xAVX e –no-vec), ambas do compilador *ifort*, assim é possível garantir que o código será vetorizado e que uma versão, executada, não utilizara nenhuma forma de vetorização, podendo assim, comparar as execuções do modelo BRASIL-SR em modo vetorizado e não vetorizado.

Figura 3.11 – Exemplo de uma somatória de dois vetores.



Fonte: Adaptada NACAD (2018).

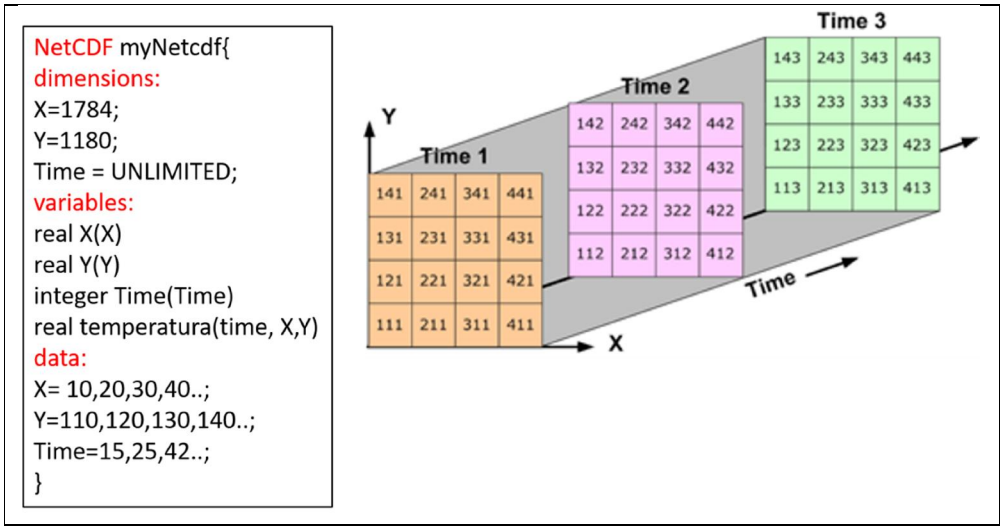
A vetorização utiliza a arquitetura *SIMD* e a técnica de pipelining, pois, muitas instâncias de uma simples tarefa são executadas simultaneamente.

3.8. **NetCDF**

O NetCDF (Network Commom Data Form) que é uma especificação desenvolvida pela fundação UNIDATA e oferecida sob a forma de bibliotecas utilitárias, visando o armazenamento de grandes volumes de dados. Além disso, um dos objetivos do formato é o acesso eficiente a pequenos subconjuntos de grandes conjuntos de dados, fazendo uso do acesso direto em vez de acesso sequencial, sendo muito mais eficiente quando a ordem em que os dados são lidos é diferente da ordem em que foram gravados (UNIDATA, 2018).

Na Figura 3.12 podemos observar como é o dataset do NetCDF e como os dados ficam disposto nos arquivos em NetCDF. No dataset é definido o tamanho das matrizes e o tipo de informação que existe dentro do NetCDF.

Figura 3.12 – Dataset do NetCDF e disposição dos dados.



Fonte : Adaptado Arcgis (2018)

Com os dados podendo ser agrupados em um arquivo em formato NetCDF, tanto a leitura como a escrita deste arquivo, são realizadas de forma mais eficiente pelo código, conseguindo assim, um ganho no tempo de processamento do modelo.

3.9. **Métricas de desempenho**

As métricas de desempenho visam relacionar os dados obtidos com o tempo de processamento, podendo assim prever possíveis problemas com o código inserido no supercomputador. Com elas, é possível identificar uma sobrecarga (*overhead*), permitindo que o código possa ser alterado para melhorar a eficiência.

3.9.1. GNU profiler

A ferramenta GNU Profiler (GPROF) foi utilizada para determinar quais funções ou rotinas do código estão consumindo mais tempo de processamento, quando o modelo é executado em seu modo sequencial original. Com o GPROF foi possível apontar quanto tempo está sendo gasto em cada parte do código. Ele gera estatísticas indicando a percentagem de tempo de execução de cada função, tempo gasto por função, número de vezes que a função foi chamada e tempo em segundos gasto em uma função. Assim, foi possível realizar uma análise do tempo de consumido por cada função.

3.9.2. Ganho de desempenho (*speedup*)

Existem diversas métricas para verificação de melhoria de desempenho de um código. O *speedup* é uma dessas, com a divisão do tempo de processamento serial (T_s) pelo tempo de execução em paralelo (T_p), como pode ser observado na Equação 3.5. Com este cálculo é possível observar quantas vezes o algoritmo paralelo é mais rápido que o algoritmo serial. Ou seja, se $S = P$, logo, $T_p = {}^{T_s}/_{P}$, quando isso ocorre, chama-se de *speedup* linear. Quando $S < P$, diz que o *speedup* é sublinear, e se $S > P$, o *speedup* é super linear, sendo que a super linearidade não é algo comum, porém pode ocorrer.(LIN; SNYDER, 2008)

$$\boldsymbol{S = \frac{T_s}{T_p}} \qquad (3.5)$$

Onde *S* é o valor de *speedup*, *P* é o número de processadores, T_p é o tempo paralelo e T_s é o tempo serial.

3.9.3. Eficiência

A partir do *speedup* é possível obter a métrica de eficiência, que representa o quanto o sistema paralelo foi explorado pelo algoritmo. Ela é uma medida normalizada do *speedup* e é calculada pela Equação 3.6, onde S é o *speedup* e P o número de processadores.

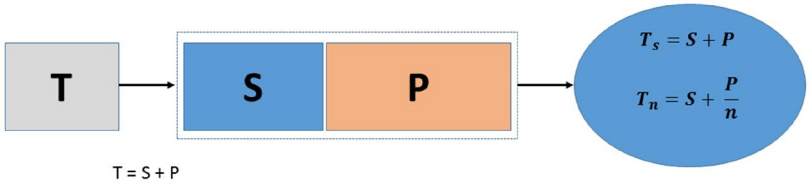
$$\boldsymbol{E = \frac{S}{P}} \qquad (3.6)$$

Para a eficiência ser ótima, ela precisa ser igual a 1 (100%), assim ela mostraria que todos os processadores foram plenamente utilizados e a eficiência é linear, sendo que ela varia de 0 a 1. Se a eficiência for maior que 1 ela é chamada de eficiência super linear. (LLNL, 2016)

3.9.4. Lei de Amdahl

A lei de Amdahl identifica o limite de *speedup* possível quando dentro de um código há uma região que sempre será executada em modo serial e uma outra que pode ser executada de forma paralelizada. De acordo com a lei de Amdahl, o *speedup* máximo é limitado pela fração do código que precisa ser executada em modo serial. Observando a Figura 3.13, em que n é o número de processadores, combinando as expressões com a definição de *speedup*, obtém-se a lei de Amdahl, mostrada na equação 3.6.

Figura 3.13 – Trecho serial e paralelo de um código.



$$S_n = \frac{1}{Sf + {}^{(1-Sf)}/_{n}} \quad onde \ \lim_{n\to\infty} S_n = \frac{1}{Sf} \begin{cases} Sf - Fração\ serial\ do\ código \\ n - Número\ de\ processadores \end{cases} \qquad (3.6)$$

Com a lei de Amdahl é possível encontrar a máxima melhora esperada para um código quando apenas uma única parte do código é melhorada, podendo assim ser utilizado para prever o máximo ganho de desempenho teorico utilizando múltiplos núcleos.

3.10. **Aplicações em supercomputadores**

Com a evolução dos supercomputadores, vários estudos foram conduzidos para melhoria do desempenho das aplicações. Freytag et al. (2015) utiliza técnicas de paralelização para melhorar a eficiência e diminuir o tempo de processamento do problema do caixeiro viajante. Naquele trabalho foram utilizadas ferramentas de *PThreads*, OpenMP, MPI e CUDA, sendo que *PThreads* e MPI reduziram o tempo de execução em 2 vezes. Pereira et al. (2010) utilizaram OpenMP e MPI para melhorar o tempo computacional do modelo utilizado na previsão da dinâmica da ionosfera terrestre. Com a implementação de ambas as técnicas de paralelização uma abordagem hibrida reduziu o tempo de execução de 2852,34 segundos para 720,96 segundos. Andrade et al. (2016) realizou uma avaliação de paralelização de um algoritmo genético aplicado ao problema de roteamento de veículos com OpenMP. Os resultados foram comparados com trabalhos anteriores e foi observado apenas um pequeno aumento no *speedup*, identificando assim uma limitação na implementação do algoritmo genético.

Tomita et al. (2003) apresentou a metodologia utilizada na paralelização com OpenMP e vetorização do modelo numérico de previsão do tempo Eta. A primeira etapa visava aumentar a vetorização devido ao uso dos pipelines. Naquele trabalho foram utilizadas técnicas de transformação de laços

sequenciais em laços paralelos substituindo o algoritmo sequencial por outro paralelo, inversão de laços permitindo a vetorização na maior dimensão da matriz e quebrando laços grandes sequenciais em sequencias de laços menores, muitos deles paralelos. Na segunda etapa foi realizada a paralelização para memória compartilhada e o resultado obtido foi muito satisfatório: a versão do código original rodava com o tempo de 3967 segundos, com o primeiro nível de paralelização foi para 2378 segundos e no segundo nível de paralelização foi para 522 segundos. Amritkar et al. (2014) descrevem estratégias de paralelização em OpenMP e MPI para o método de elementos discretos, para simular sistemas de partículas densas acopladas à dinâmica de fluidos computacionais (sigla em inglês CFD). As equações de campo de CFD são melhor paralelizadas pela decomposição do domínio espacial e a fase de partículas de N-corpos é melhor paralelizada sobre o número de partículas. Quando acoplados, ambos os modos são necessários para a paralelização eficiente. O resultado apresentado mostra que a versão com OpenMP está entre 50-90% mais rápida que à versão com MPI.

Com os trabalhos acima, podemos observar a importância da arquitetura paralela e a colaboração que ela está dando à ciência e à sociedade. Com o mundo cada vez mais conectado e com uma quantidade cada vez maior de dados a serem analisados, decorrente de uma representação mais precisa dos fenômenos físicos que estão sendo modelados, os supercomputadores precisam cada vez mais melhorar o seu desempenho de processamento.

4 MATERIAIS E MÉTODOS

Para a realização dos estudos previstos neste trabalho, foi utilizado o banco de dados de imagens de satélite da família GOES, disponibilizada pela DSA/CPTEC/INPE ao LABREN nos últimos anos. Para uma avaliação mais consistente, optou-se pela escolha do mês de abril de 2016, pois este é um dos meses com maior quantidade de imagens de satélite disponíveis, o que oferece o maior custo computacional possível. Portanto, foram utilizadas 665 imagens deste mês, com resolução temporal de 30 minutos e resolução espacial de 0.03º em longitude e 0.05º em latitude, o que corresponde a aproximadamente a 3km x 5km do ponto nadir do satélite.

O supercomputador Santos Dumont foi o recurso utilizado para esta pesquisa, pois é um ambiente adequado para os testes propostos. Com a utilização dos processadores Intel Xeon com 12 núcleos por unidade de processamento, havendo dois processadores por nó, permite uma adequada avaliação do uso de processadores multi-core modernos. Foi utilizado o nó computacional contendo 504 nós com 2 unidades centrais de processamento Intel Xeon com 24 núcleos por nó.

Para a compilação do código do modelo, foi utilizado o compilador *Intel ifort* versão 16.0.2 em um ambiente computacional Linux, disponível no Santos Dumont. O intel ifort foi utilizado pois em várias pesquisas comparando compiladores e linguagens ele sempre mostrou resultados melhores com manipulação de arquivos em NetCDF, manipulação de matrizes e cálculo numérico. (NASA, 2019). Mesmo sendo um compilador com licença proprietária, a Intel disponibiliza uma versão gratuita para estudantes e/ou pesquisa, assim, os testes iniciais do modelo são executados em computadores pessoais e depois de verificado a não existência de erros no código, ele é enviado para o supercomputador Santos Dumont para ser compilado e executado.

A paralelização do modelo foi realizada utilizando as diretivas de OpenMP. Esta forma de paralelização foi escolhida devido a dois fatores principais: (a) permite a paralelização gradativa do código sequencial original, através de alterações sucessivas e independentes em trechos do código; e (b) grande popularização atual de processadores multi-núcleo (multi-core), com custo moderado, que podem ser facilmente empregados pelas instituições interessadas em executar o modelo BRASIL-SR no futuro próximo.

A paralelização, neste trabalho, foi realizada nos 7 módulos que compõem o modelo BRASIL-SR. Em cada um dos módulos foram paralelizados todos os laços dando uma atenção maior para os mais custosos computacionalmente. Em todos os módulos, foram utilizados as diretivas *!$omp parallel do private(i,j)* como pode ser observado na Figura 4.1.

Figura 4.1 – Código do módulo 1 com a diretiva de OpenMP

```
!$omp parallel do private(I,J)
DO J=1,1180
  DO I=1,1784
    IF(XALT(I,J).GT.-98.0) THEN
      IF (DMACC(I,J) < SSIW(I,J))THEN
        SSIW(I,J) = DMACC(I,J)
      ENDIF
      CALL TRANSMIT(XLAT(J),XALB(I,J),XUMI(I,J),XALT(I,J),SSIW(I,J),TCLEAR(I,J),TDIR(I,J))
      CALL TRANSMIT(XLAT(J),XALB(I,J),XUMI(I,J),XALT(I,J),SSIW(I,J),TCLOUD(I,J),TTDIR)
    ELSE
       TCLEAR(I,J) = -1.0
       TCLOUD(I,J) = -1.0
       XTDIR(I,J) = -1.0
    ENDIF
  END DO
END DO
!$omp end parallel do
```

O modelo, em sua forma atual, realiza suas operações de E/S em formato binário e texto. Também foi avaliada a implementação destas operações a partir de arquivos em formato NetCDF afim de avaliar os tempos de processamento de cada módulo, inclusive o impacto das operações de E/S.

Além da paralelização do código utilizando OpenMP e da conversão dos arquivos de E/S, outras otimizações de desempenho foram investigadas

através de técnicas disponíveis na literatura atual, tais como blocagem e vetorização, visando melhorar ainda mais o desempenho dos laços e o uso da hierarquia de memória pelo programa.

Com a blocagem, pequenos trechos são criados e alocados na memória *cache*, como pode ser observado na Figura 3.8. Os *strideI* e *strideJ* são os responsáveis pelo tamanho do bloco a ser obtido e que será armazenado na *cache* e a função *min* é utilizado para verificar se o laço chegou ao seu final.

Para realizar os testes com blocagem no modelo, considerando que a resolução da grade (equivalente a resolução da imagem de satélite para a área em questão) é de 1784x1180 pontos, foi escolhido o máximo divisor comum entre eles, gerando assim, matrizes quadradas de tamanho 4x4. Além deste tamanho, foi utilizado nos testes matrizes de 8x8 e 32x32.

Foi utilizado o PAPI (*Performance* Application Programming Interface) que fornece o acesso independente do sistema operacional e da máquina aos contadores de desempenho de *hardware* contidos nos processadores. Esse acesso é padronizado com as chamadas de eventos, que podem ser diferentes em cada fabricante de processador (ICL,2018).

Para cada chamada de evento um contador é acessado e a partir destas informações é possível verificar as taxas de acertos e erros na *cache*. Isto é muito útil para verificar junto a técnica de blocagem se o tamanho da matriz utilizada está sendo alocada na *cache* ou memória principal. Pode-se citar, como exemplo, os contadores:

- *PAPI_LD_INS,* que realiza a contagem de instruções de leitura na memória;
- *PAPI_SR_INS,* que realiza a contagem de instruções de escrita na memória;

- *PAPI_L1_TCM,* que mostra a quantidade de acesso não atendidos pela *cache L1*.

Foram realizados testes com o modelo atual e com o modelo com dados em NetCDF, compilados com a *flag* de vetorização e com a *flag* que não permite a vetorização.

A alteração do código do modelo para executar de forma diária foi realizada através de alterações de alguns laços *DO* e para o caso do modelo sendo executado em modo diário, uma nova metodologia para a estimativa da transmitância atmosférica, já que atualmente o código do modelo define sendo que atualmente a mesma é calculada para o meio do intervalo de integração (nos moldes atuais de execução, um mês) e depois é extrapolada para o restante do intervalo.

Para automatizar o pré-processamento dos dados de entrada do modelo, foi desenvolvido um *script* em linguagem *Python*, que também utiliza o *software* de pacote de comandos CDO (Climate Data Operators). Este pacote de linhas de comando é utilizado para a análise e o processamento de dados meteorológicos e climáticos através de técnicas estatísticas e aritméticas, sendo extremamente uteis para manipulação de informações que estão em formato binário, de grade ou mesmo em formato ascii (SANTOS, 2014).

O processamento do modelo BRASIL-SR era executado quase que manualmente – com a chamada de módulo por módulo por meio de um *script shell*, a sua modernização também inclui a inserção no pré-processamento de demandas específicas, como a padronização dos dados de entrada. Para realizar este procedimento, alguns códigos computacionais eram executados de forma manual, de modo a alterar a forma de escrita do dado de satélite (*Little Endian e Big Endian*), renomeá-los, reamostra-los para a dimensão da grade de interesse, calcular os índices de máxima e mínima cobertura de nuvens para cada horário e só depois, a cobertura efetiva de nuvens para cada imagem a ser utilizada nas estimativas.

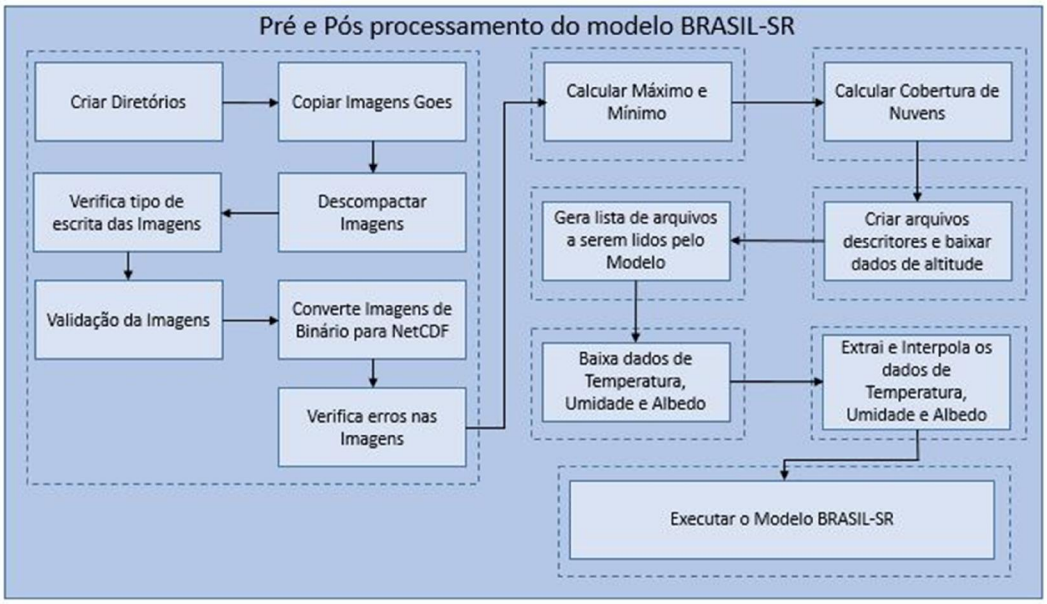
Além disso, é necessário que seja realizada uma verificação para confirmar se existem imagens de satélites em todos os canais utilizados (visível, infravermelho e de navegação). Posteriormente, as imagens de satélite passam por um processo de validação e verificação da existência de algum pixel com código de erro. Somente depois destes passos, o modelo está pronto para ser executado.

Portanto, foram desenvolvidos diversos *scripts* para realizar cada passo do pré-processamento e do efetivo processamento das simulações. Eles permitem que sejam realizadas modificações em características das funções, seja por algum tipo de atualização da base de dados ou até mesmo na inserção de novas tarefas, como verificações de consistência dos arquivos, por exemplo. Os scripts podem ser observados no Apêndice A.

Algumas das operações que são necessárias no pré-processamento como calcular a cobertura de nuvens, calcular a mínima e máxima cobertura de nuvens, validação dos dados de entrada e conversão dos dados climatológicos, não são realizadas pelos *scripts* em *Python* e sim por códigos desenvolvidos em linguagem Fortran90, podendo ser observado no Apêndice B, sendo que estas funções são mais complexas e realizam algum tipo de cálculo.

Os *scripts* desenvolvidos em *Python*, para auxiliar o pré e pós processamento, colaboram com o modelo para que o mesmo se torne operacional. Foram desenvolvidos *scripts* para baixar da página de *FTP* os dados de satélites, baixar os dados climatológicos, gerar todos os arquivos de configuração e criação dos diretórios onde são armazenados os dados de E/S, o fluxograma dos *scripts* pode ser observado na Figura 4.2. Antes do desenvolvimento dos *scripts*, era necessário realizar todos os passos citados de forma manual, acarretando em um tempo final, para realizar o pré-processamento, muito alto.

Figura 4.2. – Fluxograma dos *scripts* em python.



Foi gerado um novo código para a criação dos arquivos de máxima e mínima cobertura de nuvens. A versão atual do modelo utiliza os percentis 0 e 100 de cada um dos píxeis das imagens de satélite como sendo os valores mínimos e máximos em um determinado período de simulação. Entretanto, esta metodologia pode inserir valores espúrios nos limites de cobertura efetiva de nuvens, já que estes dados estão sempre passíveis a valores inconsistentes (o sombreamento gerado pelo relevo ou brilho acima do real, por exemplo).

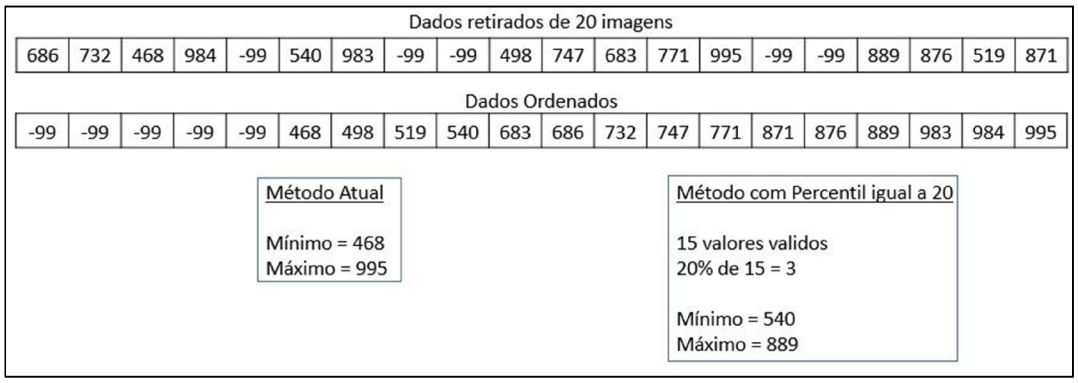
Um código foi desenvolvido para calcular os índices de máxima e mínima cobertura de nuvens. Para a realização do calculo é necessário passar como parâmetros de entrada os valores de percentis a serem utilizando. Estes valores servem como parâmetro na definição da cobertura efetiva de nuvem.

O processo para o cálculo destes valores é iniciado pela ordenação dos dados de cada um dos píxeis de todas as imagens de satélite a serem utilizadas e

posteriormente realizando o corte do percentil onde se obtém o valor mínimo e máximo de cada pixel.

Abaixo, na Figura 4.3, pode-se observar melhor como funcionam estes algoritmos, utilizando o percentil igual a 20. O método desconsidera os valores com flags de erro ou código noturno (-99) e logo após a ordenação, é realizado o corte utilizando o valor do percentil para obter o índice de máxima e mínima cobertura para cada pixel. Com está técnica, podemos observar que o índice de cobertura mínima de nuvem do canal visível do satélite aumentou de 468 para 540 e o de cobertura máxima de 995 diminuiu para 889.

Figura 4.3 – Método atual e o método com percentil igual 20.



O *script* desenvolvido para tornar o modelo BRASIL-SR operacional pode ser utilizado tanto para executar o modelo de forma diária como de forma mensal. A diferença é que na forma diária o *scritp* faz a chamada do código executável que foi alterado para ter o modelo executando apenas um dia. O modelo, em modo mensal, calcula a transmitância para o 15º dia do mês da integração e aplica para todos os horários do mês que está sendo calculado. Com o modelo, em modo diário, a transmitância é calculada para cada horário do dia a ser executado. Para isso, os laços responsáveis por percorrer todos os dias do mês, foi alterado para percorrer apenas um dia do mês.

Com a alteração do modelo para ser executado em modo diário, é fundamental a utilização de variáveis climatológicas (temperatura, albedo e umidade) para o dia que será executado o modelo. O modelo mensal utiliza médias mensais de 1999 até 2015, já, o modelo diário utilizará dados (temperatura, albedo e umidade) de análise do dia as 6, 12 e 18 horas UTC do modelo *Global Forecast System* (GFS) disponibilizados pelo *National Centers for Environmental Prediction* (NCEP).

Os dados climatológicos, obtidos no NCEP, contém aproximadamente cinquenta variáveis, sendo que somente três são utilizadas pelo modelo. Assim é necessário realizar a extração e a interpolação desses dados. Foi utilizado os comandos do CDO por ser fácil de se trabalhar com a ferramenta, pela facilidade de escrever um *script* utilizando os seus comandos, pela interface amigável, pela facilidade em se converter arquivos de *grib2* para NetCDF, recortar arquivos, interpolar arquivos e extrair variáveis dos arquivos.

5 RESULTADOS EXPERIMENTAIS

Todos os tempos apresentados neste trabalho estão em horas e segundos e foram calculados pelas médias de cinco execuções do modelo BRASIL-SR, em suas diversas configurações.

A paralelização, neste trabalho, foi realizada em cada um dos 7 módulos que compõem o modelo. O primeiro módulo é o mais custoso computacionalmente tanto na versão atual como na versão com dados de E/S em NetCDF. Utilizando apenas 1 *thread*, o tempo de processamento é maior que 60 mil segundos (16 horas). O módulo 7 é o que tem menor tempo de processando, um pouco mais de 10 segundos nas duas versões do modelo BRASIL-SR.

5.1. **GPROF**

Inicialmente, a ferramenta *GPROF* foi utilizada para realizar uma análise para saber quais os trechos mais críticos do código. Para o GPROF ser habilitado, é necessário a inserção da opção *–pg* (program GPROF) na compilação. Depois de compilado o código, é necessário executar o arquivo binário, gerado na compilação, para que as informações do código sejam enviadas para o arquivo de saída (*gmon.out*). No último passo, a ferramenta GPROF é executada com o binário criado na compilação e com o arquivo que foi gerado da execução do binário (*gmon.out)*. Estas informações são enviadas para um arquivo texto contendo todas as informações do código como, o tempo de execução de cada função.

Todos os passos necessários para a utilização do GPROF podem ser observados na Figura 5.1, onde é mostrado a sequência de linhas de comando que foram utilizadas na execução da ferramenta GPROF.

Figura 5.1 - Comandos necessários para a execução da ferramenta GPROF.

```
! Execução do módulo 1 com a flag do GPROF
ifort -pg -o Modulo_1 -qopenmp brasilsr.f90 Day.f90 Transmit.f90 d2str.f90 Astro.f90 Atmosphe.f90 Strpsrb.f90 -
I/opt/netcdf/include -L/opt/netcdf/lib –lnetcdf

! Execução do Módulo 1
./Modulo_1

! Executando a Ferramenta GPROF
gprof modulo_1 gmon.out > saida_modulo_1.txt
```

Verificando o primeiro e o segundo módulo, foi observado que a função *MATBAU*, responsável pelo cálculo da irradiação difusa em cada intervalo espectral, é uma das que mais consomem tempo de processamento, com aproximadamente 34% do tempo total da execução sequencial, conforme pode ser observado na Figura 5.2.

No módulo 3, o corpo do programa principal (*main*) é o que mais consome tempo de processamento, um pouco mais de 28%, podendo ser observado na Figura 5.3. Isto é esperado, já que, a função do módulo 3, que calcula as irradiações global, direta e difusa, precisa realizar uma quantidade pequena de cálculos. No módulo 4, o programa principal consome um pouco mais de 24% do tempo de processamento, podendo ser observado na Figura 5.4, sendo que, a função *Swtr8a*, que calcula a irradiação par, consome um pouco mais de 18% do tempo de processamento.

No módulo 6, o responsável por consumir o maior tempo de processamento é a função matemática *acos*, que calcula o arco cosseno, como pode ser observado na Figura 5.5, juntas, as funções matemáticas no modulo 6 consomem mais de 56% do tempo de processamento. Nos módulos 5 e 7, o programa principal é o que mais consome tempo de processamento, 46% e 61% respectivamente, podendo ser observado nas Figuras 5.6 e 5.7.

Figura 5. 2 – Saída do GPROF dos módulos 1 e 2, com informações de tempo e chamadas das funções.

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ks/call  Ks/call  name
 34.06    808.79   808.79 568382400     0.00     0.00  matbau_
 15.29   1171.86   363.07                             __svml_log2_h9
 14.42   1514.22   342.37  4210240     0.00     0.00  strpsrb_
 13.02   1823.53   309.31                             __svml_exp2_h9
  8.07   2015.14   191.61  4210240     0.00     0.00  ptzqd_
  5.60   2148.14   133.00                             __libm_exp_l9
  1.08   2173.69    25.55                             __svml_pow2_h9
  0.98   2197.01    23.32                             __svml_log2_mask_h9
```

Figura 5.3 – Saída do GPROF do módulo 3, com informações de tempo e chamadas das funções.

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 28.23    146.15   146.15        1   146.15   348.65  MAIN__
 14.41    220.74    74.59 556322509     0.00     0.00  swtr8a_
 14.15    293.98    73.25 524120701     0.00     0.00  diffusel_
 12.43    358.30    64.32                             __libm_sse2_sincos
  5.51    386.82    28.52                             __libm_acos_l9
  5.41    414.81    27.99                             __libm_tan_l9
  3.28    431.82    17.01 578045892     0.00     0.00  astro_
  2.95    447.11    15.29                             ncx_putn_int_short
  2.67    460.95    13.84                             __libm_cos_l9
```

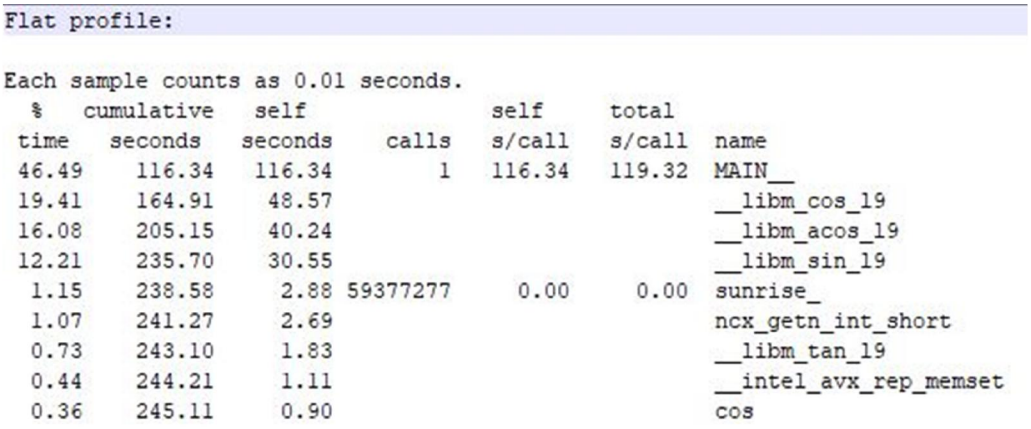
Figura 5.4 – Saída do GPROF do módulo 4, com informações de tempo e chamadas das funções.

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 24.77     93.68    93.68        1    93.68   216.83  MAIN__
 18.83    164.89    71.21                             __libm_sse2_sincos
 18.40    234.48    69.59 517002553     0.00     0.00  swtr8a_
  8.25    265.68    31.20                             __libm_tan_l9
  8.06    296.18    30.50                             __libm_acos_l9
  4.99    315.06    18.88 556049422     0.00     0.00  astro_
  3.92    329.87    14.81 550866731     0.00     0.00  sunrise_
  3.82    344.33    14.46                             __libm_cos_l9
  2.76    354.76    10.43 536503765     0.00     0.00  glod8a_
  2.50    364.20     9.45       638     0.01     0.01  movebites_
  1.07    368.26     4.06                             ncx_putn_int_short
```

Figura 5.5 – Saída do GPROF do módulo 6, com informações de tempo e chamadas das funções.

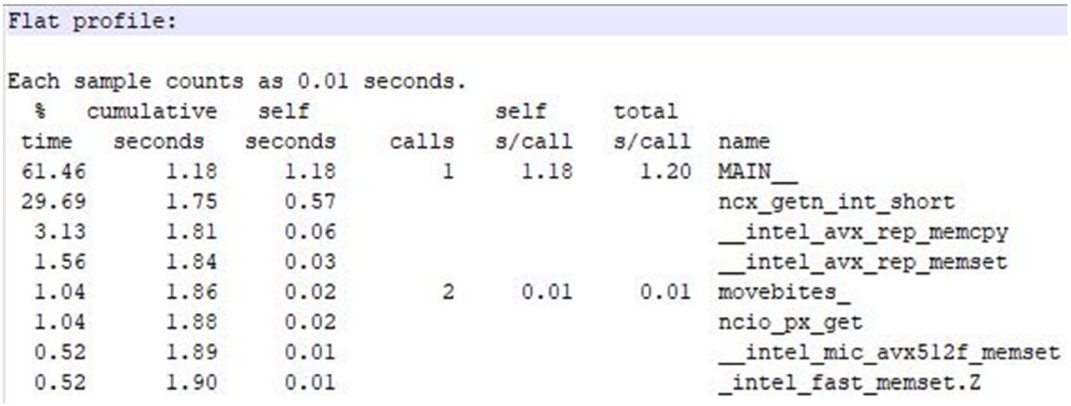
```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 19.45     59.98    59.98                             __libm_acos_l9
 18.46    116.89    56.91        1    56.91   100.91  MAIN__
 13.35    158.06    41.17                             __libm_cos_l9
 11.16    192.48    34.42                             __libm_sse2_sincos
  8.08    217.40    24.92 408904004     0.00     0.00  tilt_
  7.23    239.69    22.29                             __libm_tan_l9
  6.14    258.63    18.94 1406220160     0.00     0.00  sunrise_
  5.02    274.10    15.47                             __libm_pow_l9
```

Figura 5.6 – Saída do GPROF do módulo 5, com informações de tempo e chamadas das funções.



```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 46.49    116.34   116.34        1   116.34   119.32  MAIN__
 19.41    164.91    48.57                             __libm_cos_l9
 16.08    205.15    40.24                             __libm_acos_l9
 12.21    235.70    30.55                             __libm_sin_l9
  1.15    238.58     2.88 59377277     0.00     0.00  sunrise_
  1.07    241.27     2.69                             ncx_getn_int_short
  0.73    243.10     1.83                             __libm_tan_l9
  0.44    244.21     1.11                             __intel_avx_rep_memset
  0.36    245.11     0.90                             cos
```

Figura 5.7 – Saída do GPROF do módulo 7, com informações de tempo e chamadas das funções.



```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 61.46      1.18     1.18        1     1.18     1.20  MAIN__
 29.69      1.75     0.57                             ncx_getn_int_short
  3.13      1.81     0.06                             __intel_avx_rep_memcpy
  1.56      1.84     0.03                             __intel_avx_rep_memset
  1.04      1.86     0.02        2     0.01     0.01  movebites_
  1.04      1.88     0.02                             ncio_px_get
  0.52      1.89     0.01                             __intel_mic_avx512f_memset
  0.52      1.90     0.01                             _intel_fast_memset.Z
```

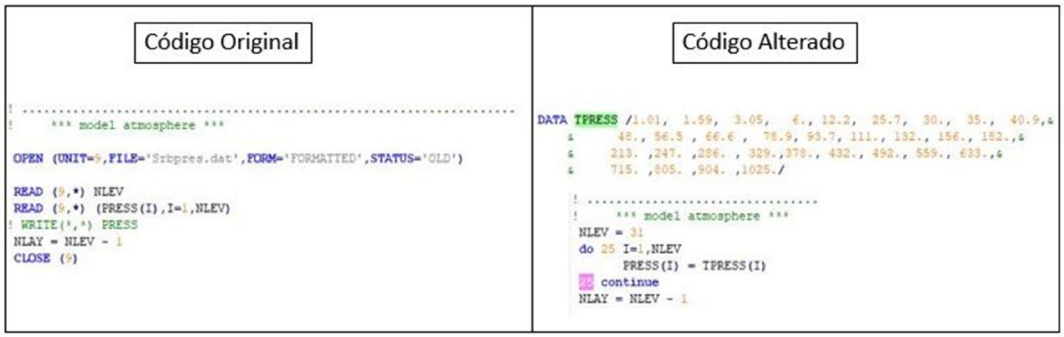
5.2. **OpenMP**

A inserção das diretivas de OpenMP foi realizada nos laços que mais consumiam tempo de processamento, como mostrado pelo *Gprof.* Com a paralelização destes laços que ficam mais interno ao código, não foi observado melhora no tempo de processamento. Assim, foi necessário mudar a estratégia e alterar o local onde foram inseridas as diretivas de OpenMP. Inserindo as

diretivas nos laços que chamam as funções mais internas, o modelo apresentou uma melhora bem pequena. Com isso, as diretivas começaram a serem inseridas nos laços mais externo, até que se encontrou o laço principal que se mostrou ideal para a paralelização.

Com a inserção da diretiva de OpenMP nos laços principais, uma região do código apresentou problema. Um trecho do código realizava a leitura de um arquivo texto, com dados de níveis de pressão, em cada pixel da imagem, Figura 5.8. Foi testada a inserção da diretiva *!$omp critical*, porém, com a diretiva, o código estava ficando com o tempo de processamento muito semelhante ao tempo serial. A solução foi alocar o conteúdo do arquivo texto em uma variável do tipo vetor e a passar por parâmetro para as funções.

Figura 5.8 – Trecho do código que realiza a leitura de um arquivo para cada pixel.



Após realizar a alteração do código, como citado acima, foi iniciado a execução do modelo a fim de validar as saídas. O modelo foi executado na forma atual e com as diretivas de OpenMP e utilizando apenas uma *thread*. Foi desenvolvido um código para calcular a diferença entre duas simulações realizadas com e sem processamento paralelo, tendo sido observadas diferenças entre elas. Foram encontrados valores que variavam em função das características da nebulosidade na região simulada. Na Figura 5.9, o gráfico exibe as diferenças da irradiação direta variando de -15 até 15 Wh/m². Entretanto, mais de 98% desta diferença se encontra próximo aos valores de -3 até 3 Wh/m².

Nas Figuras 5.10 e 5.11, podemos observar o gráfico com as diferenças das irradiações difusa e global. Como na irradiação direta, o comportamento do gráfico ficou muito parecido, mudando apenas a variação que ficou entre -34 até 31 Wh/m² para a difusa e -35 até 30 Wh/m² para a global. Como na diferença da irradiação direta, a difusa e global também tem os seus valores próximos a -3 e 3, representando, mais de 91% dos valores para ambas as irradiações, dentro desta faixa de diferença. A média da irradiação global no Brasil é de 4550Wh/m², se comparado a diferença encontrada, representa menos de 1%, não representando diferenças significativas para as simulações do modelo.

Figura 5.9 - Mapa da diferença entre a irradiação direta no modelo original com o modelo com diretivas em OpenMP.

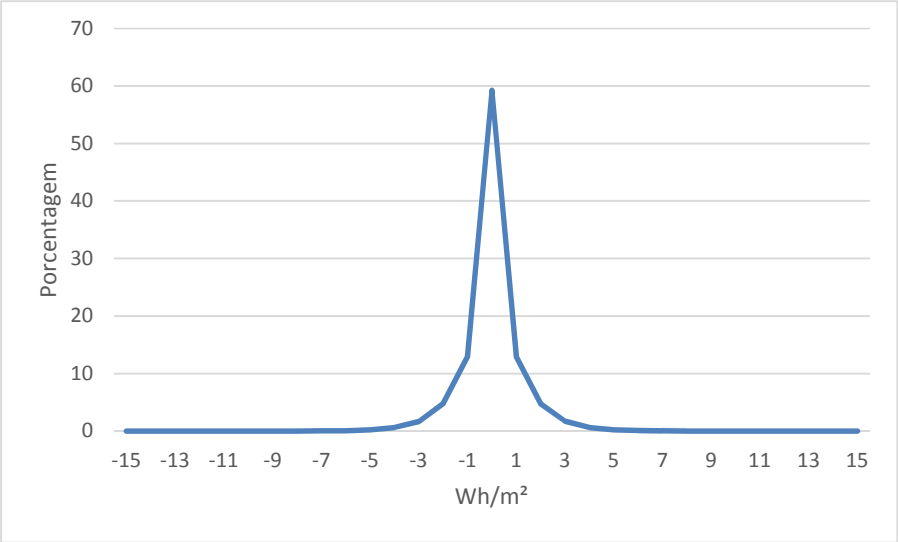


Figura 5.10 – Mapa da diferença entre a irradiação difusa no modelo original com o modelo com diretivas em OpenMP.

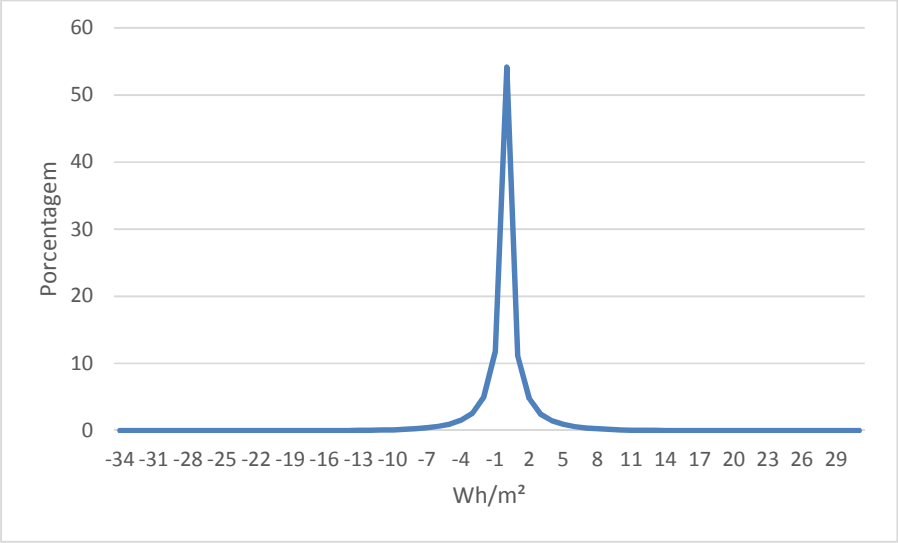
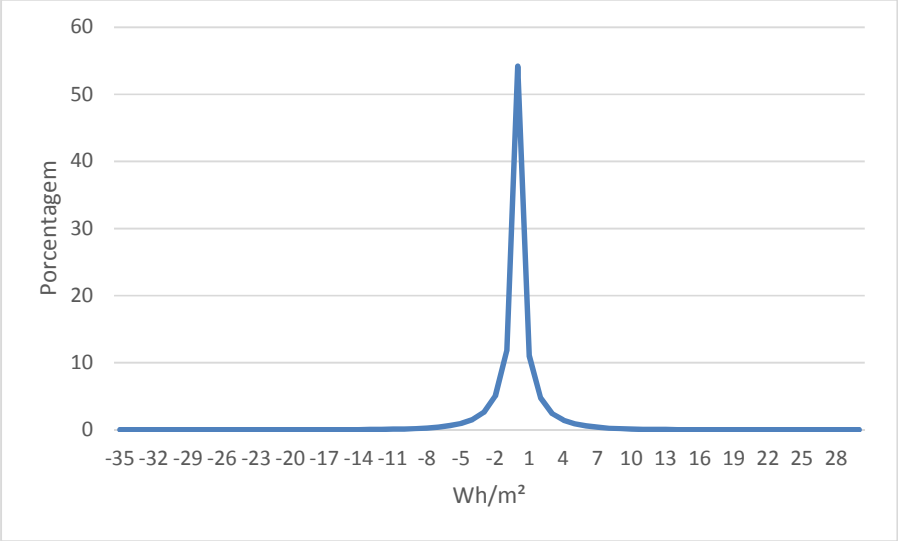


Figura 5.11 – Mapa da diferença entre a irradiação global no modelo original com o modelo com diretivas em OpenMP.

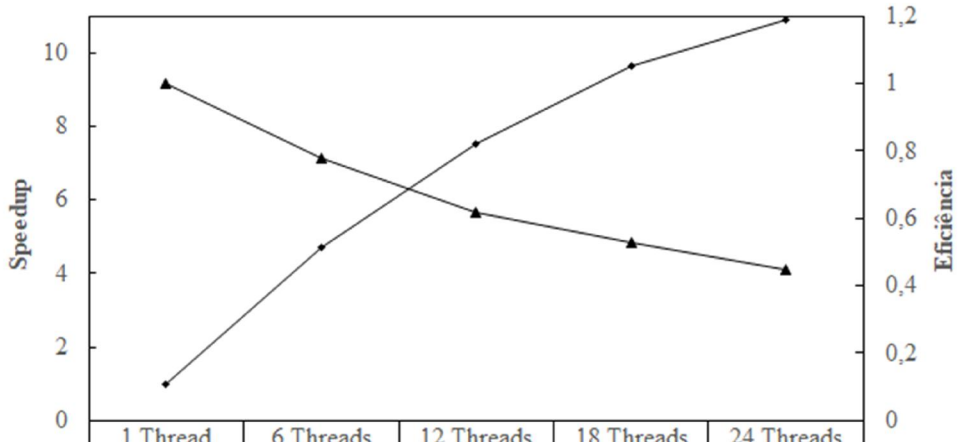


As diferenças nos valores das simulações mostradas acima foram verificados em rodadas do modelo com o código compilado com diretivas de OpenMP, isto

é, quando a *flag –qopenmp*, que habilita a paralelização multi *thread* do OpenMP era inserida na compilação do modelo. Vários testes foram realizados a fim de tentar resolver o problema das diferenças. O primeiro teste foi realizado com o modelo com diretivas de OpenMP utilizando 2 *threads* e o seu resultado foi a mesma matriz de diferenças. O mesmo código foi executado com apenas 1 *thread*, genrando a mesma matriz de diferenças. Quando o modelo com as diretivas de OpenMP foi compilado sem usar a *flag –qopenmp*, o modelo gerou um resultado igual ao resultado do modelo original. Descobrindo que o problema estava na diretiva de compilação e verificando que a diferença representava menos de 1% em relação ao modelo original, concluimos que não houve prejuízo qualitativo para o resultado final do modelo com OpenMP, sem diferenças significativas nas simulações.

Após resolver os problemas acima, iniciou-se os testes para a redução do tempo de processamento do modelo. Os testes foram realizados no supercomputador Santos Dumont utilizando 1, 6, 12, 18 e 24 *threads*. Na Figura 5.12, podemos observar que o tempo de processamento melhorou à medida que a quantidade de *threads* foi aumentando no modelo BRASIL-SR com OpenMP. Com um ganho de desempenho máximo de 10.9 e uma eficiência (0.45), nota-se que o modelo não tem uma eficiência paralela muito boa. Isto é devido, em parte, à influência dos trechos seriais originais do modelo e que não puderam ser cobertos pelas diretivas de OpenMP, e assim, limitaram o ganho de desempenho da paralelização, ou ainda aos efeitos dos processos de leitura/escrita de dados os quais também são inerentemente sequenciais.

Figura 5.12 – Tempo (segundos), ganho de desempenho e eficiência do modelo.



	1 Thread	6 Threads	12 Threads	18 Threads	24 Threads
Tempo (s)	96637	20540	12811	10036	8863
Speedup	1	4,7	7,54	9,63	10,9
Eficiência	1	0,78	0,62	0,53	0,45

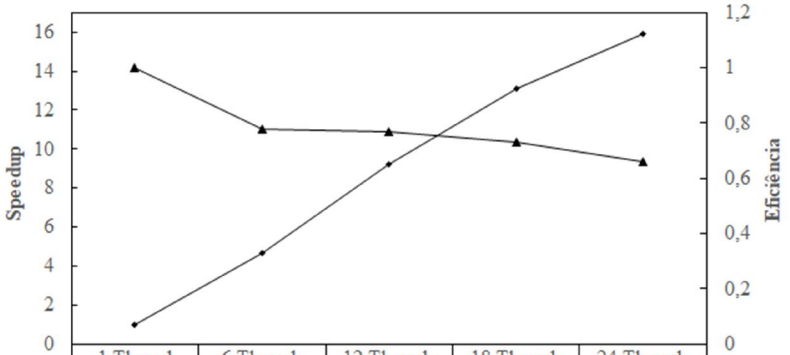
5.3. **OpenMP com arquivos NetCDF**

Como os arquivos de E/S estão em formatos binário e texto, fazendo com que a leitura e escrita consumam um tempo de processamento muito grande, foi realizada a conversão desses arquivos para NetCDF. Assim, foi melhorado o tempo de processamento e facilitada a visualização dos dados dos arquivos, já que os arquivos em NetCDF contém um dataset com informações de dimensão, variáveis e atributos, todas possuindo um nome e um número de identificação pelos quais podem ser referenciadas.

Realizando as alterações de todos os dados de E/S do modelo de binário e texto para NetCDF e alterando o modelo de forma que passe a ler este tipo de arquivos, o tempo total de processamento foi reduzido. Isto é uma consequência direta da melhora de desempenho na parte de E/S, produzida pelo uso do formato NetCDF. Na Figura 5.13, podemos observar a melhora no desempenho e a eficiência, até 12 *threads*, se mantendo constante. A eficiência e o ganho de desempenho máximo com 24 *threads* melhoraram em

relação ao modelo em sua forma atual, na qual todos os dados de E/S estavam em formato binário. Isto ocorre devido à Lei de Amdahl, pois ao usar o NetCDF a fração inerentemente sequencial do programa (onde está a parte de E/S) foi reduzida, permitindo, portanto, um maior ganho de desempenho máximo paralelo.

Figura 5.13 – Tempo (segundos), ganho de desempenho e eficiência do modelo.



	1 Thread	6 Threads	12 Threads	18 Threads	24 Threads
Tempo (s)	90700	19469	9808	6935	5703
Speedup	1	4,66	9,25	13,08	15,9
Eficiência	1	0,78	0,77	0,73	0,66

5.4. **Escalonamento de laços**

Para tentar reduzir ainda mais o tempo de processamento foram testados diferentes tipos de escalonamento dos laços paralelizados com OpenMP, utilizando as opções de *static* e *dynamic* e com tamanho fixo do passo igual a 1. O modelo foi executado utilizando 24 *threads* na versão com OpenMP e com arquivos de E/S em formato NetCDF. Nesta configuração, o modelo teve uma melhora de aproximadamente 5% em relação ao modelo sem o uso de escalonamento dos laços, como pode ser observado na Figura 5.14.

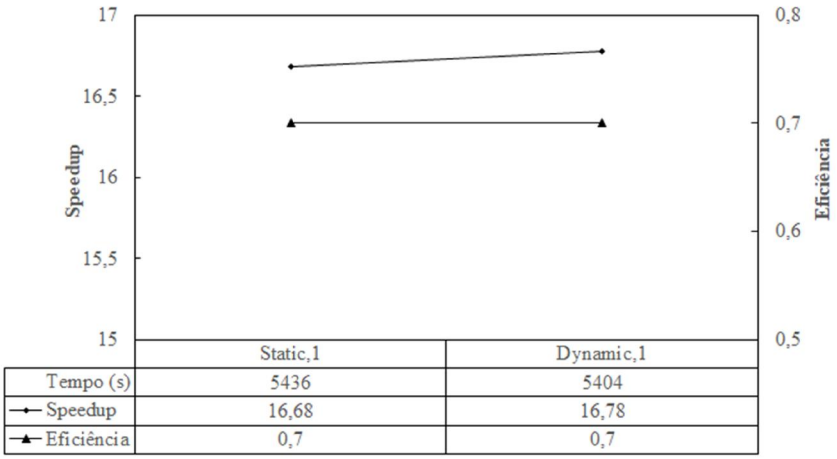
A melhora no tempo de processamento com escalonamento acontece porque quando inserido a diretiva de OpenMP *schedule(dynamic,1)* no código, cada *thread* receberá um índice para ser calculado e quando uma das *threads* terminar de calcular, essa *thread* receberá outro índice, sem a necessidade de

todas as *threads* terem finalizado suas tarefas, realizando assim, o escalonamento dinâmico.

No escalonamento estático, cada *thread* receberá um índice, e quando todas as *threads* terminarem suas tarefas, novos índices serão passados para todas as *threads* novamente. O OpenMP por padrão realiza o escalonamento dos laços a fim de ter uma melhora no tempo de processamento. Porém, sua divisão de tarefas pode fazer com que o código possa ter um desbalanceamento de carga, que é quando uma *thread* tem mais trabalho enquanto outra *thread* fique em modo de espera por outro trabalho. Desta forma, cada *thread* recebe uma quantidade igual de índices para ser calcula, por exemplo, se uma matriz possuir 100 colunas e utilizarmos 4 *threads* para o processamento, cada *thread* receberá 25 colunas para trabalhar. Se uma região desta matriz precisar de mais tempo de processamento, isso implica em uma *thread* trabalhar mais que outras, causando assim o desbalanceamento.

Como as imagens utilizadas pelo modelo são de toda América do Sul, incluindo os dois oceanos pacífico e atlântico, e o modelo não realizando cálculos sobre os oceanos, quando uma *thread* fica com uma coluna com muitos pontos sobre o oceano e com poucos pontos sobre a superfície, e uma outra *thread* com muitos dados na superfície e poucos pontos sobre o oceano, acontece um desbalanceamento de carga. Esse desbalanceamento é reduzido quando se utiliza o escalonamento estático e dinâmico, e isso é mostrado com o melhor tempo de processamento do modelo BRASIL-SR utilizando o escalonamento dinâmico (*dynamic).*

Figura 5.14 – Tempo (segundos), ganho de desempenho e eficiência do modelo com escalonamento de laços.



	Static,1	Dynamic,1
Tempo (s)	5436	5404
Speedup	16,68	16,78
Eficiência	0,7	0,7

5.5. **Técnica de blocagem**

Com a técnica de blocagem foram realizados testes com apenas um horário do primeiro módulo mostrando que o tempo de processamento se manteve constante. Utilizando os contadores (*PAPI*) para verificar como as matrizes estavam sendo armazenadas e acessadas nas memórias, foi observado que mesmo variando o tamanho dos blocos (4x4, 8x8 e 32x32) todos os contadores se mantiveram com valores praticamente constantes, como pode ser observado na Tabela 5.1.

Tabela 5.1 - Valores dos contadores e tempo de processamento (para um horário) com Blocagem

	4X4	8X8	32X32
PAPI_LD_INS	2403140908105	2403148275376	2403958401927
PAPI_SR_INS	719745845823	719745476559	719982428741
PAPI_L1_TCM	19860626832	19816497544	19644044028
PAPI_L2_TCM	1644209030	1390039100	1575671613
PAPI_L3_TCM	817986	835012	857596
TEMPO (s)	3040	3037	3034

A constância no tempo de processamento, observado na tabela 5.1, com a técnica de blocagem e nos valores dos contadores de *hardware* pode ser explicada pela forma dos laços empregados no modelo BRASIL-SR, tal como no laço da Figura 3.8, cada ponto (I,J) do domínio é visitado uma única vez, com ou sem blocagem. Desta forma, não existe potencial para ganhos devido ao uso de cache com a reutilização dos valores do ponto (I,J). Assim, o método de blocagem não obteve a melhora no desempenho pretendida.

5.6. **Técnica de vetorização**

Os testes com a técnica de vetorização foram realizados utilizando apenas um horário do primeiro módulo. Inicialmente, foi utilizada a *flag –qopt-report=6,* que é disponível no Intel *ifort*, para gerar um relatório onde são mostrados os laços vetorizados e não vetorizados. Abaixo, na Figura 5.15, pode-se observar um trecho do relatório onde é mostrado que o laço na linha 121, da função *Strpsrb.f90*, foi verificado e tem a capacidade de ser vetorizado pelo compilador.

Figura 5.15 - Relatório dos laços que podem ser vetorizados pelo compilador.

```
LOOP BEGIN at Strpsrb.f90(121,5)
   remark #15388: vectorization support: reference index(i) has aligned access   [ Strpsrb.f90(122,8) ]
   remark #15305: vectorization support: vector length 4
   remark #15399: vectorization support: unroll factor set to 5
   remark #15309: vectorization support: normalized vectorization overhead 0.067
   remark #15300: LOOP WAS VECTORIZED
   remark #15449: unmasked aligned unit stride stores: 1
   remark #15475: --- begin vector cost summary ---
   remark #15476: scalar cost: 2
   remark #15477: vector cost: 0.750
   remark #15478: estimated potential speedup: 2.190
   remark #15488: --- end vector cost summary ---
   remark #25015: Estimate of max trip count of loop=1
LOOP END
```

Como mencionado, foram utilizadas as *flag* de vetorização e não vetorização (*–xAVX* e *–no-vec*) no modelo com as alterações nos dados de E/S em formato NetCDF, estes resultados podem ser observados na Tabela 5.2.

Tabela 5.2 – Tabela com os tempos de vetorização do Modelo BRASIL-SR.

	TEMPO (s)
Modelo Atual Vetorizado	6797
Modelo Atual Não Vetorizado	6837
Modelo NetCDF Vetorizado	3208
Modelo NetCDF Não Vetorizado	3247

Como na técnica de blocagem, os resultados com a vetorização não mostraram uma melhora do tempo de processamento. Isso ocorre devido a vários laços do modelo conterem comandos *goto*, os quais inibem vetorização. Com os desvios condicionais realizados nos laços (representados pelo *goto)*, o processador não vetoriza este trecho, e como estes trechos são muito custosos computacionalmente, o tempo de processamento não apresenta melhora.

5.7. **Operacionalização do modelo BRASIL-SR**

O script desenvolvido para executar o pré-processamento e o processamento do modelo BRASIL-SR, permitindo que o modelo se torne operacional, mostrou-se muito eficiente. Se o *script* for separado em pré-processamento e processamento do modelo, o tempo obtido no processamento, utilizando 24 *threads* e com os dados de entrada no formata em NetCDF e utilizando escalonamento, foi de 1 hora e 33 minutos e no pré-processamento foi de 6 minutos e 30 segundos. Se compararmos o tempo de processamento do modelo original executado com 24 *threads* (2 horas e 27 minutos) com o tempo de processamento do modelo operacional com 24 *threads* (1 hora e 33

minutos), temos uma melhora de 36% no tempo do modelo em modo diário. Se a comparação for realizada no pré-processamento, a melhora é ainda maior. O pré-processamento realizado de forma manual consumia aproximadamente 4 horas para ser realizado, já o pré-processamento do modelo operacional leva 6 minutos e 30 segundos, uma melhora de 3692%. Esta melhora não é apenas pela utilização do *script,* mas também pela conversão de alguns códigos escritos em *Matlab* para Fortran 90.

No processamento, a transmitância nos módulos 1 e 2, que antes era calculada para cada horário e apenas para o 15º dia do mês e o valor passado para todos os outros dias e horários, agora é calculada apenas para o dia atual da execução. Assim, a transmitância não apresentará uma melhora no tempo de processamento, já que este cálculo continuará sendo realizado, para todos os horários do dia atual, seu tempo de execução se manteve o mesmo. Porém, nos módulos 3, 4, 5 e 6 a quantidade de arquivos para serem processados (E/S) são menores, já que a execução leva em consideração apenas os dados de apenas um dia, tendo assim uma melhora no tempo de processamento.

A transmitância é originalmente calculada para ser utilizada com campos de nebulosidades estimados utilizando dados climatológicos de temperatura, umidade, albedo, visibilidade e altitude, uma alteração no modelo precisou ser realizada para que ele passasse e realizar a leitura de dados meteorológicos oriundos de modelos, de acordo com o horário da imagem.

Os dados meteorológicos (temperatura, albedo, umidade) eram médias de 1999 até 2015, agora são dados de análise do dia as 6, 12 e 18 UTC do modelo *Global Forecast System (GFS)* disponibilizados pelo *National Center for Environmental Prediction (NCEP)*. Quando realizado o procedimento para baixar os dados meteorológicos, além das três variáveis que foram utilizadas, várias outras estão no mesmo arquivo, sendo necessário extrair apenas as que serão utilizadas pelo modelo (temperatura, umidade e albedo). Os dados contêm resolução espacial de 0.25 por 0.25 graus em latitude e longitude

necessitando assim serem interpolados para ficarem na mesma grade dos dados de entrada do modelo (0.03 por 0.05) graus. Todas estas operações foram realizadas utilizando o *script* com os comandos do CDO, na Figura 5.16, podemos observar o *script* desenvolvido para converter os dados climatológicos, extrair as variáveis, recortar para a grade desejada e salvar no formato NetCDF.

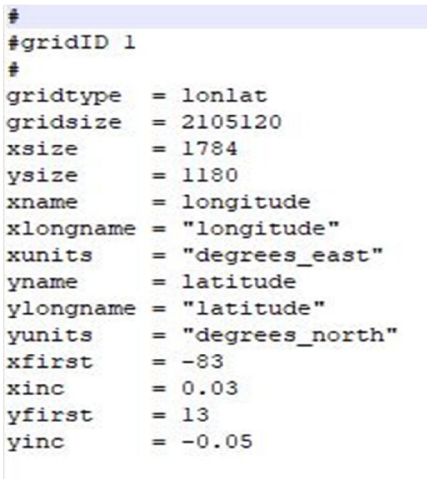
Figura 5.16 – Comandos do *CDO* para extrair e interpolar os dados

```
#!/bin/bash
for file in *.grib2; do
   /opt/cdo/bin/cdo -f nc -chname,2t,t -sellonlatbox,-85,-27.51,-47.95,15 -select,code=-26,-29,-131 $file ${file%.*}.nc
   echo $file
done

for file in *.nc; do
   /opt/cdo/bin/cdo remapbil,mygrid $file ${file%.*}Int.nc
   echo $file
done
```

No primeiro laço são listados todos os arquivos com extensão *.grib2* e posteriormente é utilizada a função *–f nc* para converter o arquivo de *grib* para NetCDF. A função *–chname* é utilizada para renomear a variável de *2t* para *t,*sendo que o NetCDF não entende variáveis começadas com numeral. A função *–sellonlatbox* é utilizado para selecionar a área que será recortada da imagem e *–select,code* para selecionar quais variáveis serão extraídas do arquivo.

No segundo laço é utilizada a função *–remapbil* para realizar a interpolação bilinear. As informações de latitude e longitude, tamanho da grade e resolução da grade são encontradas no arquivo *mygrid,* Figura 5.17.

Figura 5.17 – Arquivo com informações para interpolação no CDO.



```
#
#gridID 1
#
gridtype  = lonlat
gridsize  = 2105120
xsize     = 1784
ysize     = 1180
xname     = longitude
xlongname = "longitude"
xunits    = "degrees_east"
yname     = latitude
ylongname = "latitude"
yunits    = "degrees_north"
xfirst    = -83
xinc      = 0.03
yfirst    = 13
yinc      = -0.05
```

No pré-processamento as etapas (funções) que realizam as copias das imagens de satélite e de climatologia, dependem da rede local e da internet, fazendo com que o tempo de pré-processamento possa sofrer alterações. Outra etapa que também tem uma dependência e que pode variar o tempo do pré-processamento é o cálculo da máxima e mínima cobertura de nuvem, o qual dependendo da quantidade de arquivos (dias ou meses), pode levar um tempo de processamento maior. Para este trabalho foi utilizado 31 dias de imagens de satélite, onde a partir do dia a ser processado, é obtido 15 dias antes e depois. Posteriormente, será verificado qual a quantidade de imagens ideal para este tipo de variável.

6 DISCUSSÃO

O paradigma de paralelização OpenMP foi escolhido principalmente para permitir aos usuários do modelo a exploração de processadores *multi-core*, amplamente disponíveis até mesmo em computadores pessoais. Com isto, a utilização do modelo BRASIL-SR poderia atender satisfatoriamente aos requisitos operacionais de tempo de execução hoje existentes. A extensão deste trabalho para ambientes de memória distribuída, via paralelização com *MPI*, é uma possibilidade para trabalhos futuros.

Antes de começar a paralelizar o modelo BRASIL-SR, foi verificado se existia alguma dependência entre as iterações dos laços. A dependência acontece quando uma iteração com um certo índice necessita de alguma informação gerada por uma iteração com um índice próximo (vizinhos). Esta verificação foi realizada executando o modelo de forma invertida, ou seja, executando do último índice para o primeiro. Realizado o teste e não encontrando dependência, foi iniciado o processo de paralelização com as diretivas de OpenMP.

A paralelização do modelo BRASIL-SR, utilizando diretivas de OpenMP, mostrou problemas iniciais que não eram esperados. Como em um trecho do código que cada vez que um pixel fosse lido, uma leitura de um arquivo texto era realizada. Esse problema foi inicialmente resolvido com a inserção da diretiva *critical*, onde apenas um *thread* por vez pode entrar no trecho indicado pelas diretivas de OpenMP. Esta alteração fez com que o modelo ficasse com o mesmo tempo de processamento do modelo atual, ou seja, ele ficou serial. Para resolver o problema foi então utilizada a diretiva *master*, que faz com que apenas a *thread master* execute o trecho. Mesmo com essa alteração o tempo de processamento ficou alto (17 horas e 40 minutos) e um ganho de desempenho máximo de 1.5. Com o tempo de processamento não se mostrando ideal, o trecho do código foi alterado. Ao invés de realizar a leitura

do arquivo, seu conteúdo foi armazenado em um vetor e este vetor foi passado por parâmetro para a função que estava realizando a leitura.

Outro problema encontrado foi o das pequenas diferenças entre os valores de saída do modelo atual e do modelo com as diretivas de OpenMP. A simples inserção da *flag* do OpenMP (*-qopenmp*) na linha de compilação foi o causador do problema. Em todos os testes realizados, com números de *threads* variados, o problema com as diferenças estava sempre presente e quando não era utilizada a *flag* o problema com a diferença deixava de existir. O modelo foi testado exaustivamente a fim de encontrar qual a variável poderia estar causando o problema com o OpenMP. Quase todas as variáveis existentes no modelo foram impressas e comparadas com o modelo original, e quando alguma diferença era observada ela estava, por exemplo, na décima casa decimal. Estas diferenças mostram que mais de 90% destes valores estão na casa entre -3 e 3 Wh/m² representando menos de 1% da média de irradiação do dia, que fica entorno de 4500 Wh/m².

O modelo foi testado também com o compilador *gfortran* (GNU Fortran), para verificar se o problema poderia estar no compilador da Intel (*ifort*). O modelo foi executado na versão atual e na versão com OpenMP, compilada com a *flag* –fopenmp (*flag* do *gfortran* para o OpenMP). Estas duas execuções também apresentaram resultados de saída ligeiramente distintos. Desta forma, acredita-se que o problema não seja exclusivo do compilador Intel, e possivelmente seja causado pelo uso da biblioteca OpenMP. De qualquer forma, foi também verificado que no caso de *gfortran*, as diferenças de valores encontradas estavam abaixo de 1%.

Mesmo com os problemas iniciais encontrados, o modelo utilizando diretivas de OpenMP mostrou uma redução considerável no seu tempo de processamento com 24 *threads,* chegando a 2 horas e 27 minutos, correspondendo a um ganho de desempenho máximo de 10.9. A implementação do modelo utilizando OpenMP relativa às operações de E/S em formato NetCDF mostrou uma

melhora adicional, com uma melhor eficiência paralela. O tempo de execução utilizando 24 *threads* foi de 1 hora e 35 minutos, ou seja, uma melhora de 35% em relação ao modelo atual, sem alterações. Realizando os testes com o modelo com os dados em formato NetCDF e utilizando a opção de escalonamento de laços *Dynamic* com 24 *threads*, o modelo foi executado em 1 hora e 30 minutos, melhorando em 5,2% em relação ao melhor tempo de processamento do modelo com dados em NetCDF. Se comparado com o modelo atual, a execução com o escalonamento *Dynamic* apresentou uma melhora no tempo computacional de 39%.

A utilização das diretivas de OpenMP com escalonamento e utilizando os dados de E/S em formato de NetCDF, mostrou uma melhora bem significativa no tempo de processamento do modelo BRASIL-SR possibilitando sua execução de forma diária.

Nos testes realizados com blocagem, em que não obtivemos uma melhora no tempo de processamento, foram verificadas as taxas de acerto/erro em *cache*, de modo a caracterizar precisamente quais valores de *stride* seriam mais vantajosos. Observamos que as *caches* têm aproximadamente o mesmo número de acertos/erros e os tempos de processamento ficaram muito próximos, mostrando que a técnica, para o modelo BRASIL-SR, não teve um efeito de redução de tempo mesmo alterando o tamanho de bloco. Isto ocorreu devido à estrutura dos loops do modelo BRASIL-SR.

Na técnica de vetorização também não obtivemos uma melhora no tempo de processamento. Os tempos de processamento com e sem vetorização foram muito próximos. A ausência de melhora se deve ao fato de que o modelo contém muitos laços com comandos do tipo *goto*. Todos os trechos de código do modelo com este tipo de laço não foram vetorizados, fazendo com que estes trechos do código ficassem em modo sequencial.

Para ambas as técnicas uma possível solução seria realizar a conversão de todos os laços com comandos *goto* para laços simples do Fortran como o laço

do. Neste trabalho isso não foi realizado pois envolveria uma alteração muito grande no código do modelo BRASIL-SR e seria necessário refazer todos os testes realizados anteriormente para que fosse possível realizar uma comparação de resultados de forma igual para todos os testes realizados.

O *script* desenvolvido para a execução do pré e processamento juntamente com o modelo BRASIL-SR alterado para ser executado para um dia, se mostrou muito eficiente e com um bom tempo de processamento se comparado com o modelo atual e com o pré-processamento realizado de forma manual. Além do script desenvolvido, foi também convertido os códigos de linguagem *Matlab* para *Fortran* 90, colaborando ainda mais com a redução no tempo de pré-processamento do modelo.

O *script* de pré-processamento está auxiliando não apenas nas execuções dos códigos para obter imagens de satélite ou climatologia, mas também em todo o processo de criação de diretórios necessários para o modelo, arquivos texto contendo listas de imagens e horários que serão utilizados tanto na criação dos dados de entrada para o modelo, como também para que o modelo possa saber quais os dados serão utilizados na sua execução.

Com o *script* e o modelo alterado é possível executar as transmitâncias do modelo diariamente e a partir do momento que uma imagem de satélite é disponibilizada já é possível calcular as irradiações horarias e utilizar estes dados de saída em modelos de previsão de curto prazo. O *script*, está fazendo a utilização de imagens de satélite de um banco de dados e não diretamente da base de dados da Divisão de Satélites e Sistemas Ambientais (DSA), pois houve mudança de satélite em 2017 (GOES 13 saiu e entrou o GOES-R) e com esses novos dados o modelo precisara passar por alterações no código, para ler imagens com resolução espacial maior e também para utilização de algum novo produto gerado pelo satélite.

Com o desenvolvimento do *script,* tarefas realizadas de forma manual como baixar as imagens de satélites, validar essas imagens, baixar os dados de

climatologia e calcular a cobertura efetiva de nuvens, são realizadas de forma automática pelo *script*. Com isso, o tempo que seria perdido tendo uma pessoa realizando todos esses passos e ainda criando as listas de arquivos necessárias para a execução do modelo, pode ser agora utilizado para outras tarefas de pesquisas.

7 CONCLUSÕES

Neste trabalho foram apresentados os resultados da paralelização do modelo BRASIL-SR, que calcula as estimativas da radiação solar incidente na superfície, através da paralelização com OpenMP e suas diretivas e de técnicas de otimização de desempenho.

Apenas com a inserção das diretivas de OpenMP o modelo BRASIL-SR mostrou uma redução no tempo de processamento de 90.8% em relação ao modelo atual, e foi melhorado quando alterados os dados de E/S para o formato em NetCDF. Quando realizado o escalonamento dos laços no modelo com OpenMP e com os aquivos de E/S em NetCDF, o melhor tempo de processamento foi obtido 1 hora e 30 minutos com a opção de escalonamento *Dynamic* e utilizando 24 threads, representando uma redução no tempo de processamento, quando comparado com o modelo atual, de 94%, ganho de desempenho máximo de 17.88 e uma eficiência de 0.75.

As técnicas de blocagem e vetorização não apresentaram melhora no tempo de processamento do modelo. Como cada ponto (i,j) do domínio é visitado uma única vez, com ou sem blocagem, fez com que a blocagem não tivesse potencial para ganhos devido ao uso de cahce com a reutilização dos valores dos pontos (i,j). Como o modelo possui vários laços com o comando *goto*, a vetorização acaba sendo inibida já que este tipo de comando faz com que a região a ser vetorizada se torne serial.

Mesmo tratando-se de um modelo escrito na década de 90, a redução do tempo de processamento foi muito boa e a criação do script operacional de execução será muito útil já que não será mais necessário que uma pessoa fique criando o ambiente, arquivos e executando os códigos de pré e processamento. Se for realizada a conversão dos laços contendo *goto* para laços regulares, o modelo deve apresentar resultados ainda melhores, e ficar

com seu código escrito para trabalhar com arquiteturas de processadores mais atuais, as quais suportam vetores cada vez maiores.

Uma paralelização mais ampla, para ambientes de memória distribuída, através do uso de um paradigma de troca de mensagens (por exemplo com *MPI*), é uma possibilidade viável para desenvolvimentos futuros.

O modelo BRASIL-SR paralelizado com as diretivas de OpenMP e modificado para executar de forma diária, juntamente com o *script* desenvolvido em Python, possibilitou o modelo a se tornar operacional. O tempo do pré-processamento do modelo diário obteve uma melhora de 3700% comparado com o pré-processamento realizado de forma manual. O processamento do modelo diário obteve uma melhora de 36.5% em relação ao modelo com OpenMP escalonado e com os dados de E/S em NetCDF, sendo executado com 24 *threads.*

Inicialmente o modelo BRASIL-SR operacional será executado em um computador do tipo *workstation* utilizando processadores Intel Xeon de 2.4GhHz com 12 núcleos (*threads*), estando locado no LABREN/CCST/INPE. Posteriormente, será verificado a possibilidade de o modelo ser executado no supercomputador TUPÃ, locado no CPTEC/INPE. Com o modelo em modo diário, e com as novas metodologias utilizadas para os dados meteorológicos e para o cálculo dos índices da máxima e mínima cobertura de nuvens, espera-se que o viés do modelo melhore, já que esses dados irão representar exatamente a condição climática do dia a ser processado. Os dados gerados serão muito uteis para pesquisas na área de energia solar, como a utilização dos dados para realizar previsão de curto e curtíssimo prazo para empreendimentos solares, colaborando com a manutenção preventiva e operação do empreendimento, gerenciamento de despacho em linha de transmição e gestão de carga no sistema interligado pela ONS.

O modelo BRASIL-SR, precisa agora ser alterado para receber as imagens do novo satélite da família GOES (GOES-R) sendo que será necessário pesquisar

novas formas de otimização para o modelo já que as novas imagens do GOES-R têm resolução espacial melhor que a utilizada hoje pelo modelo. Uma alternativa será a remoção dos comandos *goto* para tentar utilizar melhor a vetorização no modelo.

REFERÊNCIAS BIBLIOGRÁFICAS

ASSOCIAÇÃO BRASILEIRA DE ENERGIA SOLAR FOTOVOLTAICA (ABSOLAR). **Por que querem impedir o crescimento da energia solar fotovoltaica.** Disponível em: http://www.absolar.org.br/noticia/artigos-da-absolar/por-que-querem-impedir-o-crescimento-da-energia-solar-fotovoltaica.html. Acesso em: 15 nov. 2018.

ARCGIS FOR DESKTOP. **Fundamentals of netCDF data storage.** Disponível em: http://desktop.arcgis.com/en/arcmap/10.3/manage-data/netcdf/fundamentals-of-netcdf-data-storage.htm. Acesso em: 06 jun. 2018

AGÊNCIA NACIONAL DE ENERGIA ELÉTRICA (ANEEL). **Atlas de energia electrica do Brasil**. Brasília. 2012. Disponível em: http://www2.aneel.gov.br/aplicacoes/atlas/download.htm. Acesso em: 02 fev. 2017.

AGÊNCIA NACIONAL DE ENERGIA ELÉTRICA (ANEEL) . **Download de dados.** Disponível em: https://sigel.aneel.gov.br/Down/. Acesso em: 03 dez. 2018.

AGÊNCIA NACIONAL DE ENERGIA ELÉTRICA (ANEEL). **BIG - Banco de Informações de Geração**. Disponível em: http://www2.aneel.gov.br/aplicacoes/capacidadebrasil/capacidadebrasil.cfm. Acesso em: 4 jan. 2017.

AGÊNCIA NACIONAL DE ENERGIA ELÉTRICA (ANEEL). **Programa de incentivo às fontes alternativas**. Disponível em: http://www.aneel.gov.br/proinfa. Acesso em: 4 jan. 2018.

ALMASI, G. S.; GOTTLIEB, J. G. **Highly parallel computing.** 2.ed. [S.l.]: University of Michigan, 1994. 689p. ISBN 0805304436.

AMRITKAR, A.; DEB, S.; TAFTI, D. Efficient parallel CFD-DEM simulations using OpenMP. **Journal of Computational Physics**, v.256, p. 501-519, 2014.

ANDRADE, G. L.; CERA, M. C. Avaliando a paralelização de um algoritmo genético com OpenMP. In: SIMPÓSIO EM SISTEMAS COMPUTACIONAIS DE ALTO DESEMPENHO, 17., 2017, Aracajú, SE. **Anais...** 2016.

ARTAXO, P.; OLIVEIRA, P. H.; LARA L. L.; PAULIQUEVIS, T. M.; RIZZO, L. V.; PIRES JUNIOR, C.; PAIXÃO, M. A.; LONGO, K. M.; FREITAS, S.; CORREIA, A. L. Efeitos climáticos de partículas de aerossóis biogênicos e emitidos em 198 queimadas na Amazônia. **Revista Brasileira de Meteorologia,** v.21, n.3, p.168-220, 2006.

BRAKMANN, G.; ARINGHOFF, R. **Solar thermal power**. [S.l.]: Greenpeace, 2003. ISBN 90-73361-82-6.

CERUZZI, P. E. **A history of modern computing**. 2.ed. [S.l.]: Techset Composition, 2003. 460 p. ISBN 0-262-53203-4.

CHAPMAN, B.; JOST, G.; PAS, R. V. D. **Using openMP**: portable shared memory parallel programming. [S.l.]: The MIT Press, 2008. ISBN- 978-0-262-53302-7.

DAVE, J. V.; CANOSA, Z. A direct solution of the radiative transfer equation: application to atmospheric models with arbitrary vertical nonhomogeneities. **Journal of Atmospheric Sciences,** v.31, p.1089-1101, 1974. DOI 10.1175/1520-0469.

DEIRMENDJIAN, D. **Electromagnetic scattering on spherical polydispersions**. New York: Elsevier, 1969. 318 p. ISBN 444-00038-0.

ECHER, E.; SOUSA, M. P.; SCHUCH, N. J. A lei de Beer aplicada na atmosfera terrestre. **Revista Brasileira de Física**, São Paulo, v.23, n.3, p.276-283, 2001.

EMPRESA DE PESQUISA ENERGÉTICA (EPE). **Anuário estatístico de energia elétrica 2011**. 2011. Disponível em: http://www.epe.gov.br/sites-pt/publicacoes-dados-abertos/publicacoes/PublicacoesArquivos/publicacao-160/topico-168/Anu%C3%A1rio%20Estat%C3%ADstico%20de%20Energia%20El%C3%A9trica%202011.pdf. Acesso em: 13 mar. 2017.

EMPRESA DE PESQUISA ENERGÉTICA (EPE). **Análise da inserção da geração solar na matriz energética brasileira**. 2012. Disponível em: http://www.epe.gov.br/geracao/Documents/Estudos_23/NT_EnergiaSolar_2012.pdf. Acesso em: 13 mar. 2017.

EMPRESA DE PESQUISA ENERGÉTICA (EPE). **Plano nacional de energia 2030**. 2006. Disponível em: http://www.epe.gov.br/PNE/20080111_1.pdf. Acesso em: 13 mar. 2017.

EMPRESA DE PESQUISA ENERGÉTICA (EPE). **Estudos da demanda de energia 2050**. jan. 2016. Disponível em: http://www.epe.gov.br/Estudos/Documents/DEA%2013-14%20Demanda%20de%20Energia%202050.pdf. Acesso em: 13. mar. 2017.

EMPRESA DE PESQUISA ENERGÉTICA (EPE). **Balanço energetico nacional**: ano base 2015. 2016. Disponível em: http://www.mme.gov.br/documents/10584/1143895/2.1+-+BEN+2015+-+Documento+Completo+em+Portugu%C3%AAs+-+Ingl%C3%AAs+%28PDF%29/22602d8c-a366-4d16-a15f-f29933e816ff?version=1.0. Acesso em: 13 mar. 2017.

EMPRESA DE PESQUISA ENERGÉTICA (EPE). **Expansão da geração:** 1 leilão de energia de reserva de 2015. 2015. Diponível em: http://www.epe.gov.br/leiloes/Documents/Leil%C3%A3o%20de%20Reserva%20(2015)/NT_EPE-DEE-NT-127_2015-r0_completo.pdf. Acesso em: 13 mar. 2017.

EMPRESA DE PESQUISA ENERGÉTICA (EPE). **Anuário estatístico de energia elétrica 2016**: ano base 2015. 2016. Disponível em: http://www.epe.gov.br/AnuarioEstatisticodeEnergiaEletrica/Anu%C3%A1rio%20Estat%C3%ADstico%20de%20Energia%20El%C3%A9trica%202016.xls. Acesso em: 13 mar. 2017.

EMPRESA DE PESQUISA ENERGÉTICA. **Relatório síntese ano base 2017**. 2017. Disponivel em: http://www.epe.gov.br/sites-pt/publicacoes-dados-abertos/publicacoes/PublicacoesArquivos/publicacao-303/topico-419/BEN2018__Int.pdf. Acesso em: 16 fev. 2019.

FLYNN, M.J. Some computer organizations and their effectiveness, **IEEE Transactions on Computer**, v.C-21, p.948-960, 1972.

FONSECA FILHO, C. **História da computação**. Porto Alegre: EDIPUCRS, 2007. 205p. ISBN 978-85-7430-691-9.

FREYTAG, G.; ARRUDA, G.; MARTINS, R. S. M.; PADOIN. E. L. Análise de desempenho da paralelização do problema do caixeiro viajante. In: ESCOLA REGIONAL DE ALTO DESEMPENHO DO ESTADO DO RIO GRANDE DO SUL, 15., 2015, Gramado. **Anais...** 2015. p. 157-160.

FORTRAN 90. **Fortran 90.** 2016. Disponível em: http://www.fortran90.org/index.html. Acesso em: 13 dez. 2016.

GAUTIER, C.; DIAK, G.; MASSE, S. A simple physical model to estimate incident solar radiation at the surface from GOES satellite data. **Journal of Applied Meteorology**, v. 19, p.1005-1012, 1980.

GEPNER, P.; KOWALIK, M. F. Multi-core processors: new way to achive high system performance. In: PARALLEL COMPUTING IN ELECTRICAL ENGINEERING, 2006, Bialystok, Poland. **Proceedings…** 2006. p. 9-13.

GOLDEMBERG, J.; LUCON O. **Energia, meio ambiente e desenvolvimento**. 3.ed. São Paulo: Universidade de São Paulo, 2008. 394 p. ISBN 978-85-314-1113-7.

HERMANNS, M. **Parallel programming in Fortran 95 using OpenMP**. Madri – Espanha: Universidade Politécnica de Madri, 2002. 75 p.

HILLIS, W.D. **The connection machine.** Massachusetts: MIT Press, 1986. ISBN 0-262-08157-1.

HUANG, L; CHAPMAN, B.; KENDALL, R. OpenMP for clusters. In: EUROPEAN WORKSHOP ON OPENMP, 50., 2003, Aachen, Germany. **Proceedings…** 2003. p. 22-26.

HULD, T.; MULLER, R.; GAMBARDELLA, A. A new solar radiation database for estimating PV performance in Europe and Africa, **Solar Energy**, v. 86, n. 6, p.1803-1815, 2012.

INFINIBAND TRADE ASSOCIATION. **About infiniband**. Disponível: https://www.infinibandta.org/. Acesso em: 18 mar. 2019.

INNOVATIVE COMPUTING LABORATORY (ICL). **PAPI**. 20 dez. 2017. Disponível: http://icl.cs.utk.edu/papi/. Acesso em: 22 nov. 2018.

INTERNATIONAL ENERGY AGENCY (IEA). **Energy, climate change and environment.** 2016. Disponível em: http://www.iea.org/publications/freepublications/publication/ECCE2016.pdf. Acesso em: 19 jan. 2018.

INTERNATIONAL ENERGY AGENCY (IEA). **World energy outlook 2014**. 2014. Disponível em: http://www.iea.org/publications/freepublications/publication/WorldEnergyOutlook2014ExecutiveSummaryPortugueseversion.pdf. Acesso em: 27 mar. 2017.

INTERNATIONAL ENERGY AGENCY (IEA). **Energy technology perspective:** scenario and strategies to 2050. Paris: [s.n.], 2008. 648 p. ISBN 9789264041424.

KERSCHGENS, M.; PILZ, U.; RASCHKE, E. A modified two-stream approximation for computations of the solar radiation budget in a cloudy atmosphere. **Tellus**, v. 30, p.429-435, 1978.

LABORATÓRIO NACIONAL DE COMPUTAÇÃO CIENTÍFICA (LNCC). **Supercomputador Santos Dumont.** 2019. Disponível em: http://sdumont.lncc.br/support_manual.php?pg=support#. Acesso em: 20 jan. 2017.

LAWRENCE LIVEMORE NATIONAL LABORATORY (LLNL). **OPENMP.** Disponível em: https://computing.llnl.gov/tutorials/openMP/. Acesso em: 15 dez. 2016.

LIAO C.; LIU Z.; HUANG L.; CHAPMAN B., Evaluating OpenMP on chip multi threading platforms. In: MUELLER, M. S.; CHAPMAN, B. M.; SUPINSKI, B. R.; MALONY, A. D.; VOSS, M. J. (Ed.). **OpenMP shared memory parallel programming**. Berlin: Springer, 2006. p. 178-190.

LIMA, F. J. L. **Previsão de irradiação solar no nordeste do Brasil empregando o modelo WRF ajustado por Redes Neurais Artificiais (RNAs)**. 2015. 252p. Tese (Doutorado em Meteorologia) – Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2015.

LIN, Y. C.; SNYDER, L. **Principles of parallel programming**. California: Pearson, 2008. 338p. ISBN 978-0321487902.

LENOBLE, J. **Radiative transfer in scattering and absorbing atmospheres:** standard computational procedures. Hampton: Deepak Publishing, 1985. 314p. ISBN 978-0937194058.

LIOU, K. N. On the absorption, reflection and transmission of solar radiation in cloudy atmospheres. **Journal of Atmospheric Science**, v. 33, p. 798-805, 1976.

LIOU, K. N. **An introduction to atmospheric radiation**. 2.ed. New York: Academic Press, 2002. 583 p. ISBN 978-0080491677.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION (NASA). **GOES:** overview and history. 2017. Disponível em: https://www.nasa.gov/content/goes-overview/index.html. Acesso em: 10 fev. 2017.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION (NASA). **Modeling GURU.** 2018. Disponível em: https://modelingguru.nasa.gov/docs/DOC-2676. Acesso em: 20 jan. 2019.

NÚCLEO AVANÇADO DE COMPUTAÇÃO DE ALTO DESEMPENHO (NACAD). **Modernização de códigos**. 2016. Disponível em: http://4w.hpc.usp.br/wp-content/uploads/2016/09/workshop_usp_out2016.pdf. Acesso em: 17 ago. 2018.

THE NOBEL PRIZE (NOBEL). **The Nobel prizes of physics**. Disponível em: https://www.nobelprize.org/prizes/physics. Acesso em: 10 fev. 2018.

MARTINOT, E. **Renewable energy policy network century**: renewable global future report 21. 2013. Disponível em: http://www.ren21.net/gsr-2018/pages/foreword/foreword/. Acesso em: 07 mar. 2017.

MARTINS, F.R.; PEREIRA, E.B.; ABREU, S. L.; COLLE, S. Mapas de irradiação solar para o Brasil: resultados do Projeto Swera. In: SIMPÓSIO BRASILEIRO DE SENSORIAMENTO REMOTO, 12., 2005, Goiania. **Anais...** São José dos Campos: INPE, 2005. p.3137-3145.

MARTINS, F.R. **Influência do processo de determinação da cobertura de nuvens e dos aerossóis de queimada no modelo físico de radiação BRASIL-SR**. 2001. 314p Dissertação (Doutorado em Geofísica) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2001.

MICHALAKES, J. Design of a next-generation regional weather research and forecast model. **Towards Teracomputing**, p. 117–124, 1999.

BRASIL. MINISTÉRIO DE MINAS E ENERGIA (MME). **Número de usinas eólicas se aproxima de 500 instalações no país.** 17 mar. 2017. Disponível em: http://www.mme.gov.br/web/guest/pagina-inicial/outras-noticas/-/asset_publisher/32hLrOzMKwWb/content/numeros-de-usinas-eolicas-se-aproximam-de-500-instalacoes-no-pais. Acesso em: 26 mar. 2017.

OPERADOR NACIONAL DO SISTEMA ELÉTRICO (ONS) . **Volume útil dos principais reservatórios**. Disponível em: http://www.ons.org.br/Paginas/resultados-da-operacao/historico-da-operacao/dados_hidrologicos_volumes.aspx. Acesso em: 08 mar. 2017.

PEREIRA, E. B.; MARTINS, F. R.; ABREU, S. L.; RUTHER, R. **Atlas brasileiro de energia solar**. São José dos Campos: INPE, 2006. 60p. ISBN 978-85-17-00030-0.

PEREIRA, E. B.; MARTINS, F.R.; COSTA, R.S.; GONÇALVES, A.R.; LIMA, F.L.; RÜTHER, R.; ABREU, S.l.; TIEPOLO, G.M.; PEREIRA, S.V.; SOUZA, J.G. **Atlas brasileiro de energia solar**. 2.ed. São José dos Campos: INPE, 2017. 80 p. ISBN 978-85-17-00089-8.

PEREIRA, A. G.; PETRY, A.; SOUZA, J. R; CHARÃO, A. S; VELHO, H. F. C, Aplicação de técnicas de paralelismo em modelo para previsão de dinâmica da ionosfera terrestre. In: INTERNATIONAL SYMPOSIUM ON COMPUTER ARCHITECTURE AND HIGH PERFORMANCE, 22., Petropólis, RJ. **Anais…** 2010.

PRESS, W. H.; TEUKALSKY, S. A.; VETTERLING, W. T.; FLANNERY, B. P.; METCALF, M. **Numerical recipes in Fortran 90:** the art of parallel scientific computing. 2.ed. Cambridge: Cambridge University Press, 1996. 572 p. ISBN 0-521-57439-0.

SANTOS, J.G.M. **Introdução ao Climate Data Operators (CDO)**. 2014. Disponível em: http://urlib.net/rep/8JMKD3MGP3W34P/3MQU4Q8?ibiurl.language=pt-BR. Acesso em: 12 fev. 2018.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS. SISTEMA DE ORGANIZAÇÃO NACIONAL DE DADOS AMBIENTAIS (INPE.SONDA), **Estações Sonda.** Disponível em: http://sonda.ccst.inpe.br/infos/index.html. Acesso em: 02 fev. 2017.

SILVA FILHO, L. C. P. **Estudo de casos com aplicações científicas de alto desempenho em agregados de computadores multi-core**. 2008, 106p. Dissertação (Mestrado em Ciências da Computação) – Universidade Federal de Santa Catarina, Florianópolis-SC, 2008.

STUHLMANN, R.; RIELAND, M.; RASCHKE, E. An improvement of the IGMK model to derive total and diffuse solar radiation at the surface from satellite data. **Journal of Applied Meteorology**, v. 29, n.7, p. 586-603, 1990.

TIEPOLO, G. M.; PEREIRA, E. B.; URBANETZ JR, J.; PEREIRA, S. V.; GONÇAVES, A. R.; LIMA, F. J. L.; COSTA, R. S., ALVES, A. R. **Atlas de energia solar do Estado do Paraná**. Curitiba: UTFPR, 2017. 107 p. ISBN 978-85-17-00091-1.

TANENBAUM, A. S. **Organização estruturada de computadores**. 5.ed. São Paulo: Prentice-Hall, 2007. 464 p. ISBN 857-60506-76.

TOLMASQUIM, M. T. **Energia renovável:** hidráulica, biomassa, eólica, solar, oceânica. 1.ed. Rio de Janeiro: EPE, 2016. 452 p. ISBN 978-85-60025-06-0.

TOMITA S.; RODRIGUES, L. F.; PANETTA, J. Paralelização em dois níveis do Modelo Regional de Previsão de Tempo – Eta. In: WORKSHOP EM SISTEMAS COMPUTACIONAIS DE ALTO DESEMPENHO, 4., 2003, São Paulo. **Anais...** Biblioteca Digita Brasileira de Computação, 2003. p. 96-100.

TOP500. **List november 2016.** 2016. Disponível em: https://www.top500.org/lists/2016/11/. Acesso em: 28 mar. 2017.

UNIVERSIDADE ESTADUAL DE CAMPINAS (UNICAMP). **Apostila de treinamento**: introdução ao Fortran90. Disponível em: https://www.cenapad.unicamp.br/servicos/treinamentos/apostilas/apostila_fortran90.pdf . Acesso em: 06 jan. 2018.

UNIVERSIDADE ESTADUAL DE CAMPINAS; CENTRO NACIONAL DE PROCESSAMENTO DE ALTO DESEMPENHO (UNICAMP. CENEPAD). **Introdução ao OpenMP**. 2012. Disponível em: www.cenapad.unicamp.br/servicos/treinamentos/apostilas/apostila_openmp.pdf. Acesso em: 19 nov. 2018.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL; CENTRO NACIONAL DE SUPERCOMPUTAÇÃO (UFRGS, CESUP). **20 anos de história**. Disponível em: http://20anos.cesup.ufrgs.br/historia. Acesso em: 25 mar. 2016.

YAMAMOTO, G. Direct absorption of solar radiation by atmospheric water vapor, carbon dioxide and molecular oxygen. **Journal of Atmospheric Science**, v. 19, p. 182-188, 1962.

APÊNDICE A – SCRIPT PARA A OPERACIONALIZAÇÃO DO MODELO BRASIL-SR

A.1 Scripts

Figura A.1 - Script principal.

```
#!/usr/bin python
# -*- coding: utf-8 -*-
"""
@author: Jefferson G. Souza
Pré e Processamento do Modelo BrasilSR
**** Entrar com mês e ano
"""
import sys
import os
#---------Funções---------
import preprocessamento
import MaxMin
import MaxMinP
import desc_arq
import cria_dir
import des_prop
import help
import gera_dat
import max_min
import cobnuvens
import des_prop
import renomeia
import copyfiles
import baixaCLIMATOLOGIA
import gera_netcdf
import executa
#---------Path's----------
dirPD = '/Dados/brasil_SR/'
#---------Argumentos de entrada------
mes  = sys.argv[1]
ano  = sys.argv[2]
dia  = sys.argv[3]
try:
    P0   = sys.argv[4]
    P100 = sys.argv[5]
except Exception:
    P0   = '0'
    P100 = '100'
#---------Preprocessamento
preprocessamento.Preproc(dirPD, mes, ano)
#------MaxMinPxPxx------------------
MaxMinP.calcP(dirPD, mes, ano, P0, P100)
#------CEFF-------------------
cobnuvens.ceff(dirPD, mes, ano)
#------Criando arquivos de entrada------------------
des_prop.criaDP(dirPD, mes, ano)
renomeia.rename(dirPD, mes, ano)
copyfiles.files(dirPD, mes, ano)
baixaCLIMATOLOGIA.clima(dirPD, mes, ano, dia)
gera_netcdf.netcdf(dirPD, mes, ano, dia)
#------Executando Modelo------------------
executa.modelo(dirPD, mes, ano)
```

Figura A.2 - Script de pré-processamento

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jan 22 15:18:39 2018

@author: Jefferson G. Souza
Pré-processamento do Modelo BrasilSR
"""
import os

#-----Funções
import desc_arq
import verLB
import validaBSR
import conv_nc
import verif_ERRO

dirIMG = '/home/brasilsr/imagens_goes/'

def Preproc(dirPD, mes, ano):
#------Criando Diretórios
    if not os.path.exists(dirPD+'modelo'+ano):
        os.mkdir(dirPD+'modelo'+ano)
    if not os.path.exists(dirPD+'modelo'+ano+'/'+mes):
        os.mkdir(dirPD+'modelo'+ano+'/'+mes)
    if not os.path.exists(dirPD+'modelo'+ano+'/'+mes+'/lch1'):
        os.mkdir(dirPD+'modelo'+ano+'/'+mes+'/lch1')
    if not os.path.exists(dirPD+'modelo'+ano+'/'+mes+'/lch4'):
        os.mkdir(dirPD+'modelo'+ano+'/'+mes+'/lch4')
    if not os.path.exists(dirPD+'modelo'+ano+'/'+mes+'/lnav'):
        os.mkdir(dirPD+'modelo'+ano+'/'+mes+'/lnav')
    if not os.path.exists(dirPD+'modelo'+ano+'/'+mes+'/maxmin'):
        os.mkdir(dirPD+'modelo'+ano+'/'+mes+'/maxmin')
    if not os.path.exists(dirPD+'modelo'+ano+'/'+mes+'/description'):
        os.mkdir(dirPD+'modelo'+ano+'/'+mes+'/description')
    if not os.path.exists(dirPD+'modelo'+ano+'/'+mes+'/prop'):
        os.mkdir(dirPD+'modelo'+ano+'/'+mes+'/prop')
    if not os.path.exists(dirPD+'modelo'+ano+'/'+mes+'/trans'):
        os.mkdir(dirPD+'modelo'+ano+'/'+mes+'/trans')
    if not os.path.exists(dirPD+'modelo'+ano+'/'+mes+'/images'):
        os.mkdir(dirPD+'modelo'+ano+'/'+mes+'/images')
    if not os.path.exists(dirPD+'modelo'+ano+'/'+mes+'/images/cn'):
        os.mkdir(dirPD+'modelo'+ano+'/'+mes+'/images/cn')
    if not os.path.exists(dirPD+'modelo'+ano+'/'+mes+'/images/kd'):
        os.mkdir(dirPD+'modelo'+ano+'/'+mes+'/images/kd')
    if not os.path.exists(dirPD+'modelo'+ano+'/'+mes+'/images/kt'):
        os.mkdir(dirPD+'modelo'+ano+'/'+mes+'/images/kt')
    if not os.path.exists(dirPD+'modelo'+ano+'/'+mes+'/images/kp'):
        os.mkdir(dirPD+'modelo'+ano+'/'+mes+'/images/kp')
    if not os.path.exists(dirPD+'modelo'+ano+'/'+mes+'/images/netcdf'):
        os.mkdir(dirPD+'modelo'+ano+'/'+mes+'/images/netcdf')
    if not os.path.exists(dirPD+'modelo'+ano+'/'+mes+'/images/solar_nc'):
        os.mkdir(dirPD+'modelo'+ano+'/'+mes+'/images/solar_nc')
    if not os.path.exists(dirPD+'modelo'+ano+'/'+mes+'/images/total'):
        os.mkdir(dirPD+'modelo'+ano+'/'+mes+'/images/total')
    print 'Copiando ch1, ch4 e Nav'
#------Copiando Imagens de Satélite
    os.system('cp -R '+dirIMG+mes+'/ch1/* '+dirPD+'modelo'+ano+'/'+mes+'/lch1/')
    os.system('cp -R '+dirIMG+mes+'/ch4/* '+dirPD+'modelo'+ano+'/'+mes+'/lch4/')
    os.system('cp -R '+dirIMG+mes+'/nav/* '+dirPD+'modelo'+ano+'/'+mes+'/lnav/')
#------Descompactando
    desc_arq.desconp(dirPD, ano, mes)
#------Verifica LittleBig
    verLB.LB(dirPD, mes, ano)
#------Validando
    validaBSR.vBR(dirPD, ano, mes)
#------Convertendo Binário para Netcdf
    conv_nc.bin2nc(dirPD, mes, ano)
#------Verificando Erros
    verif_ERRO.VerifERRO(dirPD, mes, ano)
```

Figura A.3 - Script para descompactar arquivos.

```
#!/usr/bin python
# -*- coding: utf-8 -*-
import os
def desconp(dirp,ano,mes):
   dirpd = dirp+'modelo'+ano+'/'+mes
   print 'Descompactando ch1'
   os.chdir(dirpd+'/lch1')
   os.system('gunzip *.gz')
   print 'Descompactando ch4'
   os.chdir(dirpd+'/lch4')
   os.system('gunzip *.gz')
   print 'Descompactando nav'
   os.chdir(dirpd+'/lnav')
   os.system('gunzip *.gz')
```

Figura A.4 – Script para verificar como o arquivo está escrito.

```
# -*- coding: utf-8 -*-
"""
@author: Jefferson G. Souza
Executando o programa para verificar se arquivo é Little ou big endian
"""
import os
def LB(dirPD, mes, ano):
    os.chdir(dirPD+'modelo'+ano+'/'+mes+'/')
    os.mkdir('ch1')
    os.mkdir('ch4')
    os.mkdir('nav')
    os.chdir(dirPD+'modelo'+ano+'/'+mes+'/lch1/')
    os.system('ls S10238214* > lista.dat')
    os.chdir('/Dados/brasil_SR/tese_Jeff/progs_aux/')
    os.system('./LB '+mes+' '+ano)
    os.chdir(dirPD+'modelo'+ano+'/'+mes+'/')
    os.system('mv lch1/S10238214* ch1/')
    os.system('mv lch4/S10238220* ch4/')
    os.system('rm -R lch1')
    os.system('rm -R lch4')
    os.system('rm -R lnav')
```

Figura A.5 – Script para validar os arquivos de entrada.

```
# -*- coding: utf-8 -*-
"""
@author: Jefferson G. Souza
Validação dos dados de entrada do modelo
"""
import os
def vBR(dirpd, ano, mes):
#------Criando Listas
    os.system('cp /Dados/tese/OperDIARIO/renomeia ' +dirpd+'modelo'+ano+'/'+mes+'/ch1/.')
    os.system('cp /Dados/tese/OperDIARIO/renomeia ' +dirpd+'modelo'+ano+'/'+mes+'/ch4/.')
    os.system('cp /Dados/tese/OperDIARIO/renomeia ' +dirpd+'modelo'+ano+'/'+mes+'/nav/.')
#------Renomeando Ch1
    os.chdir(dirpd+'modelo'+ano+'/'+mes+'/ch1/')
    os.system('ls S10238213* > lista.txt')
    os.system('ls S10238213* > lista_aux.txt')
    os.system('./renomeia')
    os.system('rm lista* renomeia S10238214*')
    os.system('ls *.hdr > canal_1.dat')
#------Renomeando Ch4
    os.chdir(dirpd+'modelo'+ano+'/'+mes+'/ch4/')
    os.system('ls S10238219* > lista.txt')
    os.system('ls S10238219* > lista_aux.txt')
    os.system('./renomeia')
    os.system('rm lista* renomeia S10238220*')
#------Renomeando Nav
    os.chdir(dirpd+'modelo'+ano+'/'+mes+'/nav/')
    os.system('ls S1023* > lista.txt')
    os.system('ls S1023* > lista_aux.txt')
    os.system('./renomeia')
    os.system('rm lista* renomeia')
#------Validação
    print 'Validação'
    os.chdir('/Dados/tese/OperDIARIO/')
    os.system('./valida '+mes+' '+ano)
#------Reamostragem
    print 'Reamostragem CH1'
    os.chdir(dirpd+'modelo'+ano+'/'+mes+'/nav/')
    os.system('mv *.lat '+dirpd+'modelo'+ano+'/'+mes+'/ch1/')
    os.system('mv *.lon '+dirpd+'modelo'+ano+'/'+mes+'/ch1/')
    os.system('cp /Dados/tese/OperDIARIO/reamostra* '+dirpd+'modelo'+ano+'/'+mes+'/ch1/')
    os.system('cp /Dados/tese/OperDIARIO/navpadrao0106* '+dirpd+'modelo'+ano+'/'+mes+'/ch1/')
    os.chdir(dirpd+'modelo'+ano+'/'+mes+'/ch1/')
    os.system('ls *.hdr > canal_1.dat')
    os.chdir(dirpd+'modelo'+ano+'/'+mes+'/ch1/')
    os.system('./reamostragem -e canal_1.dat')
#------Reamostragem
    print 'Reamostragem CH4'
    os.chdir(dirpd+'modelo'+ano+'/'+mes+'/ch1/')
    os.system('mv *.lat '+dirpd+'modelo'+ano+'/'+mes+'/ch4/')
    os.system('mv *.lon '+dirpd+'modelo'+ano+'/'+mes+'/ch4/')
    os.system('cp /Dados/tese/OperDIARIO/reamostra* '+dirpd+'modelo'+ano+'/'+mes+'/ch4/')
    os.system('cp /Dados/tese/OperDIARIO/navpadrao0106* '+dirpd+'modelo'+ano+'/'+mes+'/ch4/')
    os.chdir(dirpd+'modelo'+ano+'/'+mes+'/ch4/')
    os.system('ls *.hdr > canal_4.dat')
    os.chdir(dirpd+'modelo'+ano+'/'+mes+'/ch4/')
    os.system('./reamostragem -e canal_4.dat')
#------Limpando Diretório
    print 'Limpando Diretórios'
    os.chdir(dirpd+'modelo'+ano+'/'+mes+'/ch1/')
    os.system('rm reamostra* canal_1.dat gc*')
    os.system('mv rea/* .')
    os.system('rm -R rea')
    os.chdir(dirpd+'modelo'+ano+'/'+mes+'/ch4/')
    os.system('rm reamostra* canal_4.dat gc* navp* ')
    os.system('mv rea/* .')
    os.system('rm -R rea')
    os.system('mv '+dirpd+'modelo'+ano+'/'+mes+'/nav '+dirpd+'modelo'+ano+'/'+mes+'/nc' )
```

Figura A.6 – Script para converter os arquivos de Binário para NetCDF.

```
# -*- coding: utf-8 -*-
"""
@author: brasilsr
"""
import os
def bin2nc(dirPD, mes, ano):
#------Criando lista com diretoriórios
    print 'Criando Lista com Diretórios para conversão'
    os.chdir(dirPD)
    arq = open('lista_dir.txt', 'w')
    arq.write('/Dados/brasil_SR/modelo'+ano+'/'+mes+'/nc/\n')
    arq.write('/Dados/brasil_SR/modelo'+ano+'/'+mes+'/ch4/\n')
    arq.write('/Dados/brasil_SR/modelo'+ano+'/'+mes+'/ch1/\n')
    arq.write('/Dados/brasil_SR/modelo'+ano+'/'+mes+'/lista_imagens.dat\n')
    arq.write('/Dados/tese/OperDIARIO/navpadrao0106.lat\n')
    arq.write('/Dados/tese/OperDIARIO/navpadrao0106.lon')
    arq.close()
    os.system('cp /Dados/tese/OperDIARIO/lista_imagens.dat '+dirPD+'modelo'+ano+'/'+mes+'/')
#------Realizando Conversão
    print 'Convertendo Dados Binários para Netcdf'
    os.system('./vi_ir_75')
    os.system('rm -R /Dados/brasil_SR/modelo'+ano+'/'+mes+'/ch1/')
    os.system('rm -R /Dados/brasil_SR/modelo'+ano+'/'+mes+'/ch4/')
    os.system('mv /Dados/brasil_SR/modelo'+ano+'/'+mes+'/nc/*
/Dados/brasil_SR/modelo'+ano+'/'+mes+'/images/netcdf/')
    os.system('rm -R /Dados/brasil_SR/modelo'+ano+'/'+mes+'/nc/')
    os.system('ls /Dados/brasil_SR/modelo'+ano+'/'+mes+'/images/netcdf/*.nc > lista_nc.dat')
```

Figura A.7 – Script para verificar erros nos arquivos.

```
# -*- coding: utf-8 -*-
"""
@author: brasilsr
"""
import os
def VerifERRO(dirPD, mes, ano):
    print'Verificando Erros'
    os.chdir('/Dados/tese/OperDIARIO/')
    os.system('./VErro '+mes+' '+ano)
```

Figura A.8 – Script para calcular o máximo e mínimo.

```
# -*- coding: utf-8 -*-
"""
@author: Jefferson G. Souza
Pré-processamento do Modelo BrasilSR
"""
import os
def calcP(dirPD, mes, ano, P0, P100):
   os.chdir('/Dados/brasil_SR/modelo'+ano+'/'+mes+' /imagens/netcdf/')
   os.system('ls *.nc > lista.dat')
   os.chdir('/Dados/tese/OperDIARIO/')
   os.system('./MMP '+mes+' '+ano+' '+P0+' '+P100)
```

Figura A.9 – Script para calcular a cobertura de nuvens.

```
# -*- coding: utf-8 -*-
"""
@author: Jefferson G. Souza
Calculo da Cobertura Efetiva de Nuvens do Modelo BrasilSR
"""
import os
def ceff(dirPD, mes, ano):
    os.chdir('/Dados/tese/OperDIARIO/')
    os.system('./ceffNetcdf '+mes+' '+ano) #alterar diretorios do código
```

Figura A10. Script para criar descritores e baixar climatologia mensal

```
#!/usr/bin python
# -*- coding: utf-8 -*-

import os
import shutil
import os.path
from ftplib import FTP

def criaDP(dirPD, mes, ano):
#----Criando arquivo descritor
   name = mes+ano+'.des'
   os.chdir(dirPD+'modelo'+ano+'/'+mes+'/description')
   arq = open(name,'w')
   arq.write('Imagens\nGOES12\n1\n1\n1180\n1784\n'+mes+ano+'.lat\n'+mes+ano+'.lon')
   arq.close()
   print 'Copiando Lat Lon'
#----Copiando e renomeando arquivos de lat e lon
   shutil.copy(dirPD+'BRASILSR/1784x1180/navpadrao0106.lat',dirPD+'modelo'+ano+'/'+mes+'/description')
   shutil.copy(dirPD+'BRASILSR/1784x1180/navpadrao0106.lon',dirPD+'modelo'+ano+'/'+mes+'/description')
   lat = mes+ano+'.lat'
   lon = mes+ano+'.lon'
   os.rename('navpadrao0106.lat',lat)
   os.rename('navpadrao0106.lon',lon)
#----Copiando arquivos com climatologia
# Definindo variaveis
   print 'Baixando Climatologia'
   ftp_host = '150.163.XX.XX' # Servidor FTP
   ftp_user = 'XXX'         # Usuario
   ftp_pasw = 'XXX'  # Senha
#----------------------------------------
# Logando Servidor
   arquivo  = str(mes)+'_med'       #Definir o arquivo que sera baixado
   ftp = FTP(host=ftp_host,user=ftp_user,passwd=ftp_pasw)

   os.chdir(dirPD+'modelo'+str(ano)+'/'+mes+'/prop/') # Entra no Local PC para os dados
   try:
          file = ftp.cwd('/pools/A/A0/BrasilSR/climatologia/')  #Diretorio FTP
          for filename in ftp.nlst(arquivo):
             fhandle = open(filename,'wb')
             print 'Baixando '+ filename
             ftp.retrbinary('RETR '+filename, fhandle.write)
             fhandle.close()
   except Exception:
        print '###################################'
        print '#### Sem dados para '+str(ano)+' #####' #enviar email
        print '###################################'
#----------------------------------------
# Logando Servidor
   arquivo  = str(mes)+'CLIMA1784x1180.macc' #Definir o arquivo que sera baixado
   ftp = FTP(host=ftp_host,user=ftp_user,passwd=ftp_pasw)
   os.chdir(dirPD+'modelo'+str(ano)+'/'+mes+'/prop/') # Entra no Local PC para os dados
   try:
```

```
        file = ftp.cwd('/pools/A/A0/BrasilSR/Macc/MACC_Clima_1784x1180/')  #Diretorio FTP
        for filename in ftp.nlst(arquivo):
            fhandle = open(filename,'wb')
            print 'Baixando '+ filename
            ftp.retrbinary('RETR '+filename, fhandle.write)
            fhandle.close()
  except Exception:
      print '###################################'
      print '#### Sem dados para '+str(ano)+' #####' #enviar email
      print '###################################'
#----------------------------------------
# Logando Servidor
  arquivo  = 'altitude.alt'       #Definir o arquivo que sera baixado
  ftp = FTP(host=ftp_host,user=ftp_user,passwd=ftp_pasw)
  os.chdir(dirPD+'modelo'+str(ano)+'/'+mes+'/prop/') # Entra no Local PC para os dados
  try:
        file = ftp.cwd('/pools/A/A0/BrasilSR/')  #Diretorio FTP
        for filename in ftp.nlst(arquivo):
            fhandle = open(filename,'wb')
            print 'Baixando '+ filename
            ftp.retrbinary('RETR '+filename, fhandle.write)
            fhandle.close()
  except Exception:
      print '###################################'
      print '#### Sem dados para '+str(ano)+' #####' #enviar email
      print '###################################'

  alt  = mes+ano+'.alt'
  macc = mes+ano+'.macc'
  clim = mes+'_'+ano
  os.chdir(dirPD+'modelo'+ano+'/'+mes+'/prop')
  os.rename('altitude.alt',alt)
  os.rename(str(mes)+'CLIMA1784x1180.macc',macc)
  os.rename(str(mes)+'_med',clim)
```

Figura A.11 – Script para copiar os executáveis do modelo.

```
# -*- coding: utf-8 -*-
"""
@author: Jefferson Gonçalves de Souza
"""
import shutil
def files(dirPD, mes, ano):
    shutil.copy(dirPD+'BRASILSR/Srbpres.dat',dirPD+'modelo'+ano+'/')
    shutil.copy(dirPD+'BRASILSR/ModeloTese/modeloDIA/fonte1/1_DIA',dirPD+'modelo'+ano+'/')
    shutil.copy(dirPD+'BRASILSR/ModeloTese/modeloDIA/fonte2/2_DIA',dirPD+'modelo'+ano+'/')
    shutil.copy(dirPD+'BRASILSR/ModeloTese/modeloDIA/fonte3/3_DIA',dirPD+'modelo'+ano+'/')
    shutil.copy(dirPD+'BRASILSR/ModeloTese/modeloDIA/fonte4/4_DIA',dirPD+'modelo'+ano+'/')
    shutil.copy(dirPD+'BRASILSR/ModeloTese/modeloDIA/fonte5/5_DIA',dirPD+'modelo'+ano+'/')
    shutil.copy(dirPD+'BRASILSR/ModeloTese/modeloDIA/fonte6/6_DIA',dirPD+'modelo'+ano+'/')
    shutil.copy(dirPD+'BRASILSR/VarAmb.sh',dirPD+'modelo'+ano+'/')
```

Figura A.12 – Script para baixar dados climatológicos diários.

```
#! /usr/bin/env python

import sys
import os
import urllib2
import cookielib
import Conv_mes
#
def clima (dirPD, mes, ano, dia):
    os.chdir(dirPD+'modelo'+ano+'/'+mes+'/clima/')
    passw = 'XXXX'
    verbose=True

    cj=cookielib.MozillaCookieJar()
    opener=urllib2.build_opener(urllib2.HTTPCookieProcessor(cj))
    do_authentication=False
    if (os.path.isfile("auth.rda.ucar.edu")):
      cj.load("auth.rda.ucar.edu",False,True)
      for cookie in cj:
        if (cookie.name == "sess" and cookie.is_expired()):
          do_authentication=True
    else:
      do_authentication=True
    if (do_authentication):
      opener.open("https://rda.ucar.edu/cgi-
bin/login","email=jefferson.souza@inpe.br&password="+passw+"&action=login")
      cj.clear_session_cookies()
      cj.save("auth.rda.ucar.edu",True,True)
    n_mes = Conv_mes.conv(mes)

listoffiles=[ano+"/"+ano+n_mes+dia+"/gfs.0p25."+ano+n_mes+dia+"00.f006.grib2",ano+"/"+ano+n_mes+dia+"/gfs.0p25.
"+ano+n_mes+dia+"06.f006.grib2",ano+"/"+ano+n_mes+dia+"/gfs.0p25."+ano+n_mes+dia+"12.f006.grib2",ano+"/"+ano
+n_mes+dia+"/gfs.0p25."+ano+n_mes+dia+"18.f006.grib2"]
    for file in listoffiles:
      idx=file.rfind("/")
      if (idx > 0):
        ofile=file[idx+1:]
      else:
        ofile=file
      if (verbose):
        sys.stdout.write("downloading "+ofile+"...")
        sys.stdout.flush()
      infile=opener.open("http://rda.ucar.edu/data/ds084.1/"+file)
      outfile=open(ofile,"wb")
      outfile.write(infile.read())
      outfile.close()
      if (verbose):
        sys.stdout.write("done.\n")
```

Figura A.13 – Script para converter os dados climatológicos para NetCDF

```
# -*- coding: utf-8 -*-
"""
@author: brasilsr
"""
import os
import shutil

def netcdf(dirPD, mes, ano, dia):
    shutil.copy('/Dados/tese/OperDIARIO/com_cdo.sh',dirPD+'modelo'+ano+'/'+mes+'/clima/')
    shutil.copy('/Dados/tese/OperDIARIO/mygrid',dirPD+'modelo'+ano+'/'+mes+'/clima/')
    os.chdir(dirPD+'modelo'+ano+'/'+mes+'/clima/')
    os.system('./com_cdo.sh')
    os.system('rm *.grib2')
    os.system('rm *.f006.nc')
    os.system('ls *Int.nc > lista.dat')
    os.chdir('/Dados/tese/OperDIARIO/')
    os.system('./Cclima '+mes+' '+ano)
```

Figura A.14 – Script para executar o modelo.

```
# -*- coding: utf-8 -*-
"""
@author: brasilsr
"""
import os
def modelo(dirPD, mes, ano):
    print'..... EXECUTANDO MODELO BRASIL-SR .....'
    os.chdir(dirPD+'modelo'+ano+'/')
    os.system('./VarAmb.sh')
    os.system('/bin/bash -s source /opt/intel/bin/compilervars.sh intel64')
    os.system('export OMP_NUM_THREADS=6')
    os.system('export OMP_STACKSIZE=1000M')
    os.system('ulimit -s unlimited')
    print '> > > 1/6: Trans'
    os.system('./1_DIA')
    print'> > > 2/6: Trans Par'
    os.system('./2_DIA')
    os.system('rm gkss.dat')
    os.system('mv horarios.txt gkss.dat')
    print'> > > 3/6: DF DH GL KD KT'
    os.system('./3_DIA')
    print'> > > 4/6: KP PA'
    os.system('./4_DIA')
    print'> > > 5/6: TOTAL DIARIO PAR'
    os.system('./5_DIA')
    print'> > > 6/6: TOTAL DIARIO GL, DH, DF, TL'
    os.system('./6_DIA')
    print'>>>> Fim da execuao do modelo'
```

APÊNDICE B – CÓDIGOS EM FORTRAN 90

B.1 Código para o pré-processamento do modelo

Figura B.1 – Código para verificar o modo de leitura do arquivo.

```
program VLB
 character*200 dir_ch1, dir_lis, dir_file, dir_arq
 character*200 file_lch1, file_lch4, file_llat, file_llon
 character*200 file_ch1, file_ch4, file_lat, file_lon
 character*59  HDRnome
 character*22  nome
 character*12  date
 character*7   Mnome
 character*4   ano, nll, ncc
 character*3   mes, LBtipo

 integer*2     LEITURA(4000,2500)
 integer       ierr, ERRO, nc, nl, i, j, NLC, NNLC
 logical       EXISTE_ch1, EXISTE_ch4, EXISTE_lat, EXISTE_lon

 call getarg(1,mes)
 call getarg(2,ano)

 Mnome = "S102382"
 dir_arq = "/Dados/brasil_SR/modelo"//ano//"/"//mes//"/"
 dir_ch1 = "/Dados/brasil_SR/modelo"//ano//"/"//mes//"/lch1/"
 dir_lis = "/Dados/brasil_SR/modelo"//ano//"/"//mes//"/lch1/lista.dat" !lista HDR

 OPEN(10,FILE=dir_lis)
 do
  READ(10,100,iostat=ierr) nome
  100 FORMAT(22A)
  if(ierr/=0)exit
  print*,nome
  dir_file = trim(dir_ch1)//nome
  date = nome(11:22)

  OPEN(20,FILE=dir_file)
  READ(20,101,iostat=ierr) HDRnome
  101 FORMAT(59A)
  LBtipo = HDRnome(57:59)
  nll = HDRnome(47:50)
  ncc = HDRnome(52:55)
  read(HDRnome(47:50),'(I)')nl
  read(HDRnome(52:55),'(I)')nc

  NLC  = nl * nc *2
  NNLC = nl * nc

  if (LBtipo == 'LNX')then
     file_lch1 = trim(dir_arq)//"lch1/"//Mnome//"13_"//date
     file_lch4 = trim(dir_arq)//"lch4/"//Mnome//"19_"//date
     file_llat = trim(dir_arq)//"lnav/"//Mnome//"23_"//date
     file_llon = trim(dir_arq)//"lnav/"//Mnome//"24_"//date

     file_ch4 = trim(dir_arq)//"ch4/"//Mnome//"19_"//date
     file_ch1 = trim(dir_arq)//"ch1/"//Mnome//"13_"//date
     file_lat = trim(dir_arq)//"nav/"//Mnome//"23_"//date
     file_lon = trim(dir_arq)//"nav/"//Mnome//"24_"//date

     inquire(file=file_lch1,EXIST=EXISTE_ch1)
     IF(EXISTE_ch1) THEN
```

```
        !abrindo arquivo LB
        OPEN(99,FILE=file_lch1,FORM='unformatted',RECL=NLC,STATUS='OLD',ACCESS='direct')
        READ(99,REC=1,ERR=1112) ((LEITURA(I,J),I=1,NC),J=1,NL)
        CALL MOVEBITES(LEITURA,NL,NC)
        !Salvando arquivo BIG
        OPEN(999,FILE=file_ch1,FORM='unformatted',RECL=NNLC,STATUS='replace',ACCESS='direct')
        WRITE(999,REC=1,ERR=1112) ((LEITURA(I,J),I=1,NC),J=1,NL)
        CLOSE (99)
        CLOSE (999)
      endif

      inquire(file=file_lch4,EXIST=EXISTE_ch4)
      IF(EXISTE_ch4) THEN
        !abrindo arquivo LB
        OPEN(99,FILE=file_lch4,FORM='unformatted',RECL=NLC,STATUS='OLD',ACCESS='direct')
        READ(99,REC=1,ERR=1112) ((LEITURA(I,J),I=1,NC),J=1,NL)
        CALL MOVEBITES(LEITURA,NL,NC)
        !Salvando arquivo BIG
        OPEN(999,FILE=file_ch4,FORM='unformatted',RECL=NNLC,STATUS='replace',ACCESS='direct')
        WRITE(999,REC=1,ERR=1112) ((LEITURA(I,J),I=1,NC),J=1,NL)
        CLOSE (99)
        CLOSE (999)
      endif

      inquire(file=file_llat,EXIST=EXISTE_lat)
      IF(EXISTE_lat) THEN
        !abrindo arquivo LB
        OPEN(99,FILE=file_llat,FORM='unformatted',RECL=NLC,STATUS='OLD',ACCESS='direct')
        READ(99,REC=1,ERR=1112) ((LEITURA(I,J),I=1,NC),J=1,NL)
        CALL MOVEBITES(LEITURA,NL,NC)
        !Salvando arquivo BIG
        OPEN(999,FILE=file_lat,FORM='unformatted',RECL=NNLC,STATUS='replace',ACCESS='direct')
        WRITE(999,REC=1,ERR=1112) ((LEITURA(I,J),I=1,NC),J=1,NL)
        CLOSE (99)
        CLOSE (999)
      endif

      inquire(file=file_llon,EXIST=EXISTE_lon)
      IF(EXISTE_lon) THEN
        !abrindo arquivo LB
        OPEN(99,FILE=file_llon,FORM='unformatted',RECL=NLC,STATUS='OLD',ACCESS='direct')
        READ(99,REC=1,ERR=1112) ((LEITURA(I,J),I=1,NC),J=1,NL)
        CALL MOVEBITES(LEITURA,NL,NC)
        !Salvando arquivo BIG
        OPEN(999,FILE=file_lon,FORM='unformatted',RECL=NNLC,STATUS='replace',ACCESS='direct')
        WRITE(999,REC=1,ERR=1112) ((LEITURA(I,J),I=1,NC),J=1,NL)
        CLOSE (99)
        CLOSE (999)
      endif
    endif

  enddo

  close(10)

  STOP
  1112 PRINT*, 'Erro de leitura do arquivo de imagem'
 end program

 !*****************************************
      SUBROUTINE MOVEBITES (READING, L, C)
      INTEGER L, C
      INTEGER*2 READING(4000,2500)
      INTEGER*2 ITEMP
      CHARACTER*1 JTEMP(2)
      CHARACTER*1 KTEMP

      EQUIVALENCE(JTEMP(1), ITEMP)
      SAVE
```

```
      DO N = 1, L
       DO M = 1, C
        ITEMP = READING(M,N)
        KTEMP = JTEMP(2)
        JTEMP(2) = JTEMP(1)
        JTEMP(1) = KTEMP
        READING(M,N)= ITEMP

      END DO
      END DO
      RETURN
      END SUBROUTINE
```

Figura B.2 – Código para verificação de erros nos arquivos.

```
program verifica_erro
 use netcdf

 character*200 img,nc,ncdf, erro, rela
 character*31  line
 character*18  nome
 character*4   ano,hora
 character*3   mes,Jdia
 integer       ierr, i, j
 integer    :: noturno, soma_vi, soma_ir
 integer    :: ndims_in, nvars_in, ngatts_in, unlimdimid_in
 integer    :: lat_varid, lon_varid, ilat, ilon
 integer    :: ir_varid, vi_varid, ang_sol_varid, ang_sat_varid, ang_sat_sol_varid
 integer    :: mat_erro_ir(1784,1180), mat_erro_vi(1784,1180)

 real       :: erro_vi, erro_ir
 real, dimension(:,:), allocatable :: vi, ir, ang_sol
 real, dimension(:),   allocatable :: lats, lons

 call getarg(1,mes)
 call getarg(2,ano)

nc   = "/Dados/brasil_SR/modelo"//ano//"/"//mes//"/images/netcdf/"
img  = "/Dados/brasil_SR/modelo"//ano//"/"//mes//"/lista_imagens.dat"
erro = "/Dados/brasil_SR/modelo"//ano//"/"//mes//"/Erros.dat"
rela = "/Dados/brasil_SR/modelo"//ano//"/"//mes//"/Relatorio.dat"

open(20,file=erro)
open(30,file=rela)

open(10,file=img)
do
  read(10,100,iostat=ierr)line
  100 format(31A)
  if(ierr/=0)exit
  Jdia = line(29:31)
  hora = line(9:12)
  ncdf = trim(nc)//'sat'//Jdia//ano//'.'//hora//'.nc'
  nome = 'sat'//Jdia//ano//'.'//hora//'.nc'
  print*,ncdf
!------------ Open netcdf com dados de Satelite
       call check(nf90_open(ncdf, nf90_write, ncid) )
       call check(nf90_inquire(ncid, ndims_in, nvars_in, ngatts_in, unlimdimid_in))

       call check(nf90_inq_varid(ncid,"latitude", lat_varid))
       call check(nf90_inq_varid(ncid,"longitude", lon_varid))
       call check(nf90_inq_varid(ncid,"imagem_ir", ir_varid))
       call check(nf90_inq_varid(ncid,"imagem_vi", vi_varid))
       call check(nf90_inq_varid(ncid,"angulo_sol", ang_sol_varid))
       call check(nf90_inq_varid(ncid,"angulo_satelite", ang_sat_varid))
```

```
      call check(nf90_inq_varid(ncid,"angulo_satelite_sol", ang_sat_sol_varid))

      call check(nf90_inquire_dimension(ncid,lat_varid,len=ilat))
      call check(nf90_inquire_dimension(ncid,lon_varid,len=ilon))

      allocate(vi(ilon, ilat))
      allocate(ir(ilon, ilat))
      allocate(ang_sol(ilon, ilat))

      allocate(lats(ilat))
      allocate(lons(ilon))

      call check(nf90_get_var(ncid,lat_varid,lats))
      call check(nf90_get_var(ncid,lon_varid,lons))
      call check(nf90_get_var(ncid,vi_varid,vi))
      call check(nf90_get_var(ncid,ir_varid,ir))
      call check(nf90_get_var(ncid,ang_sol_varid,ang_sol))
!-----------------------------------------
!----------Identificando pixel's com Erro e Noturnos
noturno = 0
soma_vi = 0
soma_ir = 0

do i=1,1784
 do j=1,1180
   mat_erro_ir(i,j) = 0
   mat_erro_vi(i,j) = 0
 enddo
enddo

do i=1,1784
 do j=1,1180
   if((ir(i,j) <= 0).and.(ir(i,j) /= -99))then
     ir(i,j) = -32768
     mat_erro_ir(i,j) = 1
     soma_ir = soma_ir + 1
   endif

   if((vi(i,j) <= 0).and.(vi(i,j) /= -99))then
     vi(i,j) = -32768
     mat_erro_vi(i,j) = 1
     soma_vi = soma_vi + 1
   endif
   !Marcando Pixel noturno
   if(ang_sol(i,j) >= 85)then
     vi(i,j) = -99
     ir(i,j) = -99
     noturno = noturno + 1
   endif
 enddo
enddo

erro_vi = ((soma_vi*100)/2105120)
erro_ir = ((soma_ir*100)/2105120)
print*,"Fim da Verificação de Erro e Codigo Noturno"
write(30,*)'Arquivo ',nome
write(30,*)'Erro_vi: ',erro_vi,'%'
write(30,*)'Erro_ir: ',erro_ir,'%'
write(30,*)'Codigo noturno: ',((noturno*100)/2105120),'%'

if (erro_vi >= 50)then
  write(20,*)'Arquivo ',nome
  write(20,*)'Erro_vi: ',erro_vi,'%'
endif

if (erro_ir >= 50)then
  write(20,*)'Arquivo ',nome
  write(20,*)'Erro_vi: ',erro_ir,'%'
endif
```

```
soma_vi = 0
soma_ir = 0
noturno = 0
!-------------------------------------------
!----------Correção de Erros
do i=1,1784
 do j=1,1180
   !Verifica Primeira Linha
   if(j==1)then
    if((mat_erro_vi(i,j) == 1).and.(mat_erro_vi(i,j+1) == 0))then
      vi(i,j) = vi(i,j+1)
      mat_erro_vi(i,j) = 1
      soma_vi = soma_vi + 1
    endif
    if((mat_erro_ir(i,j) == 1).and.(mat_erro_ir(i,j+1) == 0))then
      ir(i,j) = ir(i,j+1)
      mat_erro_ir(i,j) = 1
      soma_ir = soma_ir + 1
    endif
   endif

   !Verifica Última Linha
   if(j==1180)then
    if((mat_erro_vi(i,j) == 1).and.(mat_erro_vi(i,j-1) == 0))then
      vi(i,j) = vi(i,j-1)
      mat_erro_vi(i,j) = 1
      soma_vi = soma_vi + 1
    endif
    if((mat_erro_ir(i,j) == 1).and.(mat_erro_ir(i,j-1) == 0))then
      ir(i,j) = ir(i,j-1)
      mat_erro_ir(i,j) = 1
      soma_ir = soma_ir + 1
    endif
   endif

   !Verifica regiões fora dos extremos
   if((j /= 1).and.(j /= 1180))then
    if((mat_erro_vi(i,j) == 1).and.(mat_erro_vi(i,j+1) == 0).and.(mat_erro_vi(i,j-1) == 0))then
      vi(i,j) = nint((vi(i,j+1) + vi(i,j-1))/2)
      mat_erro_vi(i,j) = 1
      soma_vi = soma_vi + 1
    elseif((mat_erro_vi(i,j) == 1).and.(mat_erro_vi(i,j+1) == 0))then
        vi(i,j) = vi(i,j+1)
        soma_vi = soma_vi + 1
        mat_erro_vi(i,j) = 1
    elseif((mat_erro_vi(i,j) == 1).and.(mat_erro_vi(i,j-1) == 0))then
        vi(i,j) = vi(i,j-1)
        soma_vi = soma_vi + 1
        mat_erro_vi(i,j) = 1
    endif

    if((mat_erro_ir(i,j) == 1).and.(mat_erro_ir(i,j+1) == 0).and.(mat_erro_ir(i,j-1) == 0))then
      ir(i,j) = nint((ir(i,j+1) + ir(i,j-1))/2)
      mat_erro_ir(i,j) = 1
      soma_ir = soma_ir + 1
    else if((mat_erro_ir(i,j) == 1).and.(mat_erro_ir(i,j+1) == 0))then
        ir(i,j) = ir(i,j+1)
        soma_ir = soma_ir + 1
        mat_erro_ir(i,j) = 1
    else if((mat_erro_ir(i,j) == 1).and.(mat_erro_ir(i,j-1) == 0))then
        ir(i,j) = ir(i,j-1)
        soma_ir = soma_ir + 1
        mat_erro_ir(i,j) = 1
    endif
   endif
 enddo
enddo
```

```
print*,"Fim das Correções de Erros"
call check(nf90_put_var(ncid, ir_varid, ir))
call check(nf90_put_var(ncid, vi_varid, vi))
write(30,*)"Erro_corrigido_vi: ",((soma_vi*100)/2105120)
write(30,*)"Erro_corrigido_ir: ",((soma_ir*100)/2105120)
soma_ir = 0
soma_vi = 0
call check(nf90_close(ncid))
!---------------------------------------------
 deallocate (vi)
 deallocate (ir)
 deallocate (ang_sol)
 deallocate (lats)
 deallocate (lons)
enddo !Final arquivo img (lista_imagens)

 close(10)
 close(20)
 close(30)
 contains
!---------------------------------------------
 subroutine check(status)
   integer, intent (in) :: status
   if(status /= nf90_noerr) then
     print *, trim(nf90_strerror(status))
     stop 2
   end if
 end subroutine check
!*************************************************
end program
```

Figura B.3 – Código para calcular o máximo e mínimo com percentil.

```
Program geraPonto
use netcdf

character*80  dire
character*18  arq
character*12  date
character*200 ncdf, CTRANS
character*4   hor, hora_img, hora_padrao, ano,vH
character*3   mes
character*3   P0, P100
integer ::    indMax, indMin, hora, time
integer ::    ir_varid, vi_varid, ang_sol_varid, ang_sat_varid, ang_sat_sol_varid
integer ::    max_varid, min_varid, soma
integer ::    ndims_in, nvars_in, ngatts_in, unlimdimid_in
integer ::    lat_varid, lon_varid, ilat, ilon, i, j, ierr
integer ::    hmin,hmax
integer ::    Vhoras(28), VnewH(27)
integer   ::  p95,P,p05, flag
integer*2 ::  vi(1784,1180)

real*8        lats(1180),lons(1784)
real          Pmax,Pmin
double precision t1, t2, mysecond

!integer, dimension(20,1784,1180),allocatable :: MM
integer :: MM(1000,1784,1180)

integer*2 ::  Mmin(1784,1180),Mmax(1784,1180)
integer ::    Mcount(1784,1180),McountE(1784,1180)
integer ::    atual
integer :: lat_dimid, lon_dimid
integer(kind=4),dimension(2) :: dimids
integer, parameter :: NLAT = 1180, NLON = 1784
```

```
character (len = *), parameter :: LAT_NAME  = "latitude"
character (len = *), parameter :: LON_NAME  = "longitude"
character (len = *), parameter :: UNITS     = "units"
character (len = *), parameter :: LAT_UNITS = "degrees_north"
character (len = *), parameter :: LON_UNITS = "degrees_east"
character (len = *), parameter :: max_NAME  = "MAXIMO"
character (len = *), parameter :: min_NAME  = "MINIMO"
character (len = *), parameter :: max_UNITS = ""
character (len = *), parameter :: min_UNITS = ""

data Vhoras
/745,815,845,915,945,1015,1045,1115,1145,1215,1245,1315,1345,1415,1445,1515,1545,1615,1645,1715,1745,1815,1
845,1915,1945,2015,2045,2115/
data VnewH
/800,830,900,930,1000,1030,1100,1130,1200,1230,1300,1330,1400,1430,1500,1530,1600,1630,1700,1730,1800,1830,
1900,1930,2000,2030,2100/

 call getarg(1,mes)
 call getarg(2,ano)
 call getarg(3,P0)
 call getarg(4,P100)

!Calculando Pmáximo
read(P100,'(I)')P
Pmax = (100. - P)/100.

!Calculando Pmínimo
read(P0,'(I)')P
Pmin = (0+P)/100.

!Alterar diretorio
dire  = '/home/brasilsr/jefferson/'//mes//'/'
 !Zerando a matrix principal com eixo de horarios
  do time=1,27
    if(soma > 0)then
       do k=1,soma
        do i=1,1784
         do j=1,1180
            MM(k,i,j) = 0
         enddo
        enddo
       enddo
    endif
    h=1
    soma = 0
    flag = 0
    !Zerando Matrix dos contadores
    do i=1,1784
     do j=1,1180
        Mcount(i,j)  = 0
        McountE(i,j) = 0
        Mmax(i,j)    = 0
        Mmin(i,j)    = 32768
     enddo
    enddo

    open(unit=10,file="/home/brasilsr/jefferson/"//mes//"/lista.dat")
    do while(.true.) !lista
       read(10,100,iostat=ierr)arq
       100 format(26A)
       if(ierr/=0)exit
       ncdf = trim(dire)//arq
       hor = arq(12:15)
       read(hor,'(I10)')hora
       if((hora >= Vhoras(time)).and.(hora < Vhoras(time+1)))THEN
       print*,'Lendo ',arq
       soma = soma+1
       flag = 1
```

```
!----------Open netcdf com dados de Satelite
      call check(nf90_open(ncdf, nf90_nowrite, ncid) )
      call check(nf90_inquire(ncid, ndims_in, nvars_in, ngatts_in, unlimdimid_in))
      call check(nf90_inq_varid(ncid,"latitude" ,  lat_varid))
      call check(nf90_inq_varid(ncid,"longitude",  lon_varid))
      call check(nf90_inq_varid(ncid,"imagem_ir",  ir_varid))
      call check(nf90_inq_varid(ncid,"imagem_vi",  vi_varid))
      call check(nf90_inq_varid(ncid,"angulo_sol", ang_sol_varid))
      call check(nf90_inq_varid(ncid,"angulo_satelite",     ang_sat_varid))
      call check(nf90_inq_varid(ncid,"angulo_satelite_sol", ang_sat_sol_varid))

      call check(nf90_inquire_dimension(ncid,lat_varid,len=ilat))
      call check(nf90_inquire_dimension(ncid,lon_varid,len=ilon))

      call check(nf90_get_var(ncid,lat_varid,lats))
      call check(nf90_get_var(ncid,lon_varid,lons))
      call check(nf90_get_var(ncid,vi_varid,vi))
!---------Valores----------------------
      do i=1,1784
       do j=1,1180
         if(vi(i,j) > 0)then
           MM(h,i,j) = vi(i,j)
           Mcount(i,j) = Mcount(i,j) + 1
         else
           MM(h,i,j) = -99
           McountE(i,j) = McountE(i,j) + 1
         endif
       enddo
      enddo
      h=h+1
      call check(nf90_close(ncid))
      endif
    enddo !end arq lista.dat
    close(10)
!-------------Insertion Sort (FUNCIONANDO)-------------------------
t1 = mysecond()
print*,"->>>>>>> Ordenado"
if ((flag == 1).and.(soma >= 1))then
do i=1,1784
  do j=1,1180
    do h=1,soma
     do k=h+1,(Mcount(i,j)+McountE(i,j))
       if(MM(h,i,j) > MM(k,i,j))then
         atual = MM(h,i,j)
         MM(h,i,j) = MM(k,i,j)
         MM(k,i,j) = atual
       endif
     enddo
    enddo
  enddo
enddo
t2 = mysecond()
print*, "Time Sort ", t2 -t1
!-------------P95-------------------------
print*,"->>>>>>> Calculando P"
do i=1,1784
  do j=1,1180
    if (Pmin /= 0)then
       p05 = NINT(Pmin*Mcount(i,j))
       indMin = (1 + p05)+McountE(i,j)          !Calculando indice
    else
      indMin = 1+McountE(i,j)
    endif
    if(Pmax /= 0)then
      p95 = NINT(Pmax*Mcount(i,j))
      indMax = (Mcount(i,j)+McountE(i,j)) - p95  !Calculando indice
    else
      indMax = Mcount(i,j)+McountE(i,j)
    endif
```

```
 !Calculando Maximo
  if(indMax == Mcount(i,j))then
    Mmax(i,j) = MM(indMax,i,j)
  else
   if(MM(indMax,i,j) == -99)then
     hmax = indMax
     do while(hmax /= Mcount(i,j)+1)
       if(MM(hmax,i,j) /= -99)then
         Mmax(i,j) = MM(hmax,i,j)
         exit
       else
         hmax = hmax + 1
       endif
     enddo
     if(hmax == Mcount(i,j)+1)then
       Mmax(i,j) = -99
     endif
   else
     Mmax(i,j) = MM(indMax,i,j)
   endif
  endif
  if(Mmax(i,j) == 0)then
    Mmax(i,j) = -99
  endif

 !Calculando Minimo
    if(MM(indMin,i,j) == -99)then
      if(indMin == Mcount(i,j))then
       Mmin(i,j) = MM(indMin,i,j)
      else
         hmin = indMin
       do while(hmin /= Mcount(i,j)+1)
         if(MM(hmin,i,j) /= -99)then
           Mmin(i,j) = MM(hmin,i,j)
           exit
         else
           hmin = hmin + 1
         endif
       enddo
       if(hmin == Mcount(i,j)+1)then
         Mmin(i,j) = -99
       endif
      endif
    else
      Mmin(i,j) = MM(indMin,i,j)
    endif
    if(Mmin(i,j) == 0)then
      Mmin(i,j) = -99
    endif

  enddo
enddo

!-----------
!SALVAR
  IF(VnewH(time)<=930) THEN
    WRITE(vH,'(I3)')VnewH(time)
    vH='0'//vH
  ELSE
    WRITE(vH,'(I4)')VnewH(time)
  END IF
!--------------------------------------------------------------------------------
print*,'->>>>>>> Salvando arquivo Máximo e Mínimo das '//vH//'h'
! Define as dimensões.
  CTRANS=trim('/Dados/brasil_SR/modelo2016/apr/maxmin/'//mes//ano//'.'//vH//'_maxmin.nc')
!Abrindo arquivo NETCDF para salvar a saida
  call check(nf90_create(CTRANS,nf90_write,ncid))
```

```
  call check( nf90_def_dim(ncid,LAT_NAME, NLAT, lat_dimid))
  call check( nf90_def_dim(ncid,LON_NAME, NLON, lon_dimid))
  call check( nf90_def_var(ncid, LAT_NAME, NF90_REAL, lat_dimid, lat_varid) )
  call check( nf90_def_var(ncid, LON_NAME, NF90_REAL, lon_dimid, lon_varid) )
  call check( nf90_put_att(ncid, lat_varid, UNITS, LAT_UNITS) )
  call check( nf90_put_att(ncid, lon_varid, UNITS, LON_UNITS) )
  dimids = (/ lon_dimid, lat_dimid /)

  call check( nf90_def_var(ncid, max_NAME, NF90_INT, dimids, max_varid))
  call check( nf90_def_var(ncid, min_NAME, NF90_INT, dimids, min_varid))

  call check( nf90_put_att(ncid, max_varid, UNITS, max_UNITS))
  call check( nf90_put_att(ncid, min_varid, UNITS, min_UNITS))
  call check( nf90_enddef(ncid))

  call check( nf90_put_var(ncid, lat_varid,   lats) )
  call check( nf90_put_var(ncid, lon_varid,    lons) )

  call check( nf90_put_var(ncid, max_varid, Mmax) )
  call check( nf90_put_var(ncid, min_varid, Mmin) )

  call check( nf90_close(ncid))
  flag = 0
 else
   print*,'Não tem imagens das '//vH
 flag = 0
 endif
 print*,''
 print*,''
 enddo !Final time

 print*, '*** Fim ***'
!---------------------------------------------
 contains
!---------------------------------------------
 subroutine check(status)
   integer, intent (in) :: status
   if(status /= nf90_noerr) then
     print *, trim(nf90_strerror(status))
     stop 2
   end if
 end subroutine check
!*****************************************************
end program
```

Figura B.4 – Código para converter dados climatológicos em NetCDF.

```
program cor_erro
 use netcdf

 character*150 NCDF
 character*80  arq
 character*4   ano
 character*2   hora
 character*3   mes

 integer :: lat_varid, lon_varid,ilat,ilon, ncid, ierr
 integer :: temp_varid,umid_varid,albe_varid,visi_varid
 integer :: ndims_in, nvars_in, ngatts_in, unlimdimid_in

 real*8    temp(1784,1180),  albe(1784,1180),  umid(1784,1180),  visi(1784,1180)
 integer*2 tempN(1784,1180), albeN(1784,1180), umidN(1784,1180), visiN(1784,1180)
 real*8 lats(1180), lons(1784)

 integer :: lat_dimid, lon_dimid
```

```
 integer(kind=4),dimension(2) :: dimids
 integer, parameter :: NLAT = 1180, NLON = 1784

 character (len = *), parameter :: LAT_NAME  = "latitude"
 character (len = *), parameter :: LON_NAME  = "longitude"
 character (len = *), parameter :: UNITS     = "units"
 character (len = *), parameter :: LAT_UNITS = "degrees_north"
 character (len = *), parameter :: LON_UNITS = "degrees_east"
 character (len = *), parameter :: temp_NAME  = "temperatura"
 character (len = *), parameter :: umid_NAME   = "umidade"
 character (len = *), parameter :: albe_NAME   = "albedo"

 character (len = *), parameter :: temp_UNITS  = "Kelvin"
 character (len = *), parameter :: umid_UNITS  = "%"
 character (len = *), parameter :: albe_UNITS  = "%"

 call getarg(1,mes)
 call getarg(2,ano)
 print*, "/Dados/brasil_SR/modelo"//ano//"/"//mes//"/clima/lista.dat"
open(unit=10,file="/Dados/brasil_SR/modelo"//ano//"/"//mes//"/clima/lista.dat")
do while(.true.) !lista
  read(10,100,iostat=ierr)arq
  100 format(26A)
  if(ierr/=0)exit

  hora = arq(18:19)

  NCDF = '/Dados/brasil_SR/modelo'//ano//'/'//mes//'/clima/'//arq
  call check(nf90_open(NCDF, nf90_nowrite, ncid))
  call check(nf90_inquire(ncid, ndims_in, nvars_in, ngatts_in, unlimdimid_in))
  call check(nf90_inq_varid(ncid, "longitude", lon_varid))
  call check(nf90_inq_varid(ncid, "latitude" , lat_varid))
  call check(nf90_inq_varid(ncid, "t", temp_varid))
  call check(nf90_inq_varid(ncid, "r", umid_varid))
  call check(nf90_inq_varid(ncid, "al", albe_varid))
  call check(nf90_inquire_dimension(ncid,lat_varid,len=ilat))
  call check(nf90_inquire_dimension(ncid,lon_varid,len=ilon))
  call check(nf90_get_var(ncid,lat_varid,lats))
  call check(nf90_get_var(ncid,lon_varid,lons))
  call check(nf90_get_var(ncid,albe_varid,albe))
  call check(nf90_get_var(ncid,temp_varid,temp))
  call check(nf90_get_var(ncid,umid_varid,umid))
  call check(nf90_close(ncid))

!------- Calculando
 do j=1,ilat
    do i=1,ilon
       if (temp(i,j) < 273.15)then
          tempN(i,j) = 27315
       else
          tempN(i,j) = temp(i,j)*100
       endif

       if(albe(i,j) < 0)then
         albeN(i,j) = -9900
       else
         albeN(i,j) = albe(i,j)*100
       endif

       if(umid(i,j) < 0)then
          umidN(i,j) = 5000
       else
          umidN(i,j) = umid(i,j)*100
       endif
    enddo
 enddo

!---------
  NCDF = '/Dados/brasil_SR/modelo'//ano//'/'//mes//'/prop/'//mes//'_'//ano//'_'//hora//'N'
```

```
   call check(nf90_create(NCDF,nf90_clobber,ncid))
!---------Salvando NC
  call check( nf90_def_dim(ncid,LAT_NAME, NLAT, lat_dimid))
  call check( nf90_def_dim(ncid,LON_NAME, NLON, lon_dimid))
  call check( nf90_def_var(ncid, LAT_NAME, NF90_REAL, lat_dimid, lat_varid) )
  call check( nf90_def_var(ncid, LON_NAME, NF90_REAL, lon_dimid, lon_varid) )
  call check( nf90_put_att(ncid, lat_varid, UNITS, LAT_UNITS) )
  call check( nf90_put_att(ncid, lon_varid, UNITS, LON_UNITS) )
  dimids = (/ lon_dimid, lat_dimid /)

  call check( nf90_def_var(ncid, temp_NAME, NF90_INT, dimids, temp_varid))
  call check( nf90_def_var(ncid, umid_NAME, NF90_INT, dimids, umid_varid))
  call check( nf90_def_var(ncid, albe_NAME, NF90_INT, dimids, albe_varid))

  call check( nf90_put_att(ncid, temp_varid, UNITS, temp_UNITS))
  call check( nf90_put_att(ncid, umid_varid, UNITS, umid_UNITS))
  call check( nf90_put_att(ncid, albe_varid, UNITS, albe_UNITS) )
  call check( nf90_enddef(ncid))

  call check( nf90_put_var(ncid, lat_varid,  lats))
  call check( nf90_put_var(ncid, lon_varid,  lons))  !arrumar
  call check( nf90_put_var(ncid, temp_varid, tempN))
  call check( nf90_put_var(ncid, umid_varid, umidN))
  call check( nf90_put_var(ncid, albe_varid, albeN))
  call check( nf90_close(ncid))
!---------
enddo
 contains
!--------------------------------------------
 subroutine check(status)
   integer, intent (in) :: status
   if(status /= nf90_noerr) then
     print *, trim(nf90_strerror(status))
     stop 2
   end if
 end subroutine check

end program
```

PUBLICAÇÕES TÉCNICO-CIENTÍFICAS EDITADAS PELO INPE

Teses e Dissertações (TDI)

Teses e Dissertações apresentadas nos Cursos de Pós-Graduação do INPE.

Manuais Técnicos (MAN)

São publicações de caráter técnico que incluem normas, procedimentos, instruções e orientações.

Notas Técnico-Científicas (NTC)

Incluem resultados preliminares de pesquisa, descrição de equipamentos, descrição e ou documentação de programa de computador, descrição de sistemas e experimentos, apresentação de testes, dados, atlas, e documentação de projetos de engenharia.

Relatórios de Pesquisa (RPQ)

Reportam resultados ou progressos de pesquisas tanto de natureza técnica quanto científica, cujo nível seja compatível com o de uma publicação em periódico nacional ou internacional.

Propostas e Relatórios de Projetos (PRP)

São propostas de projetos técnico-científicos e relatórios de acompanhamento de projetos, atividades e convênios.

Publicações Didáticas (PUD)

Incluem apostilas, notas de aula e manuais didáticos.

Publicações Seriadas

São os seriados técnico-científicos: boletins, periódicos, anuários e anais de eventos (simpósios e congressos). Constam destas publicações o Internacional Standard Serial Number (ISSN), que é um código único e definitivo para identificação de títulos de seriados.

Programas de Computador (PDC)

São as sequências de instruções ou códigos, expressos em uma linguagem de programação compilada ou interpretada, a ser executada por um computador para alcançar um determinado objetivo. São aceitos tanto programas fonte quanto executáveis.

Pré-publicações (PRE)

Todos os artigos publicados em periódicos, anais e como capítulos de livros.