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

UMA ABORDAGEM PARA A EXECUÇÃO PARALELA DE MODELOS DE SIMULAÇÃO

Angela Mazzonetto

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

Orientador: Prof. Dr. Carlos Amaral Hölbig
Coorientador: Prof. Dr. Marcelo Trindade Rebonatto

Passo Fundo
2016

CIP – Catalogação na Publicação

M478a Mazzonetto, Angela

Uma abordagem para a execução paralela de modelos de simulação / Angela Mazzonetto. – 2016.

66 f. : il. color. ; 30 cm.

Orientador: Prof. Dr. Carlos Amaral Hölbig.

Coorientador: Prof. Dr. Marcelo Trindade Rebonatto

Dissertação (Mestrado em Computação) – Universidade de Passo Fundo, 2016.

1. Processamento paralelo (Computadores). 2. Simulação (Computadores). 3. Programação paralela (Computadores). I. Hölbig, Carlos Amaral, orientador. II. Rebonatto, Marcelo Trindade, coorientador . III. Título.

CDU: 004.272

Catalogação: Bibliotecária Cristina Troller - CRB 10/1430

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

ATA DE DEFESA DO TRABALHO DE CONCLUSÃO DE CURSO DO ACADÊMICO

ANGELA MAZZONETTO

Aos doze dias do mês de janeiro do ano dois mil e dezesseis, às 15 horas, realizou-se, no Instituto de Ciências Exatas e Geociências, prédio B5, da Universidade de Passo Fundo, a sessão pública de defesa do Trabalho de Conclusão de Curso **"Uma abordagem para execução paralela de modelos de simulação"**, de autoria de Angela Mazzonetto, acadêmica do Curso de Mestrado em Computação Aplicada do Programa de Pós-Graduação em Computação Aplicada – PPGCA/UPF. Segundo as informações prestadas pelo Conselho de Pós-Graduação e constantes nos arquivos da Secretaria do PPGCA, a aluna preencheu os requisitos necessários para submeter seu trabalho à avaliação. A banca examinadora foi composta pelos doutores Carlos Amaral Hölbig, Marcelo Trindade Rebonatto, Willingthon Pavan e Adenauer Correa Yamin. Concluídos os trabalhos de apresentação e arguição, a banca examinadora considerou a candidata APROVADA. Foi concedido o prazo de até quarenta e cinco (45) dias, conforme Regimento do PPGCA, para a acadêmica apresentar ao Conselho de Pós-Graduação o trabalho em sua redação definitiva, a fim de que sejam feitos os encaminhamentos necessários à emissão do Diploma de Mestre em Computação Aplicada. Para constar, foi lavrada a presente ata, que vai assinada pelos membros da banca examinadora e pela Coordenação do PPGCA.

Prof. Dr. Carlos Amaral Hölbig
Presidente da Banca Examinadora
(Orientador)

Prof. Dr. Marcelo Trindade Rebonatto
(Coorientador)

Prof. Dr. Adenauer Correa Yamin - UFPEL
(Avaliador Externo)

Prof. Dr. Willingthon Pavan
(Avaliador Interno)

Prof. Dr. Carlos Amaral Hölbig
Coordenador do PPGCA

Dedico este trabalho aos meus pais e minhas irmãs por estarem sempre presentes na minha caminhada me ajudando e apoiando.

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me dado saúde, força e capacidade para superar as dificuldades.

A Universidade de Passo Fundo, seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbro um horizonte de realização de sonhos, fortalecidos pela confiança, por ensinos sólidos no mérito e ética aqui presentes.

Ao meu orientador Dr. Carlos Amaral Hölbig, e meu coorientador Dr. Marcelo Trindade Rebonatto, professores que tenho grande admiração pelo profissionalismo, conhecimentos na área, paciência em auxiliar e me ensinar com dedicação. Muito obrigada pelo suporte, auxílio na elaboração deste trabalho, incentivos e amizade.

A minha querida mãe Neli Feix Mazzonetto, meu querido pai Adilson Antônio Mazzonetto, e minhas amadas irmãs Paula e Julia, pelo apoio, incentivo e amor incondicional, sem eles eu não chegaria tão longe.

Às secretárias Renata Jung e Marcieli Alves, pela eficiência no trabalho que realizam e principalmente a amizade que me dedicaram nestes dois anos de mestrado.

Agradeço também às minhas amigas Jurema dos Santos Silva, Helia Basso e Janne Rocha pelo carinho, que me acompanharam durante esta jornada.

E a todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigada.

UMA ABORDAGEM PARA A EXECUÇÃO PARALELA DE MODELOS DE SIMULAÇÃO

RESUMO

No atual cenário científico e tecnológico muitos fatores contribuem para o aumento do volume de dados que são utilizados para a solução de aplicações reais nas mais variadas áreas. Entre estes dados estão os que são utilizados em modelos de simulação de culturas e de doenças. Os resultados gerados por estes modelos são posteriormente analisados e tem por objetivo auxiliar no processo de tomada de decisão.Porém, com o aumento da granularidade a quantidade de execuções aumenta exponencialmente nas mesmas proporções, a quantidade de dados necessários para serem processados muitas vezes pode resultar em um despendimento maior de tempo, tornando inviável a utilização dos resultados gerados pelos modelos. Por este motivo torna-se indispensável a utilização de ferramentas ou técnicas computacionais de alto desempenho que visem a otimização e a melhora do desempenho computacional da execução destes modelos. Este trabalho definiu uma abordagem de paralelização para a execução paralela dos modelos de crescimento da cultura do trigo CSM-Cropsim: Wheat e de correção estatística Model Output Calibration, aplicado na correção da previsão do tempo gerada pelo modelo Eta do CPTEC-INPE, objetos de estudo da pesquisa. Para estes modelos a abordagem mestre-escravo, utilizando a biblioteca de comunicação MPI, mostrou ser adequada, visto que cada uma das execuções (rodadas do modelo) são independentes dos resultados das outras em ambos os modelos avaliados. Os resultados obtidos com os testes realizados demonstraram-se satisfatórios, e, assim, contribuíram para a execução paralela dos modelos possibilitando, consequentemente, a ampliação de suas áreas de cobertura e/ou da série temporal a ser simulada.

Palavras-Chave: CSM-Cropsim: Wheat, Execução paralela, Model Output Calibration, Modelos de simulação, Processamento paralelo.

AN APPROACH TO PARALLEL EXECUTION OF CROP SIMULATION MODELS

ABSTRACT

In the current scenario of science and technology, many factors contribute to the increased volume of data used in the solution of real applications in different areas, including data used in simulation models of crops and diseases. The results generated by these models are analyzed after with intent to assist in the decision-making process. However, with the increased coverage area or with the expansion of the simulation period, the amount of data needed can often derail the execution and therefore the processing of the results generated by the models. Thus, the use high-performance tools or computational techniques designed for optimizing and improving the execution of these models becomes essential. This work specifies an efficient parallelization method for the parallel execution of growth models of wheat's culture CSM-Cropsim: Wheat and statistical Model Output Calibration correction, applied in the correction of weather forecast generated by the Eta model of the CPTEC-INPE, objects of this research. For these models the master-slave approach, using MPI communication library, proved the most appropriate, since each of the executions are independent in both models evaluated. The results showed the tests were satisfactory and contributed to the efficient parallel execution of the models and consequently to the extension of their coverage areas and/or the time series to be simulated.

Keywords: CSM-Cropsim: Wheat, running parallel, Model Output Calibration, Simulation models, parallel processing.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Com a crescente evolução tecnológica, a presença de novos coletores de dados meteorológicos torna-se comum. Com esses coletores, a massa de dados aumenta consideravelmente, o que ajuda na qualidade da cobertura de dados observados, utilizados como dados de entrada por centros e grupos de pesquisa e demais interessados em utilizá-las em seus ramos de atividade [1].

Uma das áreas de maior aplicação destes dados é a área agrícola, onde modelos de simulação de culturas e doenças os utilizam. Neste caso em específico, entretanto, esta crescente disponibilidade de dados pode gerar dificuldades de execução quando pretende-se utilizar a sua totalidade, fazendo com que o número de execuções exceda o poder de processamento das máquinas dos pesquisadores devido a sua área de abrangência ou ao período a ser simulado. Além dos dados observados, dados gerados por outros modelos como, por exemplo, de previsão climática, também são utilizados.

A necessidade de abranger uma maior região geográfica ou maior período de simulação torna a execução dos modelos repetitiva, pois o mesmo modelo tende a ser executado muitas vezes para abranger estas diferentes regiões, áreas ou períodos. Devido a este fato, é necessária a escolha de alternativas computacionais, como a paralelização, a otimização de código ou a escolha de uma arquitetura computacional paralela, que proporcionem um ganho no tempo de execução destes modelos, principalmente quando eles manipulam uma grande quantidade de dados observados e/ou previstos. A alternativa escolhida por esta pesquisa baseia-se no uso do processamento paralelo, possibilitando a execução dos modelos de simulação por esta abordagem.

O processamento paralelo tem por objetivo a execução de um programa ou tarefas simultaneamente em vários processadores. Contudo, é importante saber quais técnicas serão utilizadas para que a paralelização seja efetivamente eficiente na solução do problema proposto, sendo, para isto, necessário conhecer detalhes sobre o programa que terá sua execução em paralelo.

Entre as maneiras de paralelizar um programa, estão o uso de ferramentas como o MPI, o OpenMP, o framework Hadoop, a linguagem R e seus pacotes, ou bibliotecas desenvolvidas para este propósito. Juntamente com a abordagem paralela, é importante saber o ambiente computacional no qual o programa será executado, podendo ser em *grid* de computadores, em computadores com processadores multicore, em computadores com placa gráfica ou em *clusters* de computadores.

Com o objetivo de apresentar alternativas viáveis de paralelização da execução de modelos de simulação ligados à área agrícola, este trabalho selecionou dois modelos como estudo de caso para validar estas alternativas mediante alguns testes. Estes estudos de caso são compostos por um modelo de simulação de cultura do trigo, chamado de CSM-Cropsim: Wheat [2], utilizado pela Embrapa Trigo e pela UPF, e o modelo de correção estatística Model Output Calibration (MOC) [3], aplicado na correção do resultado da previsão gerada pelo modelo de previsão de tempo Eta 15km [1] desenvolvido e utilizado pelo Centro de Previsão de Tempo e Estudos Climáticos

(CPTEC) do Instituto Nacional de Pesquisas Espaciais (INPE). Portanto, o problema de pesquisa deste trabalho foi o de definir uma estratégia de paralelização da execução destes modelos.

Partindo do problema de pesquisa iniciou-se a elaboração dos capítulos organizadas da seguinte forma: no segundo capítulo, é apresentado o estado da arte sobre trabalhos relacionados com o processamento paralelo e distribuído e suas aplicações nas mais diversas áreas, em especial na área agrícola, foco desta pesquisa. O capítulo três aborda o estudo de caso sobre o modelo de simulação de crescimento do trigo CSM-Cropsim: Wheat, onde são descritos suas características, funcionamento e execução sequencial. No capítulo quatro é abordado o segundo estudo de caso do trabalho, o modelo de correção estatística MOC, aplicado na correção da previsão do tempo gerada pelo modelo Eta 15km do CPTEC/INPE. Neste capítulo, uma descrição do modelo de correção, do modelo Eta e de aspectos de sua execução são apresentados. No capítulo cinco é apresentada a estratégia de paralelização definida e utilizada para a execução paralela do CSM-Cropsim: Wheat e do MOC. No capítulo seis os testes realizados são descritos e seus resultados são apresentados e analisados. Por fim, no capítulo sete, as conclusões do trabalho são apresentadas, juntamente com as perspectivas de trabalhos futuros e as publicações já produzidas à partir deste trabalho.

2. ESTADO DA ARTE

No atual cenário científico e tecnológico, muitos fatores contribuem para o aumento do volume de dados que são utilizados para a solução de aplicações reais nos mais variados campos de aplicação. Bases de dados como, por exemplo, do CPTEC/INPE[1], do Instituto Nacional de Meteorologia (INMET)[2], do Agritempo - Sistema de Monitoramento Agrometeorológico[3], do National Oceanic and Atmospheric Administration (NOAA)[4], do World Development Indicators (WDI)[5], banco de dados governamentais, entre outros, são alguns dos mais diversos repositórios de dados com informações meteorológicas e climáticas disponíveis atualmente. Neste contexto, surgem barreiras computacionais quanto ao processamento e a análise referente a extração de informações relevantes sobre estes dados. Um exemplo desta situação é a crescente preocupação com a utilização de grande quantidade de dados meteorológicos e climáticos com foco na estimativa de possíveis impactos nos sistemas de produção de culturas [4].

Fundamentando este assunto, Pavan [5] afirma que existe um aumento da demanda por alimentos, de ameaças ao ambiente, da pressão sobre os recursos terra/água, da globalização dos mercados e que muitos sistemas de produção agrícola no mundo estão rapidamente sendo modificados. Produtores, indústrias e autoridades devem responder rapidamente a esses eventos globais, cada vez mais complexos, assim como às condições agroecológicas e socioeconômicas locais. Ainda conforme Pavan [5], os modelos de simulação apresentam-se como alternativas viáveis para auxiliar nesta questão, pois ajudam os responsáveis pelas tomadas de decisões a conhecerem as consequências e os riscos prováveis em suas decisões. Contudo, a grande quantidade de dados que são processados nestes modelos de simulação que possibilita uma cobertura maior da área de interesse faz com que ocorra um aumento no tempo de execução dos modelos, inviabilizando, muitas vezes, a sua utilização efetiva, tendo em vista necessidades computacionais.

Portanto, a importância da utilização dos modelos de simulação está no fato de que eles podem prever inúmeras ocasiões da realidade como, por exemplo, previsão de impactos ecológicos, hidrológicos e geomorfológicos e previsões para os setores agrícola e de energia [5]. Utilizando-se de dados de prognósticos o modelo de simulação de culturas, um dos focos deste trabalho, podem prever impactos de doenças nas culturas ou seu processo de crescimento, gerando resultados para possíveis cenários da produtividade em determinadas épocas de plantio. Conforme Lazzaretti [6], a vários anos, os modelos de simulação de culturas vêm auxiliando os profissionais da área agrícola em como melhorar o rendimento da colheita.

Associados aos modelos de simulação de culturas estão os modelos de simulação da previsão do tempo, pois as informações por eles prestadas são utilizadas como dados de entrada nos

[1] http://www.cptec.inpe.br/
[2] http://www.inmet.gov.br/portal/
[3] http://www.agritempo.gov.br/
[4] http://www.noaa.gov/
[5] http://data.worldbank.org/

modelos de cultura e de doenças para auxiliar na tomada de decisão a nível de doenças de várias espécies agrícolas. Contudo, os modelos de simulação, algumas vezes, podem não gerar informações inteiramente precisas, causadas pela qualidade dos dados gerados e utilizados como entrada, necessitando, portanto, de correções. Exemplo deste processo de correção é descrito no trabalho de Ercan [7], onde é avaliado uma ferramenta de calibração para o modelo de simulação Water Assessment Tool (SWAT) voltado para bacias hidrográficas. Portanto, uma das alternativas para que hajam resultados mais precisos para os modelos de simulação de culturas e doenças é a necessidade de que os dados meteorológicos previstos estejam o mais próximo da assertividade.

Devido à importância da utilização efetiva e, também, em tempo hábil, das informações geradas pelos modelos, torna-se indispensável a utilização de ferramentas ou técnicas computacionais de alto desempenho, visando a otimização e a melhora do desempenho computacional da execução destes processos.

Várias ferramentas podem ser utilizadas para suprir estas necessidades, desde as já tradicionais para computação paralela como o *Message Passing Interface* (MPI[6]), utilizado no trabalho do Jordi [8] em que o modelo de simulação Stony Brook Parallel Ocean Model (sbPOM) utiliza a comunicação de troca de mensagens do MPI, confirmando eficiência no desempenho usando até 2048 processadores; o *Open Multi-Processing* (OpenMP[7]), até *frameworks* como o Hadoop[8] ou linguagens de programação como o R[9]. Obviamente, o uso destas ferramentas deve estar aliado a escolha correta de uma abordagem eficiente de paralelização.

Devido a estes fatores, a melhora no tempo de execução, obtida por meio da paralelização da execução dos modelos de simulação desenvolvidos em linguagens como R, Fortran e Java, pode contribuir de forma significativa para a análise de séries temporais de dados observados mais amplas e variadas. Para possibilitar uma quantidade maior de rodadas de execução ou para ampliar a área de cobertura dos modelos de forma eficiente e em um tempo computacional adequado às necessidades das aplicações que desta análise dependem para ser solucionadas.

Como exemplo de paralelização na área agrícola, o trabalho de Zhao [9] relata que o processamento paralelo auxiliou no modelo de simulação do processo de produção de trigo da Autrália que, com a combinação da estrutura de um Grid de computadores, melhorou o desempenho da simulação que levaria 30 anos para ser executada em um único computador. Com a abordagem de computação de alto desempenho High-performance computing (HPC), a execução da modelagem foi concluída dentro de 10,5 dias, um aumento de velocidade de mais de 1000 vezes.

Outros trabalhos envolvendo modelos de simulação são impulsionados a utilizar a programação paralela para manipular uma grande quantidade de dados. Entre as diversas áreas que ela está sendo aplicada estão a Física, a Biologia, a Química, a Estatística, a Geografia, entre outras. Estas aplicações tratam da paralelização de modelos geoestatísticos, técnicas de computação paralela em clusters de computadores [10], em sistemas multicore [11] e grid com R [10],

[6] http://www.mcs.anl.gov/research/projects/mpi/
[7] http://openmp.org/wp/
[8] http://hadoop.apache.org/
[9] http://cran.r-project.org/

paralelização de códigos em R com OpenMPstyle [12], cálculos de bioestatística paralelas em um supercomputador [13], utilização do paradigma MapReduce para R [14], utilização de ferramentas R para computação em Grid [15], uso do pacote paralelo Parallel R [16], o emprego de biblioteca R para reduzir complexidade da computação de alto desempenho [17], sistema de simulação agrícola paralelizado para executar em Grid de computadores [9], avaliação de ferramentas para HPC [18], desenvolvimento de plataformas escaláveis utilizando algoritmos desenvolvidos em R [19], uso do R em conjunto com a linguagem C++ [20], abordagem de pacotes e ferramentas para paralelização em R [13], apresentação da ferramenta JRBridge [19], desafios na utilização do R [14], apresentação do pacote Sprint [13], execução de algoritmos de R na infraestrutura de nuvem e abordagem sobre o projeto Bioconductor [13].

Os trabalhos que relatam estas aplicações, que visam o uso de diferentes modelos de simulação e de técnicas ou abordagens de paralelização, são resumidamente apresentados a seguir.

2.1 APLICAÇÕES PARALELAS EM R

Petrou et al [13] abordam que a linguagem R e os pacotes do projeto Bioconductor [10] favorecem para muitos bioestatísticos o processamento de dados de microarrays. Porém, a quantidade de dados produzidos por algumas análises atingiu os limites de muitas infraestruturas computacionais comuns em bioinformática e sistemas de computação de alto desempenho ofereceram uma solução para esta questão. O Simple Parallel R INTerface (SPRINT) é um pacote que fornece à bioestatística um fácil acesso aos sistemas de computação de alto desempenho e permite a adição de funções paralelizadas para R. O acesso a supercomputadores nem sempre é possível e por isso este trabalho compara o desempenho da execução do SPRINT em um supercomputador com *benchmarks* em uma variedade de plataformas, incluindo recursos de nuvem e uma máquina desktop comum com capacidades de multiprocessamento.

Hoffmann e Lange [21] descrevem a ferramenta de software P2BAT que fornece uma implementação paralela e amigável das ferramentas PBAT e FBATs para estudos de associação do genoma em R. A ferramenta de análise do genoma é totalmente automatizada e pode ser executada paralelizada. P2BAT está totalmente documentado e contém ferramentas de produção gráfica.

Eddelbuettel e Sanderson [20] afirmam que a linguagem R demonstrou pontos fortes para o desenvolvimento interativo de algoritmos estatísticos, bem como modelagem de dados e visualização. Sua implementação atual tem um interpretador em seu núcleo que pode resultar em uma melhora no desempenho da execução. Em contraste, a linguagem C++ não tem recursos de visualização, manipulação de álgebra linear ou algoritmos estatísticos. No entanto, os programas são convertidos em código de máquina de alto desempenho. Um novo método, usando o pacote Rcpp [11], em conjunto com a biblioteca matricial Armadillo C++, evita possíveis desvantagens na velocidade das execuções em R. Além das vantagens de desempenho inerentes de código compilado, Armadillo fornece um

[10] https://www.bioconductor.org/
[11] http://www.rcpp.org/

modelo fácil de usar que possibilita a partilha automática de várias operações de álgebra linear, que pode levar a novos aumentos de velocidade. Com a ajuda do Rcpp e do Armadillo, torna-se simples a conversão de algoritmos de álgebra linear de R e C++. Os algoritmos podem manter a estrutura geral bem como a leitura, mantendo uma ligação com o ambiente do R.

Venkataraman et al [14] salientam que é inviável escrever algoritmos complexos em modelos paralelos, como em MapReduce. Muitos destes algoritmos tem natureza operatória e são implementados para realizar cálculos incrementais, nenhum dos quais são eficientemente suportados por estruturas atuais. Os autores argumentam também que, as linguagens baseadas em matriz, como R, são ideais para expressar estes algoritmos, e eles podem ser estendidos para processamento na nuvem. Portanto, os autores apresentam os desafios e as abstrações de estender R.

Xie et al [19] afirmam que a demanda por plataformas de processamento paralelo de dados altamente escaláveis está aumentando devido a uma explosão no número de aplicações de dados em massa, em escala intensiva, tanto na indústria quanto nas ciências. Realizar computação estatística sobre enormes repositórios de dados representa um desafio significativo para o software estatístico e para a infraestrutura computacional existente. Após a análise de várias infraestruturas computacionais de código aberto e suas APIs e paradigmas de programação, os resultados mostraram que a maioria deles são com base Java Virtual Machine (JVM), e suas APIs são dadas como interfaces Java ou classes abstratas. Neste trabalho, os autores propuseram uma estrutura genérica JRBridge, que pode integrar R e infraestruturas computacionais baseados em JVM, gerando automaticamente código Java invólucro em torno do código R e manipulação de conversão de tipo. Com o plugin Hadoop *Distributed File System*, os autores afirmam que é possível trazer uma forma de armazenar e acessar os conjuntos de dados com milhões de objetos e com o plugin MapReduce um ambiente natural para codificar algoritmos MapReduce em R.

Kumar et al [11] descreve que o aumento exponencial na geração e coleta de dados levou-nos a uma nova era de análise de dados e extração de informações. Os sistemas convencionais baseados em processadores de uso geral são incapazes de manter o ritmo computacional com os requisitos de pesadas de técnicas de mineração de dados. Processadores de alto desempenho, como GPUs e FPGAs, têm o potencial para lidar com grandes cargas de trabalho computacional. Estes autores apresentam uma estrutura escalável que visa proporcionar uma plataforma para o desenvolvimento e utilização de aplicações de mineração de dados de alto desempenho em plataformas heterogêneas. A estrutura inclui uma infraestrutura de software e uma biblioteca de alto desempenho. Além disso, inclui uma variedade de otimizações que aumentam o rendimento das aplicações. O quadro abrange múltiplas tecnologias, incluindo R, GPUs, CPUs multicore, MPI e parallel-netCDF, e aproveita suas capacidades para cálculos de alto desempenho. Também é apresentado pelos autores vários aplicativos que oferecem ganho de desempenho significativo com a utilização da GPU. O *framework* está disponível como um pacote de software que pode ser facilmente integrado no ambiente de programação R.

Schmidberger et al [10] destacam que muitas áreas de pesquisa estatística estão se deparando com um rápido crescimento no tamanho dos conjuntos de dados. Avanços metodológicos

conduzem ao aumento do uso de simulações. Uma abordagem comum é usar a computação paralela. Neste trabalho é apresentada uma visão geral das técnicas de computação paralela em clusters de computadores, em sistemas multicore e na computação em grid com R.

Li et al [22] abordam que as linguagens de script como R e Matlab são amplamente utilizadas no processamento de dados científicos. Como o volume de dados e a complexidade das tarefas de análise tendem a crescer, o processamento de dados sequenciais usando essas ferramentas, muitas vezes, tornam-se o gargalo em experimentos científicos. Os autores descrevem o pR, um *framework* para a paralelização automática e transparente da linguagem R. Eles propuseram várias técnicas: (1) aplicação da tecnologia de paralelização para a execução (2) análise de código incrementais assistida com os resultados da avaliação, e (3) a paralelização de tempo de execução de acessos a arquivo. A estrutura apresentada não requer nenhuma modificação no código-fonte ou a implementação R subjacente.

Vera e Suppi [23] abordam que a computação paralela está se tornando essencial nos dias de hoje para a análise de dados em várias disciplinas. A fim de melhorar o tempo de processamento ferramentas computacionais e recursos de computação adequados são necessários. Foi implementada uma solução em linguagem R que permite a execução de *loops* paralelos em uma rede de computadores heterogênea.

Stokely, Rohani e Tassone [24] demonstram a utilidade da infraestrutura computacional paralela para estatística computacional utilizando o paradigma MapReduce para R. Essa estrutura permite que os usuários escrevam cálculos em uma linguagem de alto nível que depois são divididos e distribuídos para as tarefas serem trabalhadas nos *datacenters* do Google.

Jiang et al [12] descrevem que, como os programas em R estão ligados às entradas de dados, o crescimento exponencial dos dados disponíveis torna a computação de alto desempenho com R indispensável. Para facilitar o processo de escrita de programas paralelos em R, a transformação de código de um programa sequencial para uma versão paralela seria muito conveniente aos usuários de R. Neste trabalho foi apresentado a paralelização de códigos em R com OpenMPstyle e são demonstradas as vantagens significativas dos mecanismos *on-the-fly* para a computação paralela de dados.

Das et al [25] descrevem que muitas empresas modernas estão coletando dados no nível mais detalhado possível, resultando na criação de repositórios de dados que variam de terabytes a petabytes de tamanho. A capacidade de aplicar métodos de análise estatísticas sofisticados nestes dados está se tornando essencial para a competitividade de mercado. Esta necessidade de realizar uma análise profunda sobre enormes repositórios de dados representa um desafio significativo para os sistemas de gestão de software estatístico. Por um lado, o software estatístico fornece funcionalidade avançada para modelagem e análise de dados, mas pode lidar com quantidades limitadas de dados. Por outro lado, os sistemas de gerenciamento de dados, tais como sistemas baseados em MapReduce, podem ser escalados para petabytes de dados, porém oferecendo a funcionalidade de análise insuficiente. Os autores relatam experiências na construção de Ricardo, uma plataforma escalável para análises de grandes quantidades de dados. Ricardo faz parte do projeto Analytics

eXtreme Plataform (XAP) na IBM Almaden Research Center, e trata sobre uma decomposição da análise de dados de partes de algoritmos executadas pelo sistema de análise estatística R e partes tratadas pelo sistema de gerenciamento de dados Hadoop. Esta decomposição tenta minimizar a transferência de dados através das fronteiras do sistema.

Samatova et al [16] falam da atual produção de grande quantidade de dados. Questões fundamentais sobre a biologia permanecem ocultas em grande parte destes dados. Portanto, o objetivo do trabalho foi fornecer uma estrutura escalável de análise estatística de dados para ajudar os cientistas a realizar análises interativas destes dados brutos e, posteriormente, extrair o conhecimento desejado. A estrutura foi desenvolvida com um pacote paralelo *open source* de análise estatística, chamado Parallel R (ou simplesmente parallel), que permite aos cientistas empregar uma ampla gama de rotinas de análise estatística em alto desempenho, sem ter que lidar com a complexidade da paralelização dessas rotinas.

Em Mitchell et al [17] os autores salientam que, a quantidade de dados que podem ser obtida em experiências aumentou significativamente, fazendo com que as análises estatísticas ficassem demoradas, ou que não fosse possível realizá-las. A computação de alto desempenho (HPC) é uma solução para estes problemas mas pode ser complexa para o usuário final. O Simple Parallel R Interface (SPRINT) é uma biblioteca R que visa reduzir a complexidade do uso de sistemas HPC, fornecendo funções paralelas em R.

Padberg, Mirold [15] apresentam uma plataforma que apoia a paralelização automática de programas de pesquisa durante a execução. A paralelização ocorre de forma totalmente transparente para o usuário. Técnicas diferentes de paralelização podem ser aplicadas como módulos, ligados à plataforma, e combinados uns aos outros. Partes paralelizadas com sucesso do programa R são executados em um processador multicore; os resultados e as partes sequenciais restantes são realimentados no interpretador padrão R e avaliados para a conclusão. Desta forma, um usuário R pode se beneficiar de desempenho multiprocessador sem escrever uma única linha de código paralelo. Os autores [15] acreditam que esta pesquisa, bem como as ferramentas desenvolvidas por eles, podem ser usadas como uma base para o progresso no sentido da utilização dos modelos existentes, bem como na aplicação da computação de alto desempenho, tecnologias e infraestruturas nesta área.

2.2 USO DE TÉCNICAS DE COMPUTAÇÃO PARALELA APLICADAS A MODELOS DE SIMULAÇÃO

Os autores em Elliott et al [26] apresentam um *framework* para simulações paralelas do impacto no clima, o *parallel System for Integrating Impact Models and Sectors* (pSIMS). Este *framewok* é composto por ferramentas para inserir e converter grandes quantidades de dados para um tipo de dados versátil, com base em uma grade de geospatial comum; ferramentas para traduzir este tipo de dados em formatos personalizados para os modelos de base local; um *framewok* paralelo escalável usando qualquer um das várias arquiteturas computacionais: clusters, supercomputadores,

redes distribuídas ou nuvens, ferramentas e padrões de dados para reformatação de saídas para tipos de dados comuns para análise e visualização, e metodologias para agregar estes tipos de dados para escalas espaciais. Este *framework* tem sido usado na estimativa do rendimento das culturas e do impacto do clima em avaliações nos EUA e nos níveis globais em alta resolução.

Os mesmos autores em [27] apresentam os elementos técnicos do pSIMS, os resultados de uma avaliação de impacto climática e exercício de validação que envolviam grandes computações paralelas sobre *Extreme Science and Engineering Discovery Environment* (XSEDE).

Bryan et al [28] descrevem que a agricultura terá que produzir mais utilizando menos terra e de uma forma mais sustentável. Porém, em muitos lugares, os rendimentos da colheita está baixa. Os autores quantificaram a capacidade de gestão agrícola para aumentar a produção de trigo na Austrália. Utilizaram o Sistema de Simulação de Produção Agrícola (APSIM) com a abordagem da computação paralela em Grid para simular o impacto na agricultura avaliando a influência de variáveis de gestão ambientais na produtividade do trigo.

Bryan, King, Zhao [18] afirmam que é improvável que as demandas computacionais possam ser simuladas pelo Sistema de Informação Geográfica tradicional (GIS). Então, foi realizada uma avaliação de uma gama de ferramentas de computação de alto desempenho, tanto hardware como software. As vantagens de desempenho foram quantificadas e, para cada ferramenta, foram apresentados os resultados. As ferramentas de código aberto, como por exemplo a ferramenta Python, quando aplicado por meio de um espectro de recursos HPC, oferece melhorias para a modelagem. Ao reduzir a barreira computacional, a HPC pode levar a uma mudança de patamar na sofisticação da modelagem. No entanto, a migração para novos ambientes de hardware e software também tem custos significativos. Custos e benefícios de HPC são discutidos e ferramentas de código são fornecidas para ajudar na migração para HPC visando transformar a capacidade de enfrentar os desafios globais por meio da avaliação integrada da modelagem.

Zhao et al [9] descrevem que a solução dos complexos desafios globais no sistema da terra, como a segurança, a alimentação e a energia, requerem informações sobre a gestão dos sistemas agrícolas com uma alta resolução espacial e temporal sobre extensões continentais ou mundiais. No entanto, a capacidade de computação continua a ser uma barreira para a larga escala e a solução de modelagem agrícola. Para modelar a produção de trigo em regiões de cultivo da Austrália com uma resolução alta, eles desenvolveram uma abordagem híbrida que combina computação paralela e processamento em Grid. A abordagem híbrida distribui tarefas através de um conjunto heterogêneo de computadores em Grid que utiliza totalmente todos os recursos de computadores dentro dos conjuntos. Eles realizaram simulações que levariam mais de 30 anos em um único computador. A abordagem utilizou recursos ociosos existentes de computação em toda a organização e elimina a necessidade de traduzir os modelos baseados no Windows para outros sistemas operacionais para execução em clusters de computadores. Há, no entanto, numerosos desafios computacionais que precisam ser abordados para o uso efetivo dessas técnicas e ainda há várias áreas em potencial para melhorar ainda mais o desempenho.

Liu et al [29] afirmam que boas práticas de gestão *Beneficial Management Practices* (BMPs) são medidas importantes para redução de fonte não pontual (NPS) da poluição agrícola. No entanto, a seleção de BMPs para a colocação de uma bacia hidrográfica requer otimizar os recursos disponíveis para maximizar possíveis benefícios de qualidade da água. Devido à sua natureza interativa, a otimização normalmente leva muito tempo para atingir os resultados e isto não é desejável na prática. Neste estudo, um modelo de otimização que consiste em um algoritmo genético multi-objetivo, em combinação com a água no solo e ferramenta de avaliação *Assessment Tool* (SWAT), e a técnica de computação paralela, foi desenvolvido e testado na bacia Fairchild Creek no sul de Ontário no Canadá. Os dois objetivos foram de minimizar os custos e de maximizar BMPs e realizar a redução da carga total. A computação paralela permitiu a execução de vários modelos da SWAT simultaneamente e pode reduzir o tempo de otimização significativamente para atingir o objetivo. As práticas geradas podem ser usadas para atingir as metas de qualidade da água desejados com custo mínimo e para apoiar a gestão de bacias hidrográficas e a formulação de políticas.

Yalew et al [30] salientam que o crescente interesse em modelos de maior escala espacial e temporal e os dados de entrada de alta resolução tem um preço: a necessidade de uma maior demanda computacional. Os recentes avanços em computação distribuída, como a infraestrutura Grid, têm proporcionado mais uma oportunidade para este esforço. No interesse de ganhar eficiência computacional, os autores desenvolveram ferramentas e técnicas paralelas para execução em Grid, permitindo que a aplicação do modelo *Soil and Water Assessment Tool* (SWAT) seja executada nos Enabling Grids (EGEE) para projetos de *E-Science* na Europa. Posteriormente, eles conduziram simulações experimentais com vários modelos hidrológicos de escala temporal e espacial na infraestrutura de Grid.

Em Chen et al [31] é descrito o modelo de simulação *Cellular Potts Model* (CPM) que tem sido utilizado em uma ampla variedade de simulações da biologia. No entanto, o CPM é implementado atualmente usando um algoritmo sequencial que limita o tamanho das simulações. Neste trabalho os autores apresentaram um algoritmo paralelo do CPM para simulações da morfogênese, que inclui as propriedades da célula. O algoritmo usa estruturas de dados adequadas e subgrids para paralelização. Por meio de comunicação, as propriedades das células são sincronizadas e simuladas em diferentes nodos de processos.

Os autores descrevem que operações computacionais em modelagem de sistemas de energia hidrelétrica em grande escala, especificadamente na China, são um desafio computacional pois, conforme o aumento do número de reservatórios, o tempo computacional aumenta exponencialmente [32]. Algoritmos de programação dinâmica são uma opção para resolução deste problema, no entanto, o tempo computacional ainda aumenta exponencialmente com o aumento do número de reservatórios. A partir disto, um algoritmo paralelo com programação dinâmica que se baseia no *framework* fork/join, foi proposto com o intuito de melhorar a eficiência computacional para operação de longo prazo dos sistemas de energia hidrelétrica. O algoritmo foi testado usando um sistema de energia hidrelétrica em cascata, localizado no Rio Lancang na Chinaet al [32].

Um exemplo de trabalho focado na calibração de modelos como, foi proposto em Ercan et al [7], onde é avaliado uma ferramenta criada para a calibração da ferramenta Water Assessment Tool (SWAT), construída usando o ambiente de Windows Azure Cloud e para uma versão paralela do método *Dynamically Dimensioned Search* (DDS), modificado para executar neste ambiente. A ferramenta de calibração foi testada em seis cenários de modelo construídos para três bacias hidrográficas para aumentar o tamanho de cada duração de simulação de 2 e 10 anos.

Em Lieber e Wolke [33] é descrita uma estratégia de executar os códigos individuais simultaneamente em um conjunto de processadores com os códigos distribuídos e independentes. No entanto, é importante melhorar o equilíbrio de carga de trabalho entre os códigos para atingir uma alta eficiência em computadores paralelos. Nesta abordagem, todos os processadores calculam alternadamente ambos os modelos, e a carga é distribuída igualmente entre eles. Em geral, podem ocorrer variações de carga em alguns tipos de modelos. Os autores concluíram que sistemas que utilizam essas técnicas podem tirar benefício da abordagem descrita.

No trabalho em Rao [34], um modelo hidrodinâmico é testado sobre uma plataforma paralela e seu desempenho é observado em um supercomputador tradicional e um cluster. O código paralelo é baseado em princípios de decomposição de domínio e usa a biblioteca MPI para todas as comunicações entre os processadores sendo os dados são distribuídos igualmente. Como conclusão os autores afirmam que, o código paralelo final pode acelerar a solução para o estado desejado sem perder qualquer precisão na solução final.

Mais uma abordagem geral de paralelização de um modelo com utilização de memória distribuída baseando-se na biblioteca MPI em um grid foi desenvolvida no trabalho descrito por Smiatek [35]. A versão paralela atinge uma diminuição de tempo quase linear na execução quando o número de processadores é aumentado. No entanto, com alguns números de processadores, a eficiência sofre um balanceamento de carga desigual, causado por um número diferente de cálculos no interior das células do grid.

Em Teijeiro et al [36], os modelos que fazem a simulação da dinâmica das partículas estão entre os mais importantes mecanismos para estudar o comportamento das moléculas em um meio sobre condições específicas de temperatura e densidade. Vários modelos podem ser usados para calcular eficientemente as forças que atuam sobre cada partícula e também as interações entre elas. Este trabalho apresenta o projeto e execução de um código de simulação paralela para o movimento browniano das partículas em um fluído. Seguiram-se duas abordagens diferentes de paralelização: (1) usando a tradicional programação de passagem de mensagens memória com MPI e (2) usando o modelo de programação *Partitioned Global Address Space* (PGAS), orientado para sistemas de memória compartilhada/distribuídos de híbridos, com a linguagem Unified Parallel C (UPC). Diferentes técnicas para a distribuição de trabalho e de decomposição de domínio são analisadas em termos de eficiência e de programação, a fim de selecionar a estratégia mais adequada. Resultados de desempenho em um supercomputador usando até 2048 núcleos também foram apresentados para ambos MPI e UPC.

Topografia é uma das mais importantes propriedades em impactos ecológicos, hidrológicos e geomorfológicos em processos ativos em uma paisagem [37]. Tesfa et al [37] desenvolveram um modelo de profundidade de solo com base na topografia e variáveis de cobertura de terra resultando em um conjunto de medidas hidrológicas de proximidade (HPMs) de um modelo *Digital Elevation Model* (DEM) como variáveis em potencial que são explicativas para a profundidade do solo. Anteriormente, estes HPMs foram calculados utilizando algoritmos recursivos, que sofriam de problemas de estouro de pilha quando usados para processar grandes conjuntos de dados, limitando o tamanho de DEMs a serem analisados. Para superar esta limitação, os autores desenvolveram uma abordagem paralela usando o MPI, projetada para aumentar o tamanho e a velocidade com que estes HPMs são computados. Os algoritmos paralelos de HPM particionam as entradas de dados em listas que são atribuídas para cada processo. Cada um desses processos, em seguida, usa uma estrutura de dados fila para ordenar o processamento de células para que cada célula visitada apenas uma vez e a comunicação entre processos é tratadas de forma eficiente pelo padrão MPI.

Yablonsky, Ginis, Thomas [38] apresentam o uso do MPI no modelo Princeton Ocean Model for Tropical Cyclones (MPIPOM-TC), criado na University de Rhode Island (URI). MPIPOM-TC é derivado de uma combinação da versão paralelizada de Princeton Ocean Model (POM), chamado Stony Brook Parallel Ocean Mode (sbPOM) e da sessão não-paralelizada POM da URI para ciclones tropicais (POM-TC), que tem sido usado por muitos anos como o componente do oceano dos modelos do NOAA e para modelos de previsão de furacões operacionais da Marinha dos EUA.

Por fim, no trabalho descrito por Zhang et al [39], é realizado o desenvolvimento de um software de computação paralela para melhorar a eficiência de calibração de modelos de simulação de bacias hidrográficas, pois é necessário executar iterativamente complexos modelos para calibração. É baseado em Python, utilizando o pacote para computação paralela PP-SWAT, para calibração eficiente do modelo solo e a ferramenta de avaliação da água SWAT. Resultados de teste em um cluster de computador mostraram que o PP-SWAT pode alcançar um aumento de velocidade dependendo da complexidade do modelo. Aumentando a contagem de processador além de certo limite não necessariamente melhora a eficiência, porque a concorrência de intensificação do recurso pode resultar em um gargalo de I/O. A eficiência alcançada por PP-SWAT também o torna prático para implementar vários esquemas de ajuste do parâmetro operando em diferentes escalas no tempo disponível, e supervisão cuidadosa do seu poder deve ser exercido para atingir resultados de calibração fisicamente significativas.

Com base nesta sessão, observa-se que existem inúmeros modelos de simulação, nas mais diversas áreas de aplicação, que, aliando a paralelização, mesmo com diferentes metodologias, vêm trazendo fatores significativos para o melhor aproveitamento dos dados processados e, ao mesmo tempo, uma melhora no desempenho computacional das aplicações envolvidas. Devido a estes fatores, um estudo sobre as características dos dois modelos utilizados como estudo de caso nesta pesquisa, foi realizado com o intuito de identificar, as estratégias de paralelização de suas execuções que melhor se adequam aos mesmos. Esta descrição e estratégias são descritas nos dois próximos capítulos deste texto.

3. MODELO DE SIMULAÇÃO DE CRESCIMENTO DO TRIGO CSM-CROPSIM: WHEAT

Entre as diversas espécies de diferentes culturas, o trigo destaca-se como um importante alimento para a humanidade. É cultivado em diversos lugares do mundo e por isso possui grande relevância econômica e social. Porém, sua produção pode ser afetada por diversos fatores como as condições do solo, do clima, do manejo, entre outras. Por isso são necessários meios de evitar que estes fatores hajam de forma negativa na produtividade da espécie. Os modelos de simulação de culturas combinam dados meteorológicos, dados do solo, fisiologia vegetal, etc e geram informações sobre possíveis impactos com a variação destes fatores no crescimento da cultura [6]. Entre estes modelos está o modelo CSM-Cropsim: Wheat, estudado neste trabalho e apresentado a seguir.

3.1 DESCRIÇÃO DO MODELO E SUAS CARACTERÍSTICAS

O modelo de simulação escolhido para ser um dos estudos de caso deste trabalho é um modelo de simulação de cultura utilizado no grupo de pesquisa Mosaico da Universidade de Passo Fundo e na Embrapa Trigo. Este modelo, chamado CSM-Cropsim: Wheat, foi desenvolvido por Leslie A. Hunt [2] na linguagem de programação Fortran e, atualmente, faz parte do sistema Decision Support System for Agrotechnology Transfer (DSSAT[1]), composto por diversos outros modelos de simulação de culturas. O CSM-Cropsim: Wheat simula o crescimento e desenvolvimento da cultura do trigo. Devido aos fatores já destacados neste texto, o CSM-Cropsim: Wheat necessita que sua execução quando aplicada a cada ponto (grid) deve ser organizada a fim de obter uma melhora em seu rendimento computacional em situações onde há uma grande área a ser coberta pelo modelo e/ou a série temporal a ser simulada é muito grande.

No Brasil, o CSM-Cropsim: Wheat tem sido testado, calibrado e validado por diversos pesquisadores, sendo utilizado para simular o processo do desenvolvimento de cultivares de trigo, além de contribuir para a tomada de decisão de pesquisadores e produtores da área agrícola [5].

Segundo Pavan [5], o CSM-Cropsim: Wheat é composto por um módulo planta de trigo que se conecta com o módulo de clima e solo, pertencentes à suite do DSSAT, os quais computam a energia e a água disponível para o crescimento da planta de trigo, ao passo que o módulo planta de trigo simula os eventos fenológicos, expansão foliar, acúmulo de carboidratos e a partição entre a parte aérea e raízes.

A funcionalidade deste modelo integra uma série de entrada de dados que compreendem dados meteorológicos, dados de solo, de fisiologia vegetal, de genótipo da planta, conforme representado na Figura 1. Eles estão parametrizados em arquivos de formato texto, sendo os resultados do modelo, com as informações de pós processamento, também armazenados em arquivos texto.

[1] http://dssat.net/

Figura 1. Estrutura da suíte do Sistema de Apoio à Decisão para Transferência de Agrotecnologia (DSSAT) [5].

Para um melhor gerenciamento destes arquivos, a aplicação realizada no trabalho de Lazzaretti [6] integrou o modelo CropSim com um banco de dados. Nele, os dados são organizados de forma que cada diretório contém arquivos textos com os dados do tratamento, num determinado período e, também, o executável do modelo. Após o modelo ser executado, os arquivos de saída são armazenados no mesmo diretório correspondente. Conforme ainda Lazzareti [6], os arquivos gerados pela sua aplicação são os seguintes:

- `.WTH`: arquivo com os dados meteorológicos. Os principais dados necessárias nessa etapa são: temperaturas do ar (máxima e mínima), precipitação, radiação solar e concentração de CO2 na atmosfera;
- `.SOIL`: arquivo com os dados de solo por camadas;
- `.WHX`: arquivo que gerencia a execução, no qual são descritos os tratamentos, a identificação dos arquivos de solo e clima, detalhes da semeadura, condições iniciais e manejo;
- `.CUL`: arquivo que contém os coeficientes da cultivar como, por exemplo: sensibilidade ao fotoperíodo, características fenológicas, taxa de fotossíntese à luz saturada, área foliar específica, peso máximo de sementes;
- `.ECO`: arquivo que contém os coeficientes de resposta da planta em relação ao ambiente (ecotipo). Possui alguns atributos genéticos que permitem diferenciar cultivares de hábito de crescimento determinado e indeterminado;
- `.CFG`: arquivo com as configurações para a execução do CSM-Cropsim: Wheat, tais como diretórios, módulos, arquivos e programas;

- `.SOM`: abreviatura de *Soil Organic Matter*. Arquivo que contém os parâmetros necessários para a simulação da matéria orgânica no solo.

- `.SPE`: arquivo que caracteriza a espécie. Possui coeficientes que caracterizam a composição básica dos tecidos e alguns processos da planta como: fotossíntese, respiração, assimilação de nitrogênio, partição de fotoassimilados, senescência, fenologia e crescimento.

- `.OUT`, `.LOG` e `.WHM`. arquivos de saída gerados após a simulação.

Na Figura 2 é demonstrado a estrutura de diretórios montada para a execução do modelo. O diretório "Localidade"contém os diretórios com os tratamentos, estes, identificados por um código inserido pelo banco de dados AgroDB desenvolvido por Lazzareti [6], e cada tratamento contém o conjunto de arquivos descritos anteriormente.

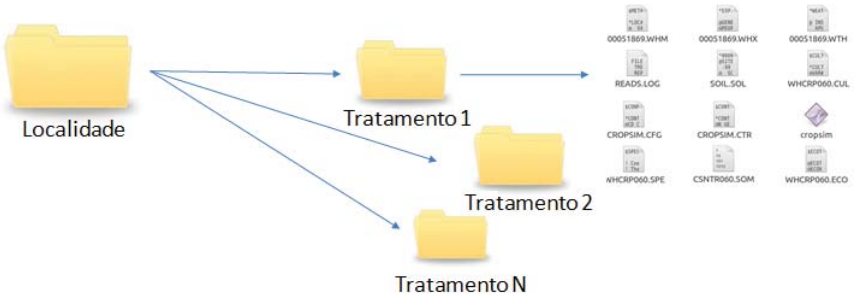

Figura 2. Estrutura dos diretórios montada para a execução do modelo CSM-Cropsim: Wheat.

Conforme a Figura 2, uma localidade está relacionada a um experimento. Dentro do diretório estão vários tratamentos, e cada tratamento contém conjuntos de dados de solo, épocas de semeadura e um conjunto de dados meteorológicos. Estes dados obtidos do banco de dados AgroDB [6], são dados que abrangem dez épocas de semeadura e trinta anos de dados meteorológicos observados do estado do Paraná - Brasil entre os anos de 1980 e 2009.

Por exemplo, nos dados utilizados neste trabalho, são processados 30 anos, com 5 épocas de semeadura e 16 ensembles de dados meteorológicos, totalizando 2400 tratamentos que equivalem à 2400 execuções do modelo.

3.2 EXECUÇÃO DO CROPSIM

O modelo de simulação CropSim é executado via linha de comando no Linux (`./cropsim 00051962.WHX 1 1`), onde `./cropsim` é o comando para a execução do arquivo binário do modelo de simulação,`00051962.WHX` é o arquivo de configuração responsável por comandar o processo de execução do modelo, ou seja, utilizar os dados de todos os outros arquivos de entrada, e os parâmetros `"1 1"` são valores que controlam qual tratamento do modelo deve ser executado.

Com esta linha de comando, que é correspondente à apenas uma execução de um tratamento, o tempo de execução sequencial é de aproximadamente 0.070 segundos. Porém, para cada

localidade, que contém vários tratamentos, são necessárias inúmeras execuções deste modelo, o que acaba acarretando em mais tempo de simulação para que este possa abranger todas os experimentos da mesma localidade. Geralmente, um mesmo experimento contém inúmeros tratamentos. Portanto, para executá-lo, são necessários vários arquivos de entrada e, para sua execução, várias chamadas desta linha de comando.

3.3 CONCLUSÕES DO CAPÍTULO

O CropSim apesar de ser um modelo de simulação desenvolvido em 1995, ainda é muito utilizado pelos pesquisadores da área agrícola para auxílio na tomada de decisão. Um exemplo disto pode ser citado a parceria entre a Embrapa Trigo e da Universidade de Passo Fundo, em que os dados são processados pelo modelo no servidor da universidade e, posteriormente as informações são utilizadas pelos pesquisadores da Embrapa e pelos agricultores. Esta ampla utilização e a necessidade de ampliar sua área e período de cobertura é que justifica a escolha de se trabalhar na definição de uma estratégia para a sua execução paralela.

4. MODELO DE CORREÇÃO ESTATÍSTICA DA PREVISÃO DO TEMPO GERADA PELO ETA 15KM

As previsões de modelos dinâmicos, como os de previsão do tempo, apresentam erros que podem ter várias origens. Estes erros podem ser gerados devido a simplificações ou a dificuldades nas representações numéricas dos processos atmosféricos, nas definições da condição inicial e das condições de contorno, e, até mesmo, devido à dinâmica do fenômeno que pode apresentar baixa previsibilidade. O conhecimento dos erros sistemáticos pode permitir a redução dos erros para uso posterior em modelos nas áreas agrícolas e energéticas, e, desta forma, obter uma previsão mais exata. A redução do erro da previsão do tempo em algumas localidades que possuem medidas observacionais pode ser obtida a partir da correção estatística. Este é o caso da correção estatística realizada sobre a previsão do tempo gerada pelo modelo Eta 15km do CPTEC/INPE.

4.1 MODELO REGIONAL DE PREVISÃO DO TEMPO ETA

O modelo Eta foi desenvolvido na Universidade de Belgrado (Sérvia) [40] e tornado operacional no National Centers for Environmental Prediction (NCEP) [41]. Trata-se de um modelo baseado na utilização de coordenadas verticais, o qual permanece aproximadamente horizontal em áreas montanhosas, tornando-o adequado para estudos de regiões de topografia íngreme, como, por exemplo, a cordilheira dos Andes [42]. Tal fato justifica sua nomenclatura, visto que o termo Eta é derivado da letra grega η (eta), a qual significa coordenada vertical. Sua estrutura é, então, formada por grades horizontais (Grade E de Arakawa) e coordenadas verticais [43] (Figura 3).

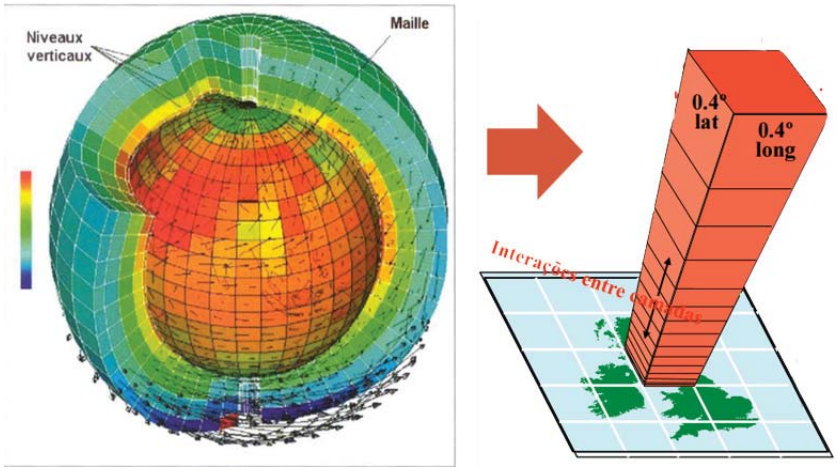

Figura 3. Modelo de previsão regional baseado em grades horizontais e coordenadas verticais(Adaptações de [44]).

O modelo Eta utilizado no Brasil trata-se de um modelo regional, o qual é utilizado no CPTEC/INPE em forma operacional, abrangendo a América do Sul, desde 1997 para previsões meteorológicas e, desde 2002, para previsões climáticas [1]. As principais fontes de derivação dos

dados observados do modelo Eta são: estações meteorológicas, estações de superfície, aeroportos, navios, aviões, satélites e radares (Figura 4).

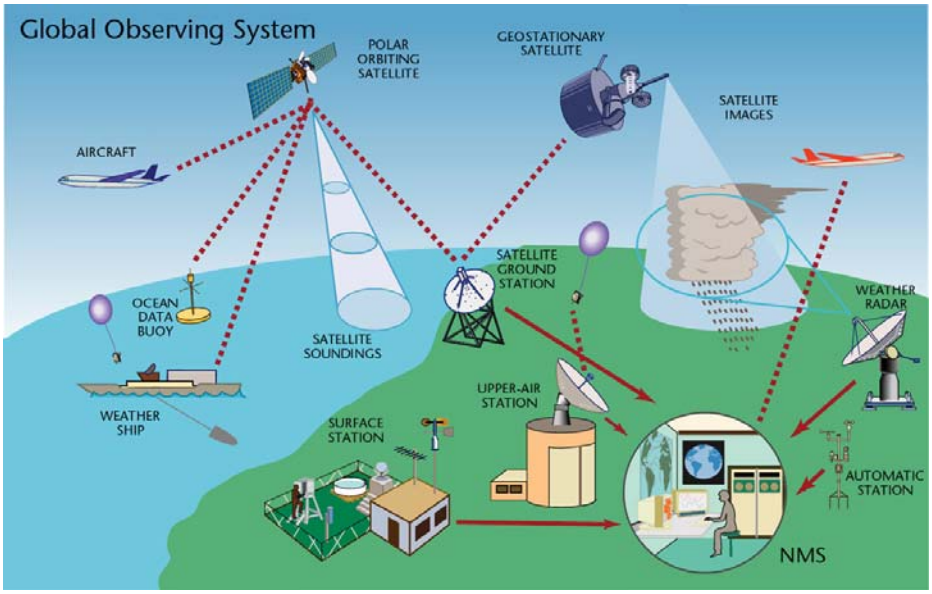

Figura 4. Sistema de fontes de Observação Global [45].

As variáveis prognósticas do modelo Eta são: temperatura do ar, componentes zonal e meridional do vento, umidade específica, pressão à superfície e energia cinética turbulenta. Assim, o modelo propõem-se a prever fenômenos atmosféricos, de forma mais detalhada, quando associado a estas variáveis, como: tempestades, nevoeiros, etc. Como tais variáveis não possuem uma linearidade, podendo variar com uma frequência maior em grandes espaços de tempo devido à mudanças climáticas envolvidas, tal previsão é melhor definida quando ocorrida em um curto espaço de tempo. O modelo regional Eta estendem-se em até 168 horas, sendo fornecida 2 vezes ao dia (horas 00:00 e 12:00 UTC), para o modelo regular de grade de 40km e de 15km (modelo abordado nesta pesquisa). Possui, também, grades em modelos de 5km e 1km.

No estudo realizado em [46], por meio de comparações dos resultados das previsões do modelo Eta com dados meteorológicos provindos de estações meteorológicas convencionais e automáticas dispostas em alguns locais avaliados, foi constatado que a modelagem apresentou deficiências nos elementos da circulação da superfície e das possíveis interações entre estes elementos, ou seja, os resultados da previsão do modelo não eram equivalentes aos dados observados. O erro foi obtido utilizando métodos estatísticos da diferença entre o valor médio previsto e o valor médio observado. Com isso é possível identificar, por exemplo, se a previsão sistematicamente superestima as observações. Assim, conforme descrito pelos autores pôde-se concluir que o modelo Eta pode não prever adequadamente as previsões [46].

Outro ponto importante à ser avaliado nas previsões deste modelo é sobre os valores das variáveis dentro de um mesmo ponto de grade que podem variar de uma localidade para outra repercutindo também na média dos valores previstos, como é o caso exemplificado em [46] que cita a localidade de São Paulo onde as previsões geradas pelo Eta são influenciadas pelas características locais.

4.2 MODEL OUTPUT CALIBRATION (MOC)

O modelo Eta calcula a previsão com base em dados médios, como altitude média e vegetação predominante na resolução de grade horizontal de aproximadamente 15 km x 15 km de cada camada. Entretanto, quando regiões com características de topografia e vegetação distintas estão em um mesmo bloco de 15 x 15 km, ocasiona em um erro de previsão [1]. Com isso, é utilizado um modelo de correção Model Output Calibration (MOC), que corrige para um ponto específico estatisticamente o erro da previsão numérica a partir do dados históricos observados nos dias anteriores [1].

Além do MOC, há outros modelos de correção como o Model Output Statistics (MOS) [47] e os baseados em redes neurais artificiais (RNA) [1]. Geralmente, para se obter correções, é necessário, uma série longa de dados históricos e um modelo com suas características mantidas constantes, isto é, congeladas para permitir o treinamento de padrões [1]. Estes modelos de correção geralmente são escritos em Fortran pois os modelos no qual se pretende realizar a correção também o são, caso do Eta. Porém, o MOC trabalhado nesta pesquisa contém módulos reescritos em R, possibilitando uma otimização de código original e fazendo uso das funcionalidades estatísticas e de visualização do R. O MOC baseia-se na regressão linear multivariada desenvolvida por Mao et al [3]. Entre as características do MOC estão:

- é um esquema simples e adequado para variáveis bem comportadas como a temperatura do ar;
- estima o erro da previsão da variável;
- possui tamanho curto da série de treinamento, permitindo que o esquema facilmente se ajuste às mudanças e às melhorias nas previsões do modelo;
- permite que um aumento de resolução ou de mudança de parâmetros da física do modelo possa rapidamente ser submetidas ao refinamento com a construção de um período de 28 dias de previsão com esta nova configuração.

O MOC foi desenvolvido em Fortran com alguns *scripts* com funções em R, e consiste em formular equações de regressão linear multivariada com base nos dados das séries de poucas semanas das previsões e das observações, estimando e ajustando, assim, os erros da previsão da variável a ser corrigida, neste caso a temperatura do ar a 2 metros da superfície e a umidade relativa [1].

As variáveis candidatas à preditoras, a serem usadas pelo modelo são selecionadas a partir da correlação entre o erro da previsão e as variáveis previstas pelo modelo, sempre calculadas por meio de equação de regressão linear detalhadas em [1], onde 72 variáveis são utilizadas como candidatas a preditoras extraídas de dados de várias cidades observados em aeroportos e codificadas em código *Meteorological Aerodrome Report* (METAR), que são distribuídas de hora em hora.

Ainda conforme [1], após a seleção das variáveis preditoras, são construídas 24 equações para cada horário do dia, para cada cidade. O refinamento é aplicado para o ciclo de 24 a 48 horas da previsão.

As comparações entre a série observada e a série após a aplicação do método de correção do MOC mostraram resultados satisfatórios quanto à redução do erros na previsão realizada somente pelo Eta. Com isso, os autores concluíram que o uso é viável para que as previsões cheguem mais próximo possível da realidade observada [1].

4.3 EXECUÇÃO DO MOC

A execução sequencial do modelo é feita utilizando linha de comando no Linux e o tempo sequencial deste modelo é de aproximadamente 720 segundos, na máquina descrita na secção 6.2.1, para corrigir dados da previsão do tempo para 36 localidades no Brasil, corrigindo a temperatura e a umidade relativa para as horas 24 a 47 da previsão (dia seguinte) do Eta 15 km. Esta correção está sendo utilizada atualmente para uma aplicação na área de energia coordenada pelo Operador Nacional do Sistema Elétrico (ONS) em uma parceria CPTEC-INPE/PPGCA-UPF. Na Figura 5 são descritas as etapas da correção estatística realizadas no modelo Eta pelo MOC, especificamente para este projeto com o ONS. O sistema atual foi implementado utilizando as linguagens R (Figura 5 - etapas 1 e 3) e Fortran (Figura 5 - etapas 2 e 3), em conjunto com scripts Linux para o gerenciamento de alguns processos (Figura 5 - etapas 2, 3 e 4).

4.4 CONCLUSÕES DO CAPÍTULO

O MOC tem sua importância devido a sua utilização na correção dos erros da previsão do tempo gerada pelo modelo Eta. Um exemplo da validade desta correção pode ser constatado nas Figuras 6 e 7, onde são apresentados os dados observados, os previstos pelo Eta e a previsão corrigida pelo MOC para as cidades do Sul do Brasil, integrantes do projeto com o ONS, referente ao período de julho a setembro de 2015. Nestas figuras pode-se constatar a validade da correção e a melhoria na qualidade dos dados corrigidos em relação aos previstos inicialmente pelo modelo Eta.

Porém, este processo deve ser estendido para mais localidades (podendo chegar a quase 500 localidades), corrigindo mais variáveis além das duas atuais (temperatura e umidade relativa) e para todas as 168 horas da previsão do Eta e não apenas para o período entre as horas 24 e 47. Devido a esta demanda futura a paralelização da execução deste modelo de correção se fez extremamente necessária.

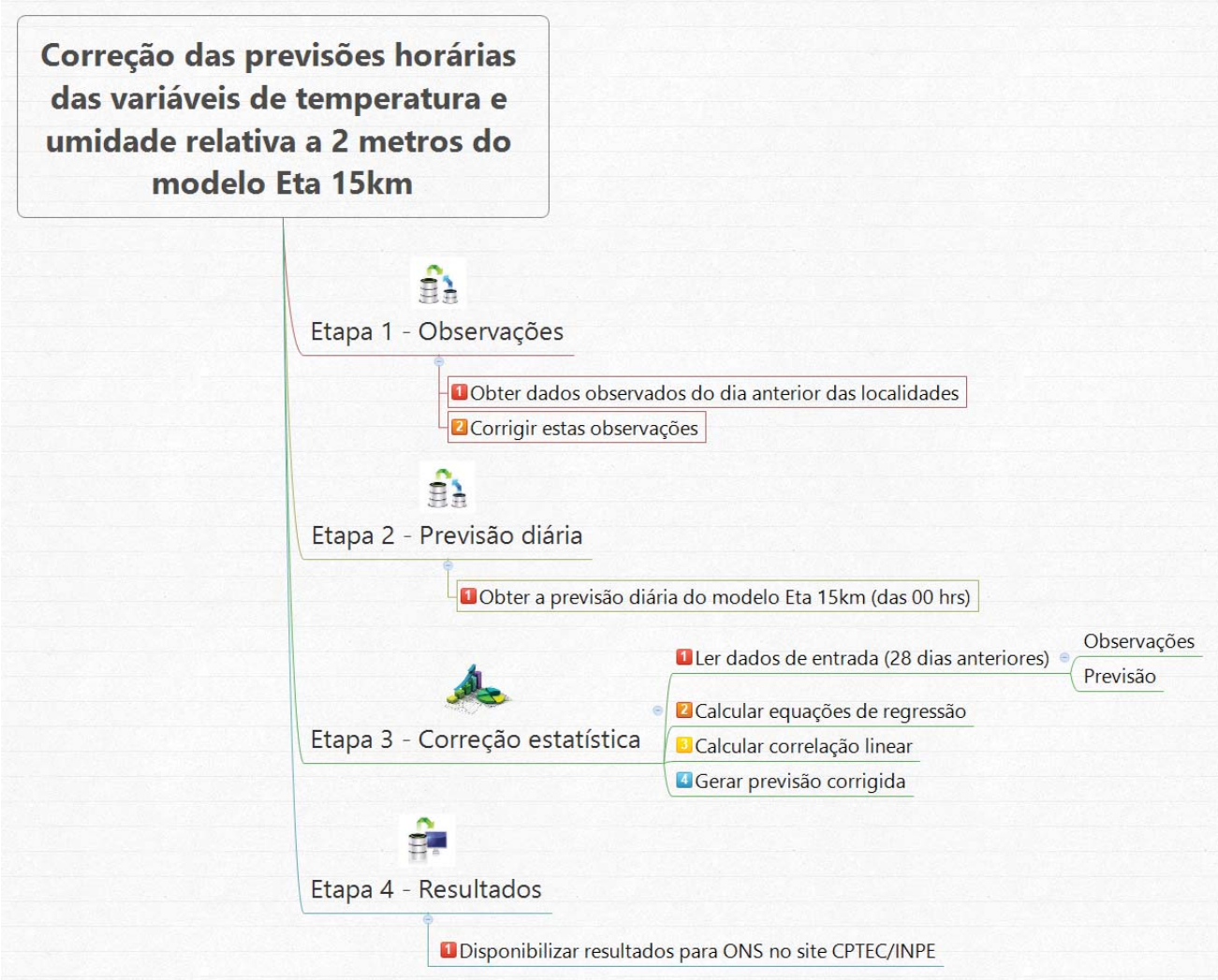

Figura 5. Etapas para a correção estatística da previsão do modelo Eta.

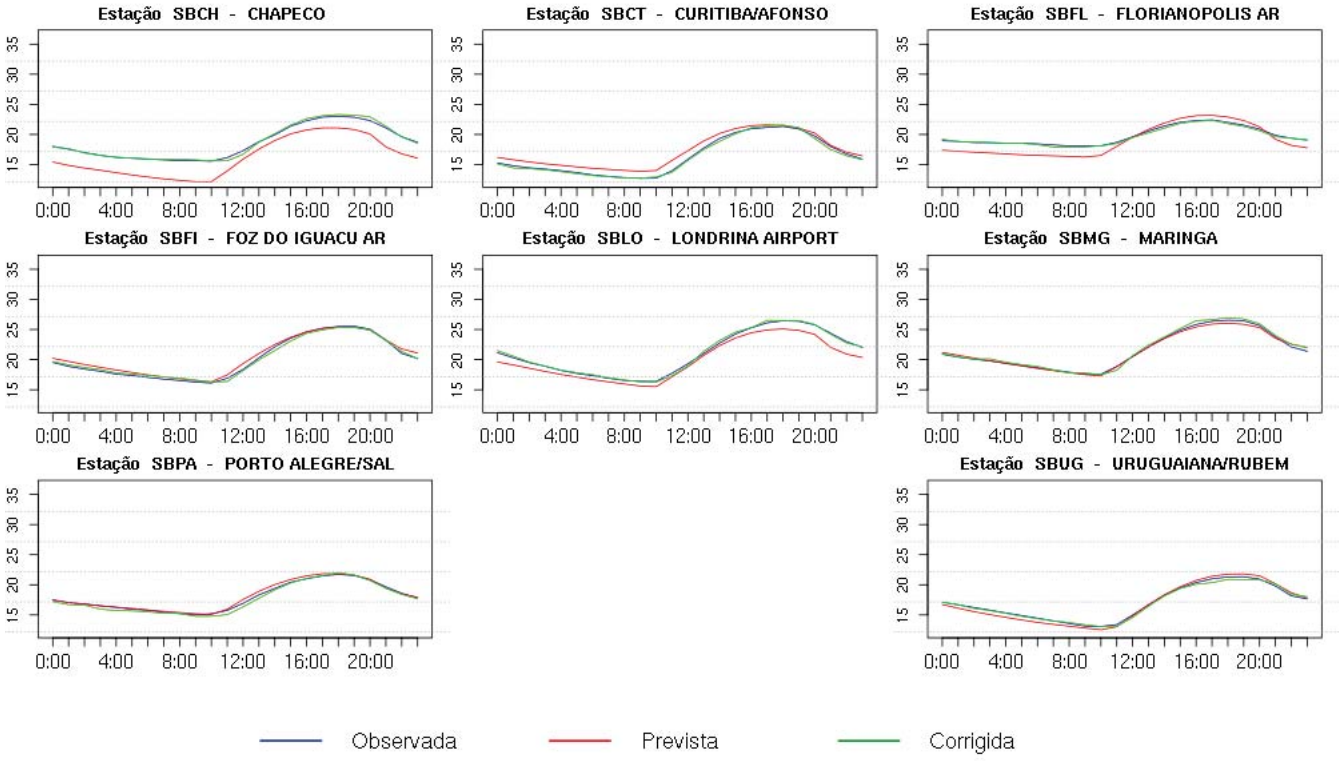

Figura 6. Temperatura do ar observada, prevista e corrigida para o Subsistema Sul.

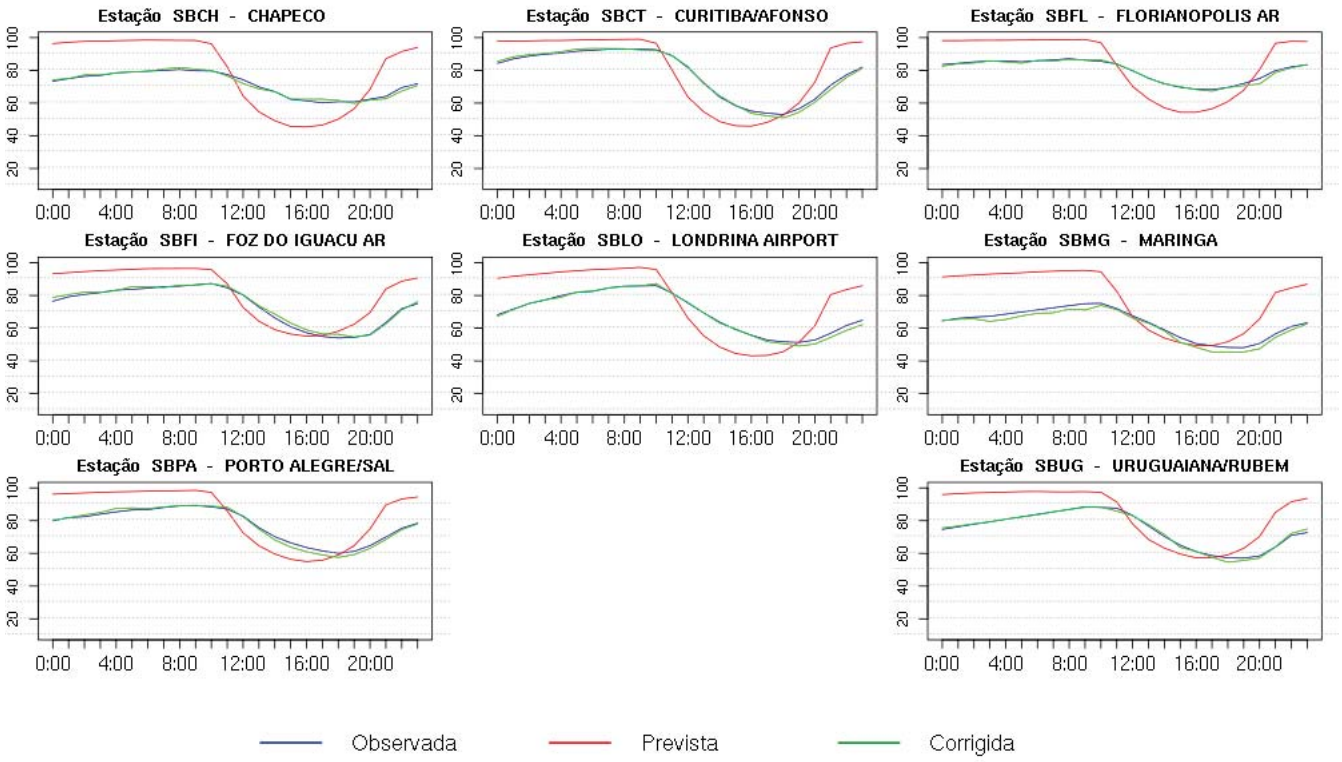

Figura 7. Umidade relativa observada, prevista e corrigida para o Subsistema Sul.

5. ESTRATÉGIA DE PARALELIZAÇÃO

Neste capítulo é apresentada a estratégia de paralelização aplicada para o modelo de simulação do crescimento do trigo CSM-Cropsim: Wheat e para o modelo de correção da previsão do tempo MOC.

5.1 CSM-CROPSIM: WHEAT

Com o objetivo de executar o modelo de simulação CSM-Cropsim: Wheat de forma eficiente e rápida, abrangendo mais localidades e utilizando uma série histórica maior, buscou-se definir uma maneira viável de paralelizar sua execução visando alcançar este objetivo.

A paralelização deste modelo se deu com base no algoritmo paralelo mestre escravo. Esta abordagem foi a que melhor se adaptou a estrutura de funcionamento do modelo, visto que cada uma de suas execuções (rodada do modelo) são independentes. Este método foi escolhido pois permite realizar a programação paralela da execução do modelo sem alterar seu código fonte, ou seja, conforme os processos vão terminando suas tarefas de processamento dos dados, o mestre pode enviar outras até que não haja mais nenhum dado para ser processado. Optou-se por não paralelizar o modelo porque o cálculo para uma execução de um tratamento é de apenas $0,055$ segundos. O maior tempo gasto com a execução do modelo deve-se ao processo de leitura/escrita de arquivos. Anteriormente à este trabalho foram realizadas tentativas pelo grupo de pesquisa da Universidade de Passo Fundo em realizar a paralelização implícita do modelo, porém, após a análise do desempenho obtido e devido a estrutura de funcionamento do modelo, chegou-se à conclusão que esta paralelização não traria benefícios significativos.

Como exemplo de uma paralelização de modelo baseada no algoritmo mestre/escravo, onde houve melhora no desempenho, pode ser citado o trabalho de Hadka [48] onde foi desenvolvida a implementação paralela do algoritmo Borg Multiobjective Evolutionary Algorithm (MOEA) voltado para resolver problemas em fontes de águas. Por meio de testes com até 16.384 processadores obteve-se resultados satisfatórios. Outro trabalho com este tipo de abordagem é o realizado por Li [49], onde é apresentado o desenvolvimento de um algoritmo paralelo dinâmico para a realização de simulações de modelos hidrológicos, que utilizam a metodologia mestre/escravo e a comunicação com o padrão MPI. Os resultados desta aplicação revelam que o algoritmo é eficiente no envio das tarefas de simulação entre os processos e que há um aumento da velocidade e eficiência em função da largura da bacia.

A estratégia de paralelização da execução do modelo CSM-Cropsim: Wheat funciona de forma que um conjunto de comandos de execução de simulações sejam enviadas para cada processador. Cada linha de comando contém a execução de um tratamento, e um tratamento corresponde à um conjunto de dados referente à uma localidade. Por exemplo, se são necessárias 9600 execuções do modelo (isso corresponde à 4 localidades com 2400 tratamentos cada uma)

serão enviadas 50 linhas de comandos de execução para cada processo, cada linha de comando é uma execução do total das 9600. Este envio é feito pelo processo mestre. Logo após, o processo escravo começa as execuções. Quando este chega ao final das 50 linhas de comando recebidas, envia um aviso ao mestre informando que finalizou suas execuções. Se ainda houver linhas de comando a serem executadas, o mestre as envia aos seus escravos até que não as tenha mais (Figura 8). Quando todo o trabalho estiver chegado ao final, o mestre envia um sinal de fim de trabalho aos escravos.

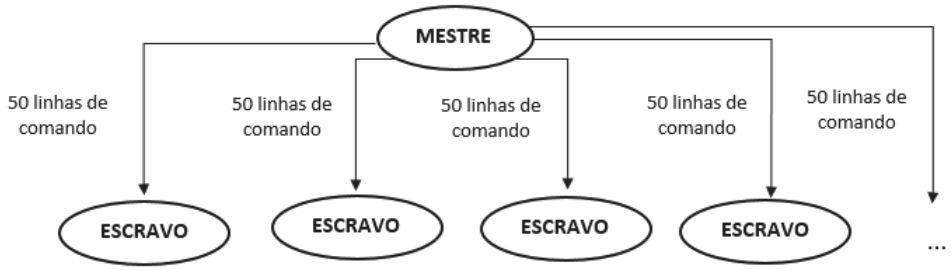

Figura 8. Modelo paralelo de execução do modelo CSM-Cropsim: Wheat.

Esta troca de mensagens é feita pelo padrão MPI, que é uma interface de programação para comunicação de dados entre processos paralelos. Esta ferramenta oferece infra-estrutura para modalidades de computação paralela quando existe a necessidade de passar informações entre os vários processadores ou nodos de um cluster com memória distribuída [50]. Também, conforme descrito em [50], a interface do MPI foi estendida com funções para gerenciamento dinâmico de processos, entrada/saída paralela e acesso a memória remota, além de ser mais adaptada a arquiteturas com memória híbrida (compartilhada entre os núcleos e distribuída entre os nós).

Outra questão que implica no desempenho da execução paralela é quanto ao aumento do número de processadores que, em um dado momento, pode tender à uma diminuição de tempo de execução. Um exemplo desta situação é o trabalho descrito por Zhang [39], onde os autores comentam sobre a crescente disponibilidade de dados e o uso cada vez maior de modelos de bacias hidrográficas distribuídos por grandes áreas com alta resolução espacial e temporal. O autor descreve que neste trabalho, foi desenvolvido um software paralelo para melhorar a eficiência de calibração dos modelos. Resultados de testes realizados em um cluster de computadores com sistema operacional Linux mostraram que o software desenvolvido pode alcançar um melhor desempenho dependendo da complexidade do modelo e que, aumentando a quantidade de processadores além de certo limite, não necessariamente melhorou a eficiência, pois a concorrência de intensificação do recurso resultou em um gargalo de *I/O*. Portanto, existe a possibilidade de melhora no desempenho até certa quantidade de processos. Isto vai depender das características de cada modelo e da tecnologia usada na implementação.

5.2 MOC

Com o objetivo de estender a correção feita pelo MOC para mais localidades é necessário utilizar uma abordagem paralela para que haja esta possibilidade e, também, para aumentar o número de variáveis à serem corrigidas. Pois, sem o uso da mesma, resultaria em um alto custo computacional para execuções sequenciais dos dados pelo modelo.

A abordagem de paralelização da execução do MOC foi realizada de forma que cada um dos conjuntos de dados para cada localidade a ser utilizado pelo modelo (Figura 5 - etapa 3 do processo) foi executado por um núcleo da máquina. A metodologia do algoritmo é mestre/escravo. A (Figura 9) ilustra o funcionamento do algoritmo.

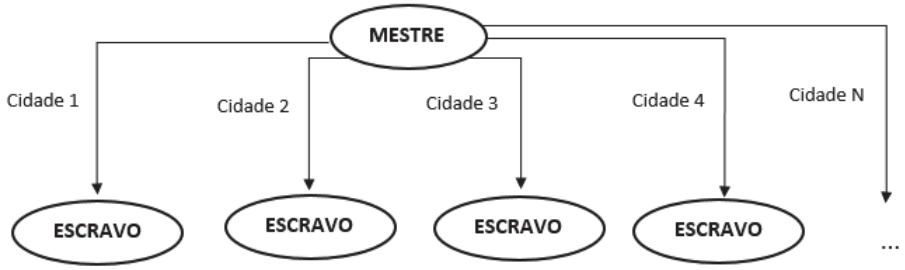

Figura 9. Funcionamento do algoritmo paralelo de execução do modelo MOC.

O modelo se estende para um número indeterminado de escravos mas dependendo da quantidade de comunicações o mestre pode ser um gargalo de *I/O*. Contudo, para o MOC, como é observado na (Figura 9), o mestre envia para cada escravo, por meio de um laço de repetição, apenas um valor correspondente à cidade que ele deverá processar. Conforme o processo escravo chega ao fim da computação ele envia um sinal ao mestre, e se ainda houver cidades o mestre envia outra, e assim sucessivamente até o final da quantidade de cidades. Ao final da execução, caso não haja mais localidades e os escravos já finalizaram suas execuções, estes enviam um sinal de fim para o mestre e o mestre encerra o trabalho dos escravos com um sinal de final de trabalho.

5.3 CONCLUSÕES DO CAPÍTULO

A programação da paralelização da execução do modelo CSM-Cropsim: Wheat não necessitou de conhecimentos aprofundados da linguagem Fortran na qual ele foi desenvolvido, facilitando, assim, este tipo de paralelização. A dificuldade encontrada esteve na grande quantidade de arquivos de entrada, resultado em muitas comunicações para leitura e escrita de arquivos.

O processo paralelo do MOC, também desenvolvido na linguagem Fortran e com algumas rotinas utilizando a linguagem R, teve sua implementação utilizando o algoritmo mestre/escravo. Este modelo acessou os dados de cada cidade em um único comando de execução por meio de um laço de repetição, não necessitando de uma mensagem entre mestre escravo com várias linhas de comando. A descrição dos *scripts* de execução e valores que comprovam o desempenho são apresentados no capítulo 6 que apresenta os testes realizados e a análise dos resultados obtidos neste trabalho.

6. ANÁLISES DOS TESTES

Após o estudo e a compreensão do funcionamento dos modelos de simulação e da aplicação de uma abordagem paralela para a execução dos mesmos, foram realizados testes para avaliar e analisar o desempenho obtido com esta execução paralela, utilizando a biblioteca de comunicação MPI.

Inicialmente são demonstrados os resultados obtidos na execução do primeiro estudo de caso, o CSM-Cropsim: Wheat. Este modelo de simulação é executado com processos considerados *I/O bound*, onde a maior parte do tempo de processamento é consumido em leitura e escrita de arquivos. Em seguida são apresentados os resultados da execução paralela do segundo estudo de caso, o modelo de correção MOC, modelo que também faz uso de leitura e escrita de arquivos, porém o processamento dos seus dados é influenciado pelo desempenho do processador, sendo neste caso considerado CPU *bound*, consumindo, assim, mais tempo na realização dos cálculos. O cálculo de *speedup* foi calculado dividindo o tempo sequencial pelo tempo de cada execução paralela, a eficiência foi calculada pela razão entre o *speedup* e o número de processos utilizados, em ambos os casos.

6.1 CSM-CROPSIM: WHEAT

Nesta seção é apresentado o ambiente computacional em que foram realizados os testes da paralelização da execução do modelo CSM-Cropsim: Wheat, como foi realizada a estruturação dos testes e, por fim, os resultados obtidos.

6.1.1 Ambiente de execução e simulação

A análise do desempenho da implementação paralela do modelo CSM-Cropsim: Wheat foi realizada por meio da execução com diferentes números de processadores de um sistema constituído por várias estações de trabalho, buscando obter o melhor desempenho computacional na divisão de trabalho entre os núcleos de processamento da máquina *Network of Workstations* (NOW) [51].

Utilizou-se um aglomerado homogêneo composto por 16 máquinas formado pelas máquinas do Laboratório Central de Informática (LCI) da Universidade de Passo Fundo, cada qual com a seguinte configuração: Processador Intel Core i7 2600, 3.4 GHz, memória RAM 8Gb, 4 núcleos físicos e sistema operacional Linux (Ubuntu 12.10) de 64-bits.

Os dados utilizados para a realização dos testes foram obtidos do banco de dados desenvolvido pelo trabalho descrito em Lazzaretti [6] e correspondem à quatro localidades do sul do Brasil. O tempo para a execução sequencial do modelo para estes dados foi de aproximadamente 43 minutos.

6.1.2 Estruturação dos testes

Todos os valores de tempos de execução apresentados resultam da média dos valores obtidos a partir de sete repetições para que possa haver uma maior confiabilidade dos resultados. Por este motivo, estas execuções foram realizadas no momento em que não havia uso da rede do laboratório pelos alunos, uma vez que os laboratórios estavam fechados para os alunos das 22h30min as 07h00min de segunda a sexta-feira e nos finais de semana das 11h30min do sábado até as 07h00min da manhã da segunda-feira.

Inicialmente optou-se por realizar os testes do CSM-Cropsim: Wheat com os arquivos estando compartilhados em um diretório público armazenado em um servidor do LCI. Porém, observou-se que os dados estavam constantemente sendo acessados pelos processos, ocasionando, assim, muitas comunicações devido a leitura de arquivos necessários para o processamento dos dados e pela escrita dos arquivos produzidos pela execução do modelo após processar os dados. Portanto, além dos testes centralizando dados em uma única máquina também foi realizada a distribuição dos mesmos entre as máquinas, e, assim, cada processo buscou os dados em seu disco local. Após o processamento, os arquivos de saída foram centralizados em uma única máquina para facilitar a visualização das informações dos dados processados. Ao fim da execução, um algoritmo realizou a centralização dos resultados novamente em uma única máquina.

Durante as primeiras execuções percebeu-se que estava ocorrendo um reaproveitamento dos dados da *cache* de disco. Para garantir que nenhuma informação ficasse armazenada na mesma, acarretando em uma falsa ideia de desempenho e evitando que o tempo da primeira execução fosse maior que o das conseguintes, foi realizado o envio do comando `cp arq1 pasta1` para copiar um arquivo para um diretório qualquer com o objetivo de substituir a informação da memória *cache* de disco.

Outro obstáculo encontrado e relacionado com a memória *cache*, é que a mesma estava armazenando informações da quantidade de processos utilizados, resultando, com isso, tempos mais baixos após a primeira execução. Este problema foi solucionado alternando o número de processos entre a bateria de comandos dos testes. Por exemplo, o arquivo continha a execução com um número de processos (visando substituir a informação sobre os processos) e uma cópia de arquivos (visando substituir a informação sobre os dados), então, assim, a execução era realizada com um número de processos diferente da execução anterior. Na Figura 10 é demonstrado um exemplo do arquivo *shell script* do Linux para a execução dos testes.

Na Figura 10, a linha 3, corresponde a uma linha de comando que envia 200 linhas de execução do modelo de simulação para oito processos, e, posteriormente, na linha 4, é feita a cópia de um arquivo chamado "arq1"para a "pasta1", sucessivamente. Na linha 6 outro comando para execução envia tarefas para 16 processos.

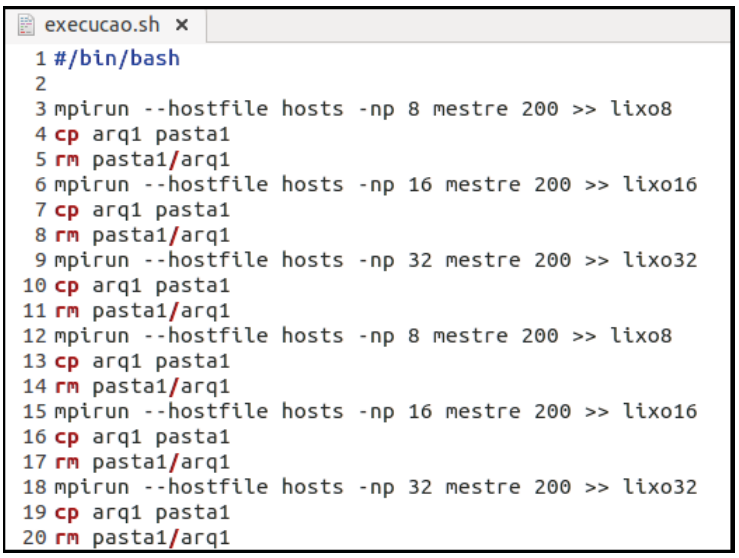

Figura 10. Script das linhas com comandos de execução do modelo CSM-Cropsim: Wheat.

6.1.3 Resultados

Na Tabela 1 são apresentados os valores em segundos do tempo necessário para a execução paralela do modelo CSM-Cropsim: Wheat em uma única máquina utilizando de 2 e 4 processos. A primeira coluna das duas tabelas a seguir, apresenta a quantidade máquinas utilizadas em cada execução, a segunda coluna a quantidade de processadores de cada máquina, a terceira coluna o número de processos que foram utilizados, a quarta coluna é a quantidade de linhas de execução (seção 5) que está sendo enviada para cada processo. A quinta coluna contém os valores em segundos do tempo gasto na execução do modelo com dados distribuídos, a sexta coluna mostra o tempo de execução em segundos dos dados centralizados em um único local em um servidor da rede. As duas colunas da direita são apresentadas para avaliar a diferença em relação à busca dos dados em um único local ou com os dados distribuídos entre as máquinas.

Tabela 1. Execução paralela do modelo CSM-Cropsim: Wheat

Máquinas	Processadores	Processos	Linhas	Dados Centralizados	Dados Distribuídos
1	4	2	50	5315,42	4326,14
1	4	2	100	5257,18	4469,89
1	4	2	200	5209,04	4536,01
1	4	2	400	4931,08	4931,77
1	4	2	800	4841,73	5015,64
1	4	2	1000	4919,14	5990,55
1	4	4	50	3403,03	2938,72
1	4	4	100	3428,60	3453,70
1	4	4	200	3431,12	3281,42
1	4	4	400	3454,07	2933,44
1	4	4	800	3417,33	3653,56
1	4	4	1000	3488,59	3521,12

Como pode ser observado na Tabela 1, utilizando os núcleos de processamento de apenas uma máquina e dobrando o número de núcleos, mesmo comparando os dois maiores tempos gastos que são $5315,42$ segundos e $3488,59$ segundos (dados centralizados), sem levar em consideração a quantidade de linhas de comando enviadas para cada processo, é possível obter uma melhora no desempenho de aproximadamente 35%. Porém, estes tempos ainda são maiores que o sequencial aproximadamente $2326,5$ segundos, pois está despendendo tempo de leitura e escrita de dados.

A Tabela 2 demonstra os valores em segundos do tempo necessário para a execução paralela do modelo CSM-Cropsim: Wheat utilizando de 2 até 16 máquinas, fazendo uso de 8 até 16 processos.

Tabela 2. Execução paralela do modelo CSM-Cropsim: Wheat

Máquinas	Processadores	Processos	Linhas	Dados Centralizados	Dados Distribuídos
2	4	8	50	1986,88	1632,06
2	4	8	100	2161,28	1847,75
2	4	8	200	2146,75	1956,46
2	4	8	400	1607,31	2008,12
2	4	8	800	1806,62	2165,43
2	4	8	1000	2019,17	2336,39
4	4	16	50	819,32	437,06
4	4	16	100	876,31	646,64
4	4	16	200	874,06	679,85
4	4	16	400	797,79	787,51
4	4	16	800	836,15	500,56
4	4	16	1000	885,03	515,88
8	4	32	50	601,33	251,25
8	4	32	100	403,02	289,53
8	4	32	200	367,53	337,33
8	4	32	400	366,63	582,30
8	4	32	800	876,22	882,02
8	4	32	1000	935,15	903,30
16	4	64	50	213,66	187,13
16	4	64	100	214,60	195,00
16	4	64	200	362,68	359,36
16	4	64	400	724,48	729,03
16	4	64	800	955,85	931,15
16	4	64	1000	1179,43	1010,84

Como pode ser observado na Tabela 2, há uma variação nos valores do tempo de execução paralela influenciados por quantidade de processos, pelo número de linhas enviados para cada processo, pelos dados de entrada e saída distribuídos entre as máquinas e pelos dados de entrada e saída armazenados em um único local.

Por exemplo, para quando foram utilizados 8, 16, 32 e 64 processos, a execução que apresentou melhor resultado foi a que enviou 50 linhas de execução para cada processo (Figura 11) com dados distribuídos. Possivelmente porque o desempenho foi influenciado pelo tamanho da

mensagem enviada para cada processo pois quanto maior for a mensagem maior é o gerenciamento do *buffer* feito pelo MPI, e o tempo, tende a crescer com o aumento do tamanho da mensagem.

A seguir, na Figura 11, é apresentado um gráfico que faz a comparação entre os valores de desempenho obtidos, com o armazenamento dos dados centralizados, e os dados armazenados distribuídos, mostrando o tempo de processamento enviando para cada processo quantidades diferentes de linhas de comando de execução.

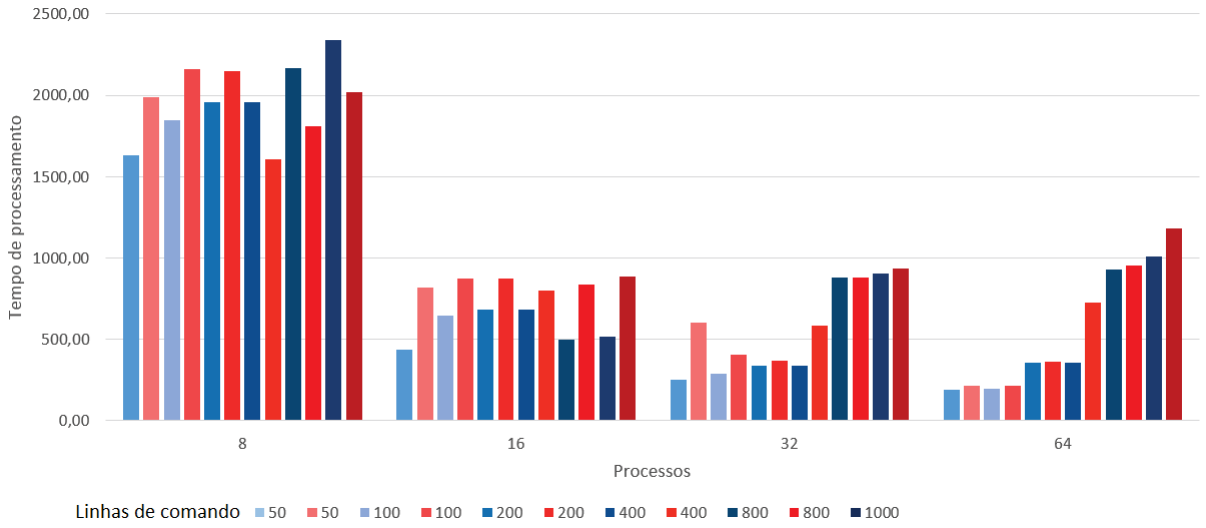

Figura 11. Comparação do desempenho obtido com os dados distribuídos (azul) e com os dados armazenados em um único local (vermelho).

Comparando os resultados das execuções paralelas de quando os dados foram centralizados em relação aos resultados das execuções paralelas de quando os dados foram distribuídos, nota-se que a variação não apresentou grandes diferenças.

Analisando a diferença, pode-se observar que quando os dados foram centralizados houve uma melhora no desempenho, mas não na mesma proporção quando os dados foram distribuídos. Por exemplo, nos casos em que foram utilizados 8, 16 e 32 processos (Figura 11) obteve-se melhor resultado enviando 400 linhas de execuções para cada processo. Isto, possivelmente devido à menos comunicação entre os processos, e também porque os mesmos buscaram em menor quantidade de vezes os dados no servidor onde estavam armazenados na rede. No entanto, quando foram utilizados 64 processos, houve mais tempo despendido por causa das comunicações entre mestre e escravos, e houve uma perda de desempenho.

Após analisar qual o melhor caso levando em consideração o desempenho adquirido observando o número das linhas, e onde os dados foram armazenados foi construída a Tabela 3, somente apresentando os valores para quando os dados foram distribuídos. Nesta tabela foram reunidos apenas os valores referentes às 50 linhas enviadas para cada processo pois de 6 diferentes quantidades de processos em 4 casos obteve-se melhor desempenho.

Levando em consideração o melhor resultado para o número de linhas de execução, analisando a Tabela 3, obteve-se melhor desempenho quando foram utilizados 64 processos para quando os dados foram distribuídos. Obteve-se um *speedup* de $13,85$ (Figura 12) mostrando em quantas

Tabela 3. CSM-Cropsim: Wheat: 50 linhas de execução e dados distribuídos

Processos	Tempo médio (seg)	Speedup	Eficiência (%)	Custo
2	4326,14	0,60	29,96%	8652,28
4	2938,07	0,88	22,06%	11752,28
8	1632,72	1,59	19,84%	13061,76
16	500,56	5,18	32,36%	8008,96
32	251,25	10,32	32,24%	8040,00
64	187,13	13,85	21,64%	11976,32

vezes foi acelerada a versão sequencial que foi de aproximadamente, 4047,5 segundos. Pode-se observar que o valor do *speedup* é menor que o número de processos, portanto ele é abaixo do esperado (linha azul em relação à laranja na Figura 12).

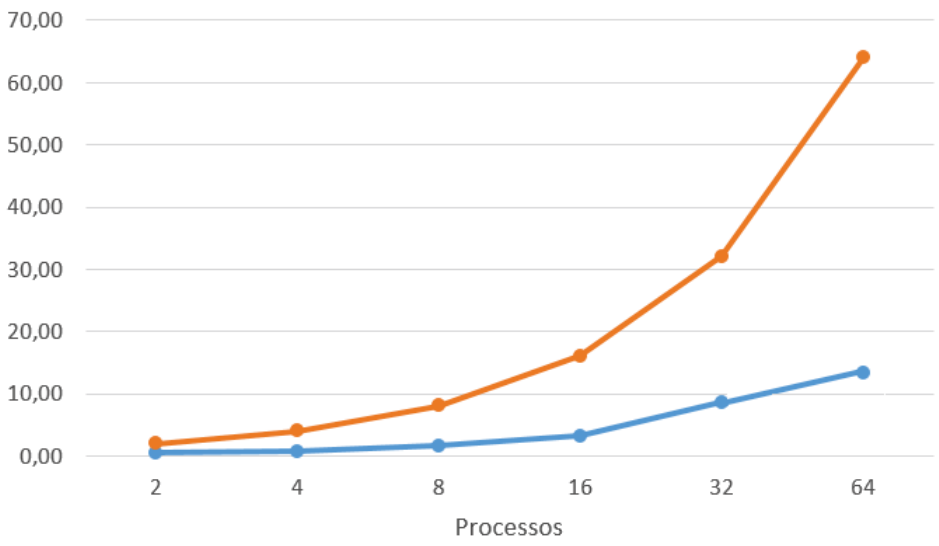

Figura 12. Gráfico de *speedup* dos dados distribuídos do CSM-Cropsim: Wheat. (Linha laranja *speedup* teórico, linha azul *speedup* obtido)

A Tabela 4 demonstra os valores dos tempos de quando os dados foram centralizados.

Quando os dados estavam centralizados em um único local, considerando também o melhor caso para o número de linhas (Tabela 4) obteve-se melhor desempenho com 32 processos. Obteve-se um *speedup* de $8,60$. Porém, novamente com 64 processos houve uma piora no desempenho.

Com base nessas observações, pode se verificar a influência de certos fatores referentes ao tamanho das mensagens e a quantidade de arquivos lidos para a execução do modelo e escritos após o processamento dos dados. Observando os valores obtidos de *speedup* e eficiência o melhor desempenho foi obtido com 64 processos (dados distribuídos) e 32 processos (dados centralizados). Em ambos os casos o desempenho computacional melhorou aproximadamente 90%. No entanto, estes resultados dizem respeito somente com as características desta arquitetura e com dados de quatro localidades (seção 6.1.1).

Tabela 4. CSM-Cropsim: Wheat: 400 linhas de execução e dados centralizados

Processos	Tempo médio (seg)	Speedup	Eficiência (%)	Custo
2	4931,08	0,53	26,28%	9862,16
4	3454,07	0,75	18,76%	13816,28
8	1607,31	1,61	20,16%	12858,48
16	797,79	3,25	20,31%	12764,64
32	301,33	8,60	26,88%	9642,56
64	724,48	3,58	5,59%	46366,72

6.2 MOC

Nesta seção serão demonstrados em qual ambiente computacional foram realizados os testes da paralelização da execução do MOC, posterior à isto é abordado como foi realizada a coleta dos resultados e finalmente os resultados obtidos.

6.2.1 Ambiente de Execução e Simulação

Visando obter uma comparação entre os ambientes computacionais disponíveis nesta pesquisa, no segundo estudo de caso, os testes foram realizados em dois ambientes computacionais diferentes. Inicialmente, utilizou-se uma máquina servidora do CPTEC/INPE, com 8 processadores físicos Intel(R) Xeon(R) processador E7- 4807 @ 1.87GHz.

Posteriormente, também foram realizados testes nas máquinas do mesmo laboratório da UPF que foi testado a paralelização do CSM-Cropsim: Wheat, cujas configurações são as mesmas descritas na seção 6.1.1.

Os dados utilizados para a realização do testes para o MOC correspondem à dados meteorológicos de 36 cidades do Brasil provenientes das bases de dados do CPTEC/INPE.

6.2.2 Estruturação dos testes

Todos os valores de tempos de execução apresentados resultam da média dos valores obtidos a partir de 10 repetições das execuções para que possa haver uma maior precisão dos resultados. Estas execuções foram realizadas em um primeiro momento no servidor do CPTEC/INPE após às 19h00min, horário que não havia expediente e no final de semana após às 12h00min do sábado. Em um segundo momento os testes foram realizados no momento foram realizados nos laboratórios da UPF, em horários em que não havia uso da rede do laboratório pelos alunos, uma vez que os laboratórios estava fechados para os alunos entre às 22h30min às 07h00min e aos finais de semana entre às 11h30min do sábado até às 07h00min da manhã da segunda-feira.

Nas duas arquiteturas os dados foram armazenados em um único local, no servidor do CPTEC/INPE foram armazenados na própria máquina que processou os dados. Nos computadores da universidade foram armazenados em um servidor na rede.

Na (Figura 13 A) é demonstrado o *shell script* que automatizou a execução das repetições, onde cada linha corresponde ao comando para executar o *shell script* (Figura 13 A) que realiza à execução paralela das 36 cidades cada qual com diferentes números de processos.

A

execucaomoc.sh

```
#/bin/bash
./start_moc_paralelo.ksh ONS 20150701 >> tempo2
cp arq1 pasta1
rm pasta1/arq1

./start_moc_paralelo4.ksh ONS 20150701 >> tempo32
cp arq1 pasta1
rm pasta1/arq1

./start_moc_paralelo3.ksh ONS 20150701 >> tempo8
cp arq1 pasta1
rm pasta1/arq1

./start_moc_paralelo2.ksh ONS 20150701 >> tempo4
cp arq1 pasta1
rm pasta1/arq1

./start_moc_paralelo.ksh ONS 20150701 >> tempo2
cp arq1 pasta1
rm pasta1/arq1

./start_moc_paralelo4.ksh ONS 20150701 >> tempo32
cp arq1 pasta1
rm pasta1/arq1

./start_moc_paralelo3.ksh ONS 20150701 >> tempo8
```

B

start_moc_paralelo.ksh

```
#!/bin/ksh -x
#
#
#
Lista_Estacoes=$1
if (($#<2)) ; then
   DATA=`date +%Y%m%d`
else
   DATA=${2}
fi

# Alterar o caminho do arquivo de configurações
source ~/publico/.../ucl/configure_${Lista_Estacoes}

cd ${DIRMOC_SCR}
mpirun -np 2 ${DIRMOC_SCR}/mestremoc ${Lista_Estacoes} ${DATA}

###############################################
# Gera arquivos para formato ONS
###############################################
${DIRFTP_SCR}/gera_arq_ONS.R ${DATA}
###############################################
# Disponibiliza arquivos para ONS
###############################################
#${DIRMOC_SCR}/put_ONSArqs_ftp.ksh ${DATA}
###############################################
```

Figura 13. Script das linhas de execução do MOC A, *shell script* execução paralela do MOC.

Este *shell script* possui uma estrutura similar ao da Figura 10 porém vai ser executado o arquivo "start_moc_paralelo"que chama o *shell script* configure da lista de estações (Figura 13 B), após o *shell script* da execução paralela do MOC.

6.2.3 Resultados

A Tabela 5 apresenta os valores de tempo gastos pelo MOC na máquina servidora do CPTEC/INPE com memória compartilhada.

Tabela 5. Resultados do servidor do INPE

Processos	Tempo médio (seg)	Speedup	Eficiência (%)	Custo
2	638,80	0,99	49,61	1277,59
3	331,00	1,91	63,76	994,03
4	229,00	2,77	69,14	916,75
5	177,00	3,57	71,48	886,72
6	148,00	4,30	71,59	885,40
7	116,00	5,49	78,36	808,86
8	101,00	6,28	78,56	806,78

Analisando os dados ilustrados na Tabela 5, observa-se que com o aumento do número de processos houve diminuição do tempo necessário para execução do modelo. Devido o MOC utilizar capacidade de processamento na realização de diversos cálculos o desempenho computacional tende melhorar ao acrescentar mais processos mesmo com os dados de entrada e saída estando centralizados em uma única máquina. Isto indica que este modelo de simulação pode ser considerado altamente paralelizável.

Analisando os tempos de execução do algoritmo paralelo apresentados na Tabela 5 observa-se que se obtiveram os menores tempos de execução, e um *speedup* máximo de 6,28 (Figura 14) e uma eficiência de 78,56%. O *speedup* de 0,99 deve-se ao fato de que apesar de serem 2 processos, um deles é o mestre, e o outro é o escravo que efetuou os cálculos.

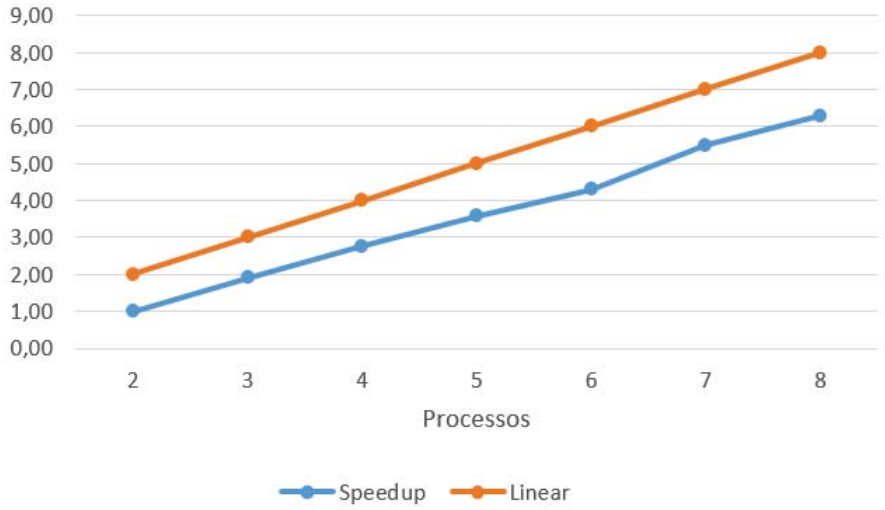

Figura 14. Gráfico de *speedup* CPTEC/INPE.

Pode ser observado na Figura 14, que nestes testes e com estas configurações, obteve-se um ganho quase relativo ao *speedup* teórico, pois, conforme os processos foram sendo adicionados observando a proporção que ele cresce, o tempo necessário para processamento dos dados diminuiu. Observa-se que não houve um *speedup* superlinear, pois manteve-se dentro do esperado em um aplicação paralela.

Após os testes com o servidor, foram realizados testes no laboratório da universidade, o mesmo já utilizado para os testes com o estudo de caso do modelo CSM-Cropsim: Wheat. Na Tabela 6 são apresentados os respectivos resultados.

Tabela 6. Testes realizados na universidade

Processos	Tempo médio	Speedup	Eficiência (%)	Custo
2	691,07	1,04	52,17	1382,13
4	594,94	1,21	30,30	2379,74
8	244,87	2,94	36,81	1958,94
16	116,30	6,20	38,75	1860,83
32	144,53	4,99	15,59	4625,07

Como pode ser observado quando realizados os testes nas máquinas do laboratório da universidade, devido à quantidade de comunicações, ou seja, troca de mensagens entre os processos, pôde ser observado que com 32 processos o desempenho foi pior. Comparando o speedup obtido com o esperando conclui-se que o desempenho foi bastante abaixo do ideal, mesmo assim, obteve-se aproximadamente 17% de melhora no tempo de execução e uma eficiência de $38,75\%$. Mesmo assim, obteve-se alguma melhora no desempenho e uma eficiência de 38,75% analisando o melhor resultado que foi com 16 processos. A causa deste resultado deve-se possivelmente pelo fato que há ainda um grande fluxo de informações trafegando pela rede o que é menos eficiente.

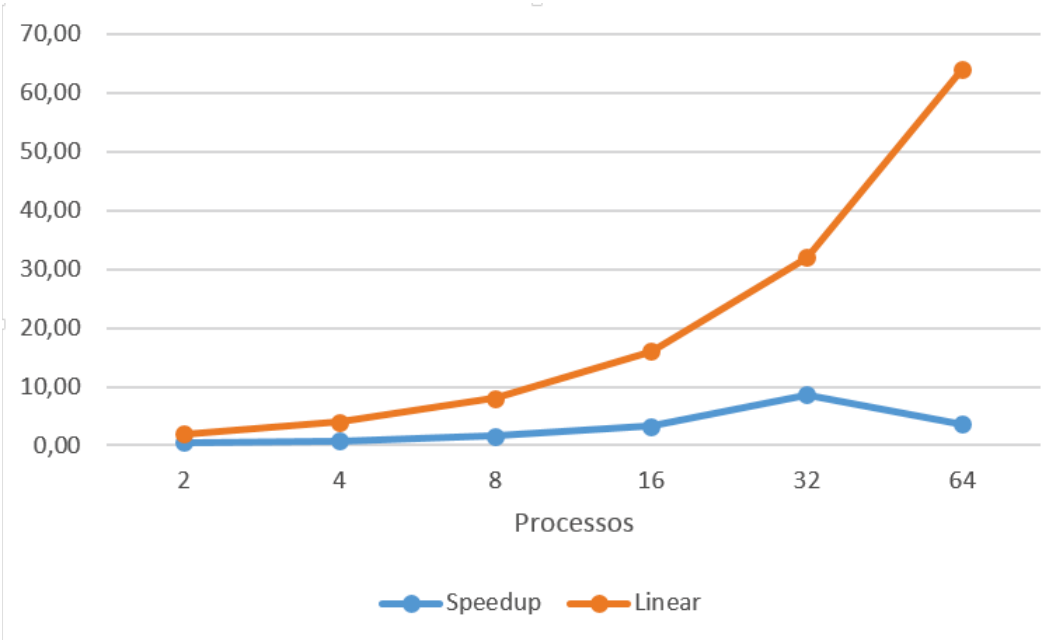

Figura 15. Gráfico de *speedup* UPF.

Embasado nestas observações, obteve-se melhora no desempenho utilizando qualquer uma das arquiteturas computacionais disponíveis. Verificando os resultados e comparando os mesmos levando em consideração estas arquiteturas computacionais a que melhor demonstrou ganhos de desempenho foi o servidor com 8 núcleos de processamento pois não houve tempo de comunicação na rede e sim apenas entre os processos. A diferença de eficiência entre o servidor do CPTEC/INPE utilizando 8 processos foi de $58,4\%$ maior que os resultados da arquitetura do laboratório da universidade.

6.3 CONCLUSÕES DO CAPÍTULO

Neste capítulo foram apresentados os testes realizados para os dois estudos de caso deste trabalho. Conforme o resultado dos testes, pôde ser observado que cada modelo de simulação continha diferenças não apenas em relação ao objetivo proposto por cada um mas também no que diz respeito ao tempo gasto em leitura e escrita.

Concluiu-se, portanto, que o CSM-Cropsim: Wheat consome menos tempo realizando computações e mais leitura e escrita de arquivos e, com isso, houve um desperdício de tempo pelas

comunicações necessárias de acesso aos arquivos de leitura e escrita. Já o MOC realiza mais cálculos e, consequentemente, apresentou melhores resultados em questões de desempenho computacional paralelo.

Para que fossem obtidos melhores ganhos no tempo despendido pelo modelo CSM-Cropsim: Wheat, os dados foram distribuídos entre as máquinas, e, com isso, cada processo não necessitou acessar arquivos em uma única máquina e nem usar a rede para este acesso. Em relação à quantidade de linhas de comando enviadas para cada processo é importante levar em consideração que o tamanho da mensagem enviada influencia no tempo despendido.

Apesar do CSM-Cropsim: Wheat apresentar processos *I/O bound* e o MOC processos cpu *bound*, para os dois modelos foram obtidos reduções no tempo necessário para a execução da simulação. No caso do modelo CSM-Cropsim: Wheat, o ganho no desempenho foi decorrente principalmente de acesso *I/O*. O MOC como é mais facilmente paralelizável os resultados foram de encontro ao esperado pela aplicação da paralelização.

É importante salientar que a execução paralela do MOC já está sendo utilizado pelo CPETEC/INPE desde julho de 2015. A previsão corrigida está sendo disponibilizada para o ONS, diariamente, no endereço ftp://ftp1.cptec.inpe.br/etamdl/MOC/. Como benefício adicional, a versão paralela mostrada neste texto já está auxiliando no desempenho da execução, o que facilitará a ampliação da correção da previsão do tempo para mais localidades brasileiras.

Já a paralelização do CSM-Cropsim: Wheat está auxiliando um conjunto amplo de pesquisadores e profissionais da área agrícola interessados na execução de modelos de simulação de culturas e doenças em larga escala e em um tempo menor. Além disso, proporcionará aos pesquisadores e profissionais da área de Tecnologia da Informação um maior conhecimento a respeito do uso de técnicas paralelas em aplicações da informática na área agrícola e de meteorologia.

7. CONCLUSÃO

Neste trabalho foram estudados dois modelos de simulação, o primeiro, utilizado pela Universidade de Passo Fundo e Embrapa Trigo, que simula o crescimento do trigo, CSM-Cropsim: Wheat, o qual auxilia profissionais responsáveis na tomada de decisão com o intuito de melhorar a produtividade da cultura em questão. O segundo modelo, MOC, realiza a correção estatística da previsão do tempo gerada pelo modelo Eta do CPTEC-INPE. As informações meteorológicas corrigidas pelo MOC, como temperatura e umidade relativa, também são utilizadas como dados de entrada para a execução da simulação do CSM-Cropsim: Wheat, existindo assim, uma relação importante entre os dois modelos do estudo.

Observou-se que havia a necessidade em ambos os casos da utilização de maiores quantidade de dados e, com isso, aumentar a área de abrangência geográfica e/ou também de dados históricos processados pelos modelos. Devido a este fato iniciou-se o estudo do funcionamento e da execução sequencial de cada um dos modelos. Posteriormente, avaliou-se possíveis técnicas computacionais que proporcionassem o processamento de maiores quantidades de dados de forma efetiva e em tempo computacional hábil.

Entre estas técnicas estão algoritmos, linguagens e bibliotecas para a computação paralela. Conforme as características dos modelos e das aplicações a que eles simulam, optou-se pela utilização da biblioteca MPI, com a metodologia do algoritmo mestre/escravo. Para validar esta escolha foram realizados testes utilizando duas arquiteturas computacionais: um sistema constituído de 16 máquinas do Laboratório Central de Informática da UPF e um servidor com 8 processadores físicos do CPTEC-INPE.

Após a realização dos testes para cada um dos estudos de caso, observou-se que na execução do CSM-Cropsim: Wheat ocorreram muitas comunicações de entrada e saída de dados (*I/O bound*). Partindo da hipótese que a rede estava aumentando o tempo de comunicação, haja visto que o CSM-Cropsim: Wheat realiza muitas comunicações de entrada de saída de dados, optou-se por distribuir os dados entre as máquinas visto que, após realizar as comparações entre a utilização de dados distribuídos e com os dados armazenados em uma única máquina, observou-se que foram obtidos melhores resultados nos tempos de execução utilizando a abordagem dos dados distribuídos entre as máquinas. Aplicando esta abordagem obteve-se resultados satisfatórios na paralelização de sua execução, pois, com 64 processadores, obteve-se um tempo de $187,13$ segundos com *speedup* de $13,85$, enquanto que o tempo sequencial para processar a mesma quantidade de dados foi de 2592 segundos. Como este modelo possuía muitas execuções, foram enviadas determinadas quantidade de execuções para cada processador e isto também influenciou no desempenho obtido. Com esta abordagem obteve-se um melhor desempenho computacional quando foi enviado 50 linhas de comandos (equivalente a 50 execuções do modelo) para cada processador.

Nos resultados dos testes da paralelização da execução do MOC foi observado que o modelo efetua menos leitura e escrita de dados e mais cálculos (cpu *bound*). Após a realização

dos testes no servidor do CPTEC, utilizando até 8 processadores, observou-se que o ganho no tempo computacional foi linear, concluindo, assim, que este modelo é altamente paralelizável e que é possível em trabalhos futuros obter melhores desempenhos com a paralelização implícita deste modelo, ou seja, ainda existe a opção de paralelizar as equações responsáveis pela correção das previsões horárias. Em testes realizados com o MOC, também nas máquinas do laboratório da UPF, obteve-se bons desempenhos computacionais, porém, como no caso dos testes com o CSM-Cropsim: Wheat, o obstáculo encontrado foi em relação à arquitetura da rede que estava sendo compartilhada com outros usuários, repercutindo assim no gasto de uma parcela maior do tempo nas comunicações feitas pelos processos.

Portanto, conclui-se que a o desempenho computacional da execução paralela da execução de modelos de simulação depende de alguns fatores principais, entre eles está se o modelo realiza mais ou menos cálculos e/ou mais leitura e escrita de arquivos. Outro fator importante é a arquitetura computacional a ser utilizada para a execução dos modelos. A partir dos resultados deste trabalho, levando em consideração os recursos computacionais disponíveis na UPF e no CPTEC-INPE, obteve-se melhor desempenho quando utilizou-se 32 e 8 processadores físicos, respectivamente.

Por fim, com este trabalho, buscou-se otimizar o desempenho computacional da execução dos modelos de simulação objetivando uma maior abrangência na utilização da quantidade de dados processados por estes modelos e, assim, auxiliar com a prática do desenvolvimento de programação paralela para os profissionais da computação e meteorologia envolvidos nos modelos, para os produtores rurais na tomada de decisão em relação às culturas e para pesquisadores interessados nos benefícios da utilização dos modelos de simulação de culturas e de previsão de tempo em aplicações na área agrícola.

7.1 PUBLICAÇÕES

Entre as publicações já realizadas relacionadas à este trabalho estão, na XV Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul, edição de 2015 realizada em Gramado, publicação com o título: Execução Paralela de Modelos de Simulação de Culturas e Doenças; no Encontro Anual de Tecnologia da Informação em 2014, realizado em Frederico Westphalen, publicação com o título: Computação de alto desempenho em R: paralelização e técnicas de otimização; e na Semana do Conhecimento da UPF em 2015 realizada em Passo Fundo.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] CHOU, S. C. et al. Refinamento estatístico das previsões horárias de temperatura a 2 m do modelo eta em estações do nordeste do brasil. *Revista Brasileira de Meteorologia*, Scielo, v. 22, p. 287 – 296, 12 2007. ISSN 0102-7786. Disponível em: <http://www.scielo.br/pdf/rbmet/v22n3/01.pdf>. Acesso em: 20 Jul. 2015.

[2] HUNT, L. A.; PARARAJASINGHAM, S. Cropsim-wheat: A model describing the growth and development of wheat. *Canadianjournal Plant Science*, Department of Crop Science, University of Guelph, Guelph, Ontario, Canada, 1995. Disponível em: <http://pubs.aic.ca/doi/pdf/10.4141/cjps95-107>. Acesso em: 11 Jul. 2014.

[3] MAO, Q. et al. An Optimal Model Output Calibration Algorithm Suitable for Objective Temperature Forecasting. *Weather and Forecasting*, v. 14, n. 2, p. 190–202, abr. 1999. ISSN 0882-8156. Disponível em: <http://journals.ametsoc.org/doi/abs/10.1175/1520-0434(1999)014%3C0190%3AAOMOCA%3E2.0.CO%3B2>. Acesso em: 11 Mai. 2014.

[4] PONTE, e. a. E. M. D. A model-based assessment of the impacts of climate variability on fusarium head blight seasonal risk in southern brazil. *Journal of Phytopathology*, v. 157, p. 675–681, 2009.

[5] PAVAN, W. *Técnicas de Engenharia de Software Aplicadas à Modelagem e Simulação de Doenças de Plantas.* Tese (Doutorado) — UPF - Faculdade de Agronomia e Medicina Veterinária, Programa de Pós Graduação em agronomia., 2007. Disponível em: <www.ppgagro.upf.br/download/Willingthon_Pavan.pdf>. Acesso em: 15 Ago. 2014.

[6] LAZZARETTI, A. T. *Interação de banco de dados e modelos de simulação de culturas para estimar o impacto de mudanças do clima no rendimento de grãos e na severidade da giberela no trigo.* Tese (Doutorado) — UPF - Faculdade de Agronomia e Medicina Veterinária, Programa de Pós Graduação em agronomia., 2013. Disponível em: <www.ppgagro.upf.br/images/stories/alexandre-tagliari-lazzareti.pdf>. Acesso em: 20 Ago. 2014.

[7] ERCAN, M. B. et al. Calibration of SWAT models using the cloud. *Environmental Modelling and Software*, Elsevier Ltd, v. 62, p. 188–196, 2014. ISSN 1364-8152. Disponível em: <http://dx.doi.org/10.1016/j.envsoft.2014.09.002>. Acesso em: 20 Jun. 2015.

[8] JORDI, A.; WANG, D. P. SbPOM: A parallel implementation of Princenton Ocean Model. *Environmental Modelling and Software*, Elsevier Ltd, v. 38, p. 59–61, 2012. ISSN 13648152. Disponível em: <http://dx.doi.org/10.1016/j.envsoft.2012.05.013>. Acesso em: 11 Nov. 2015.

[9] LARGE-SCALE, high-resolution agricultural systems modeling using a hybrid approach combining grid computing and parallel processing. *Environmental Modelling &*

Software, v. 41, n. 0, p. 231 – 238, 2013. ISSN 1364-8152. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1364815212002277>. Acesso em: 20 Jun. 2015.

[10] SCHMIDBERGER, M. et al. State of the art in parallel computing with R. *Journal of Statistical Software.*, v. 31, p. 1–27, 2009. Disponível em: <http://www.jstatsoft.org/v31/i01/paper>. Acesso em: 20 Jun. 2015.

[11] KUMAR, P. et al. High performance data mining using R on heterogeneous platforms. In: *Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum*. [s.n.], 2011. (IPDPSW '11), p. 1720–1729. ISBN 978-0-7695-4577-6. Disponível em: <http://dx.doi.org/10.1109/IPDPS.2011.329>. Acesso em: 12 Nov. 2014.

[12] JIANG, L. et al. Openmp-style parallelism in data-centered multicore computing with R. In: *Proceedings of the 17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. New York, NY, USA: ACM, 2012. (PPoPP '12), p. 335–336. ISBN 978-1-4503-1160-1. Disponível em: <http://doi.acm.org/10.1145/2145816.2145882>. Acesso em: 20 Nov. 2014.

[13] PETROU, S. et al. Optimization of a parallel permutation testing function for the SPRINT R package. *Concurrency and computation : practice & experience*, v. 23, n. 17, p. 2258–2268, 2011. ISSN 1532-0626. Disponível em: <http://www.pubmedcentral.nih.gov/>. Acesso em: 20 Nov. 2014.

[14] VENKATARAMAN, S. et al. Using r for iterative and incremental processing. In: *Proceedings of the 4th USENIX Conference on Hot Topics in Cloud Ccomputing*. Berkeley, CA, USA: USENIX Association, 2012. (HotCloud'12), p. 11–11. Disponível em: <http://dl.acm.org/citation.cfm?id=2342763.2342774>. Acesso em: 20 Nov. 2014.

[15] PADBERG, F.; MIROLD, M. An experimentation platform for the automatic parallelization of R programs. In: LEUNG, K. R. P. H.; MUENCHAISRI, P. (Ed.). *APSEC*. IEEE, 2012. p. 203–212. Disponível em: <http://dblp.uni-trier.de/db/conf/apsec/apsec2012.html#PadbergM12>. Acesso em: 20 Nov. 2014.

[16] SAMATOVA, N. F. et al. High performance statistical computing with parallel R: applications to biology and climate modelling. *Journal of Physics: Conference Series.*, v. 46, p. 505–509, 2006. Disponível em: <http://dx.doi.org/10.1088/1742-6596/46/1/069>. Acesso em: 20 Nov. 2014.

[17] MITCHELL, L. et al. A parallel random forest classifier for R. In: *Proceedings of the Second International Workshop on Emerging Computational Methods for the Life Sciences*. New York, NY, USA: ACM, 2011. (ECMLS '11), p. 1–6. ISBN 978-1-4503-0702-4. Disponível em: <http://doi.acm.org/10.1145/1996023.1996024>. Acesso em: 11 Nov. 2014.

[18] BRETT, A. B. High-performance computing tools for the integrated assessment and modelling of social–ecological systems. *Environmental Modelling & Software*, v. 39, p. 295 – 303, 2013.

ISSN 1364-8152. Thematic Issue on the Future of Integrated Modeling Science and Technology. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1364815212000540>. Acesso em: 20 Ago. 2015.

[19] XIE, X. et al. Jrbridge: A framework of large-scale statistical computing for R. In: *Proceedings of the 2012 IEEE Asia-Pacific Services Computing Conference)*. Washington, DC, USA: IEEE Computer Society, 2012. (APSCC '12), p. 27–34. ISBN 978-0-7695-4897-5. Disponível em: <http://dx.doi.org/10.1109/APSCC.2012.74>. Acesso em: 20 Nov. 2014.

[20] EDDELBUETTEL, D.; SANDERSON, C. Rcpparmadillo: Accelerating R with high-performance C++ linear algebra. *Computational Statistics & Data Analysis*, v. 71, p. 1054 – 1063, 2014. ISSN 0167-9473. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167947313000492>. Acesso em: 20 Out. 2014.

[21] HOFFMANN, T.; LANGE, C. P2bat: A massive parallel implementation of pbat for genome-wide association studies in R. *Bioinformatics*, Oxford University Press, Oxford, UK, v. 22, n. 24, p. 3103–3105, dez. 2006. ISSN 1367-4803. Disponível em: <http://dx.doi.org/10.1093/bioinformatics/btl507>. Acesso em: 20 Out. 2014.

[22] LI, J. et al. Transparent runtime parallelization of the R scripting language. *Journal of Parallel and Distributed Computing*, v. 71, n. 2, p. 157 – 168, 2011. ISSN 0743-7315. Data Intensive Computing. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0743731510001747>. Acesso em: 10 Out. 2014.

[23] VERA, G.; SUPPI, R. Integration of heterogeneous and non-dedicated environments for R. In: *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*. [S.l.: s.n.], 2010. p. 667–672.

[24] STOKELY, M.; ROHANI, F.; TASSONE, E. Large-scale parallel statistical forecasting computations in R. *JSM Proceedings*, American Statistical Association, 2011. Disponível em: <http://static.googleusercontent.com/media/research.google.com/pt-BR//pubs/archive/37483.pdf>. Acesso em: 20 Nov. 2014.

[25] DAS, S. et al. Ricardo: Integrating R and hadoop. In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: ACM, 2010. (SIGMOD '10), p. 987–998. ISBN 978-1-4503-0032-2. Disponível em: <http://doi.acm.org/10.1145/1807167.1807275>. Acesso em: 20 Nov. 2014.

[26] ELLIOTT, J. et al. The parallel system for integrating impact models and sectors (pSIMS). *Environmental Modelling & Software*, v. 62, p. 509–516, dez. 2014. ISSN 13648152. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1364815214001121>. Acesso em: 18 Abr. 2015.

[27] ELLIOTT, J. et al. The parallel system for integrating impact models and sectors (pSIMS). In: *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery*. New York, NY, USA: ACM, 2013. (XSEDE '13), p. 21:1–21:8. ISBN 978-1-4503-2170-9. Disponível em: <http://doi.acm.org/10.1145/2484762.2484814>. Acesso em: 20 Nov. 2014.

[28] BRYAN, B. A.; KING, D.; ZHAO, G. Influence of management and environment on australian wheat: information for sustainable intensification and closing yield gaps. *Environmental Research Letters*, v. 9, p. 1–12, 2014. Disponível em: <http://dx.doi.org/10.1088/1748-9326/9/4/044005>. Acesso em: 20 Nov. 2014.

[29] LIU, Y. e. a. Optimization of agricultural bmps using a parallel computing based multi-objective optimization algorithm. *The International Journal of Environmental Resources Research*, v. 1, n. 1, p. 39–50, 2012.

[30] YALEW, S. et al. Distributed computation of large scale {SWAT} models on the grid. *Environmental Modelling & Software*, v. 41, n. 0, p. 223–230, 2013. ISSN 1364-8152. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1364815212002125>. Acesso em: 20 Out. 2014.

[31] CHEN, N. et al. A parallel implementation of the Cellular Potts Model for simulation of cell-based morphogenesis. *Computer Physics Communications*, v. 176, n. 11-12, p. 670–681, 2007. ISSN 00104655.

[32] CHENG, C. et al. Parallel discrete differential dynamic programming for multireservoir operation. *Environmental Modelling and Software*, Elsevier Ltd, v. 57, p. 152–164, 2014. ISSN 13648152. Disponível em: <http://dx.doi.org/10.1016/j.envsoft.2014.02.018>. Acesso em: 8 Mar. 2015.

[33] LIEBER, M.; WOLKE, R. Optimizing the coupling in parallel air quality model systems. *Environmental Modelling and Software*, v. 23, n. 2, p. 235–243, 2008. ISSN 13648152.

[34] RAO, P. A parallel rma2 model for simulating large-scale free surface flows. *Environmental Modelling and Software*, v. 20, n. 1, p. 47–53, 2005. ISSN 13648152.

[35] SMIATEK, G. Parallelization of a grid-oriented model on the example of a biogenic volatile organic compounds emission model. *Environmental Modelling and Software*, v. 23, n. 12, p. 1468–1473, 2008. ISSN 13648152.

[36] TEIJEIRO, C. et al. Parallel brownian dynamics simulations with the message-passing and pgas programming models. *Computer Physics Communications*, Elsevier B.V., v. 184, n. 4, p. 1191–1202, 2013. ISSN 00104655. Disponível em: <http://dx.doi.org/10.1016/j.cpc.2012.12.015>. Acesso em: 23 Mai. 2015.

[37] TESFA, T. K. et al. Extraction of hydrological proximity measures from dems using parallel processing. *Environmental Modelling and Software*, Elsevier Ltd, v. 26, n. 12, p. 1696–1709, 2011. ISSN 13648152. Disponível em: <http://dx.doi.org/10.1016/j.envsoft.2011.07.018>. Acesso em: 8 Mar. 2015.

[38] YABLONSKY, R. M.; GINIS, I.; THOMAS, B. Ocean modeling with flexible initialization for improved coupled tropical cyclone-ocean model prediction. *Environmental Modelling & Software*, Elsevier Ltd, v. 67, p. 26–30, 2015. ISSN 13648152. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S1364815215000213>. Acesso em: 8 Nov. 2015.

[39] ZHANG, X. et al. Efficient multi-objective calibration of a computationally intensive hydrologic model with parallel computing software in Python. *Environmental Modelling and Software*, Elsevier Ltd, v. 46, p. 208–218, 2013. ISSN 13648152. Disponível em: <http://dx.doi.org/10.1016/j.envsoft.2013.03.013>. Acesso em: 20 Jan. 2015.

[40] MESINGER, F. et al. *The Step-Mountain Coordinate: Model Description and Performance for Cases of Alpine Lee Cyclogenesis and for a Case of an Appalachian Redevelopment*. 1988. 1493–1518 p.

[41] BLACK, T. L. *The New NMC Mesoscale Eta Model: Description and Forecast Examples*. 1994. 265–278 p.

[42] CHOU, S. C. et al. Evaluation of the Eta Simulations Nested in Three Global Climate Models. *American Journal of Climate Change*, Scientific Research Publishing, v. 03, n. 05, p. 438–454, dec 2014. ISSN 2167-9495. Disponível em: <http://www.scirp.org/>. Acesso em: 20 Jun. 2015.

[43] CPTEC-INPE. *Eta model*. 1996–2006. Disponível em: <http://etamodel.cptec.inpe.br/index.shtml>. Acesso em: 13 Jul. 2015.

[44] CHOU, S. C. *Modelagem Numérica Introdução a PNT*. 2010. Disponível em: <http://migre.me/qb8SR>. Acesso em: 20 Jun. 2015.

[45] RESEARCH, U. C. f. A. *Ch. 9: Observation, Analysis, and Prediction*. 2011. Disponível em: <http://www.goes-r.gov/>. Acesso em: 20 Jun. 2015.

[46] Vieira Junior, P. A. et al. Previsões meteorológicas do Modelo Eta para subsidiar o uso de modelos de previsão agrícola no Centro-Sul do Brasil. *Ciência Rural*, Universidade Federal de Santa Maria, v. 39, n. 2, p. 412–420, abr. 2009. ISSN 0103-8478. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0103-84782009000200015&lng=en&nrm=iso&tlng=pt>. Acesso em: 17 Out. 2015.

[47] GLAHN, H. R.; LOWRY, D. A. The use of model output statistics (mos) in objective weather forecasting. *J. Appl. Meteor*, v. 11, p. 1203–1211, 1972.

[48] HADKA, D.; REED, P. Large-scale parallelization of the Borg multiobjective evolutionary algorithm to enhance the management of complex environmental systems. *Environmental Modelling & Software*, Elsevier Ltd, v. 69, p. 353–369, 2015. ISSN 13648152. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S1364815214003041>. Acesso em: 28 Jan. 2015.

[49] LI, T. et al. Dynamic parallelization of hydrological model simulations. *Environmental Modelling and Software*, Elsevier Ltd, v. 26, n. 12, p. 1736–1746, 2011. ISSN 13648152. Disponível em: <http://dx.doi.org/10.1016/j.envsoft.2011.07.015>. Acesso em: 8 Mar. 2015.

[50] SENA M. C. R.; COSTA, J. A. C. *Tutorial OpenMP C/C++*. [S.l.], 2008. Disponível em: <http://www.lncc.br>. Acesso em: 13 Out. 2014.

[51] UFRGS. *Caderno dos Cursos Permanentes das Escolas Regionais de Alto Desempenho*. [S.l.: s.n.].