Revista Brasileira de Computação Aplicada, Novembro, 2018

DOI: 10.5335/rbca.v10i3.8663
Vol. 10, Nº 3, pp. 34-43
Homepage: seer.upf.br/index.php/rbca/index

ISSN 2176-6649

ARTIGO ORIGINAL

Modelo de simulação de cultura CSM-Cropsim: Wheat - uma abordagem paralela de execução

Angela Mazzonetto[1], Marcelo Trindade Rebonatto[1], Willingthon Pavan[1] and Carlos Amaral Hölbig[1]

[1]Programa de Pós-Graduação em Computação Aplicada - Universidade de Passo Fundo

*angelamazzonettofw@gmail.com; rebonatto@upf.br; pavan@upf.br; holbig@upf.br

Submetido: 12/07/2018. **Revisado:** 23/10/2018. **Aceito:** 07/11/2018.

Abstract

This paper shows a parallel computational approach that aims to expand the coverage area and/or the time series to be simulated by the wheat growth model called CSM-Cropsim: Wheat. The parallelization approach of the execution of the chosen model was the master-slave, along with the use of the MPI communication library, which proved to be adequate, since each of the executions (model rounds) are independent of the results of the others in the evaluated model . The results obtained with the tests performed proved to be satisfactory, demonstrating the validity of the application of this approach in the large scale execution of CSM-Cropsim: Wheat.

Key words: CSM-Cropsim: Wheat; Parallel execution; Crop model.

Resumo

Este trabalho apresenta uma abordagem computacional paralela que visa a ampliação da área de cobertura e/ou da série temporal a ser simulada pelo modelo de crescimento da cultura do trigo chamado de CSM-Cropsim: Wheat. A abordagem de paralelização da execução do modelo escolhida foi a mestre-escravo, juntamente com a utilização da biblioteca de comunicação MPI, que mostrou ser adequada, visto que cada uma das execuções (rodadas do modelo) são independentes dos resultados das outras no modelo avaliado. Os resultados obtidos com os testes realizados demonstraram-se satisfatórios, demonstrando a validade da aplicação desta abordagem na execução em larga escala do CSM-Cropsim: Wheat.

Palavras-Chave: CSM-Cropsim: Wheat; Execução paralela; Modelo de cultura.

1 Introdução

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. Bases de dados como, por exemplo, do Centro de Previsão de Tempo e Estudos Climáticos do Instituto Nacional de Pesquisas Espaciais (CPTEC/INPE), do Instituto Nacional de Meteorologia (INMET), do Agritempo - Sistema de Monitoramento Agrometeorológico, do National Oceanic and Atmospheric Administration (NOAA), do World Development Indicators (WDI), 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 em estimar possíveis impactos nos sistemas de produção de culturas (Del Ponte et al.; 2009).

Com a evolução tecnológica, a presença de novos coletores de dados meteorológicos tornou-se comum. Com esses coletores, a massa de dados aumentou 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 (Chou et al.; 2007).

Uma das áreas de maior aplicação destes dados é a área agrícola, onde modelos de simulação de culturas e doenças os utilizam. Os resultados gerados por estes modelos são posteriormente analisados e tem por objetivo auxiliar no processo de tomada de decisão. 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 à sua área de abrangência ou ao período a ser simulado. Além dos dados observados, dados gerados por outros modelos como, por exemplo, de previsão de tempo e clima, 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. Porém, com este aumento da granularidade, a quantidade de execuções aumenta exponencialmente nas mesmas proporções, e 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 estes motivos torna-se indispensável a utilização de ferramentas ou de técnicas computacionais de alto desempenho que visem a otimização e a melhora do desempenho computacional da execução destes modelos. Várias ferramentas podem ser utilizadas para suprir estas necessidades, desde as já tradicionais para computação paralela como o *Message Passing Interface* (MPI), utilizado no trabalho de Jordi and Wang (2012) 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), até *frameworks* como o Hadoop ou linguagens de programação como o R (Eugster et al.; 2011; Schmidberger et al.; 2009). Obviamente, o uso destas ferramentas deve estar aliado a escolha correta de uma abordagem eficiente de paralelização. A alternativa escolhida por esta pesquisa baseia-se no uso do processamento paralelo, possibilitando a execução dos modelos de simulação por esta abordagem.

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.

Com o objetivo de apresentar alternativas viáveis de paralelização da execução de modelos de simulação ligados à área agrícola, este trabalho selecionou um modelo como estudo de caso para validar estas alternativas mediante alguns testes. Este estudo de caso é composto por um modelo de simulação de cultura do trigo, chamado de CSM-Cropsim: Wheat (Hunt and Pararajasingham; 1995), utilizado pela Embrapa Trigo e pela UPF. Portanto, o problema de pesquisa deste trabalho foi o de definir uma estratégia de paralelização da execução deste modelo.

2 Trabalhos relacionados

Vários trabalhos destacam a importância do uso de abordagens paralelas na resolução de aplicações na área agrícola.

O trabalho de Zhao et al. (2013) descreve 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.

Bryan et al. (2014) 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 e em Grid para simular o impacto na agricultura avaliando a influência de variáveis de gestão ambientais na produtividade do trigo. Embora a gestão de resíduos mostrou pouca correlação, a adubação aumentou fortemente o rendimento do trigo. No entanto, este efeito é altamente dependente das variáveis-chave do ambiente, como a precipitação, a temperatura e a capacidade de retenção de água do solo. O efeito da adubação sobre a produtividade foi mais forte em climas mais frios e úmidos e em solos com capacidade de reter água. Eles fornecem um contexto específico de informações sobre os benefícios da fertilização na produtividade para

apoiar a tomada de decisão adaptativa agronômica. Também sugerem que nas avaliações futuras de rendimento a sustentabilidade econômica, ambiental e a intensificação da gestão sejam consideradas para cobrir as lacunas da produtividade.

Liu et al Liu (2012) afirmam que *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 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.

Em sua pesquisa, Yalew et al. (2013) 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.

No trabalho descrito por Zhang et al Zhang et al. (2013), 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.

Além dos trabalhos já citados, outras pesquisas que abordam esta integração da computação paralela com aplicações agrícolas podem ser mencionadas como, por exemplo, os trabalhos de Prabhu and Dakshayini (2018), de Luo et al. (2018) e de Janssen et al. (2017).

Com base nos trabalhos relacionados nesta seção, observa-se que existem inúmeros modelos de simulação, nas mais diversas áreas de aplicação, que, aliando a paralelização, mesmo com diferentes metodologias, tem proporcionado melhoras significativas no aproveitamento dos dados processados e, ao mesmo tempo, no desempenho computacional das aplicações envolvidas.

3 Material e métodos

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 (Lazzaretti; 2013). 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 de crescimento do trigo CSM-Cropsim: Wheat, foi desenvolvido por Leslie A. Hunt (Hunt and Pararajasingham; 1995) na linguagem de programação Fortran e, atualmente, faz parte do sistema Decision Support System for Agrotechnology Transfer (DSSAT), 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 deve ser organizada a fim de se obter uma melhora em seu rendimento computacional em situações onde há uma grande área a ser coberta pelo modelo e/ou onde a série temporal a ser simulada é muito grande. Este modelo é utilizado no grupo de pesquisa Mosaico da Universidade de Passo Fundo e na Embrapa Trigo.

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 (Pavan; 2007).

Segundo Pavan (2007), 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 calculam a energia e a água disponíveis para o crescimento da planta de trigo, ao passo que o módulo planta de trigo simula os eventos

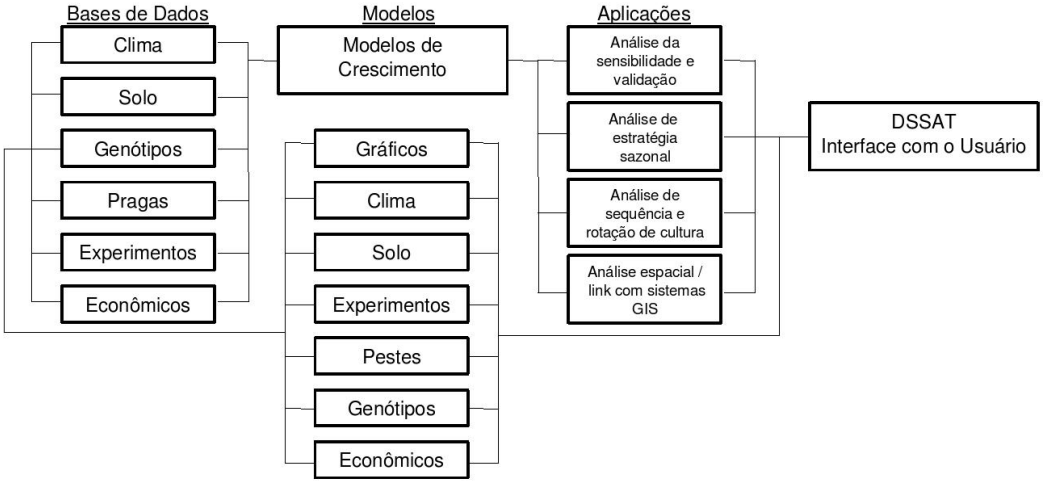

Figura 1: Estrutura da suíte do Sistema de Apoio à Decisão para Transferência de Agrotecnologia (DSSAT)

Fonte: (Pavan; 2007)

fenológicos, expansão foliar, acúmulo de carboidratos e a partição entre a parte aérea e raízes.

A funcionalidade deste modelo integra uma série de dados de entrada que compreendem dados meteorológicos, de solo, de fisiologia vegetal e 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.

Para um melhor gerenciamento destes arquivos, a aplicação realizada no trabalho de Lazzaretti (2013) 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 e os arquivos gerados pela sua aplicação são os seguintes:

- `.WTH`: arquivo com os dados meteorológicos. Os principais dados necessários 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.

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 (Lazzaretti; 2013), 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 cinco 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

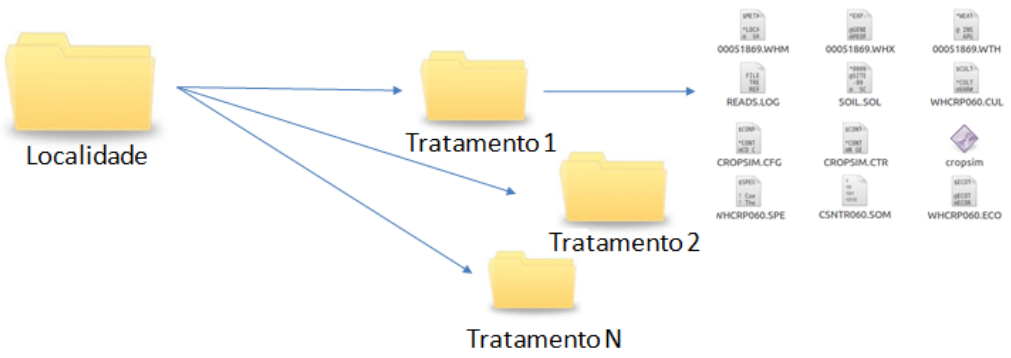

Figura 2: Estrutura dos diretórios montada para a execução do modelo CSM-Cropsim: Wheat

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 Estratégia de Paralelização

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 esta finalidade.

A paralelização deste modelo baseou-se no algoritmo paralelo mestre-escravo. Esta abordagem foi a que melhor se adaptou à estrutura de funcionamento do modelo, visto que cada uma de suas execuções (rodada do modelo) é independente das demais. 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 a este trabalho foram realizadas tentativas pelo grupo de pesquisa da Universidade de Passo Fundo para realizar a paralelização implícita do modelo, porém, após a análise do desempenho obtido e devido à estrutura de funcionamento do mesmo, chegou-se à conclusão de 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 and Reed (2015) 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 que utiliza esta abordagem é o realizado por Tiejian et al. (2011), 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. Pesquisas de trabalhos relacionando processamento paralelo também podem ser vistos nos realizados por Ross et al. (2016), onde é utilizada uma arquitetura de cluster e o algoritmo MPI para paralelização de cálculos numéricos. Em Jiang et al. (2016) é proposto o desenvolvimento de uma plataforma para aplicações paralelas com o intuito de utilizar no modelo do sistema terrestre, pois, segundo os autores este modelo necessita de alto desempenho computacional para a obtenção de resultados eficiêncientes. O trabalho realizado por van den Oord and Bakhshi (2017) descreve que a crescente resolução de modelos climáticos e meteorológicos resultou em um rápido crescimento da produção de dados. Isso exige uma abordagem moderna e eficiente para o pós-processamento desses dados. Para este fim, foi desenvolvido um pacote de software em Python que explora a natureza paralela da carga de trabalho pós-processamento para processar a saída do EC-Earth, um modelo acoplado oceano-oceano.

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 a um conjunto de dados referente a uma localidade. Por exemplo, se são necessárias 9600 execuções do modelo (isso corresponde a 4 localidades com 2400 tratamentos cada uma) serão enviadas 50 linhas de comandos de execução para cada processo até completar o total das 9600 execuções. 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 3). Quando todo o trabalho tiver chegado ao final, o mestre envia um sinal de fim de

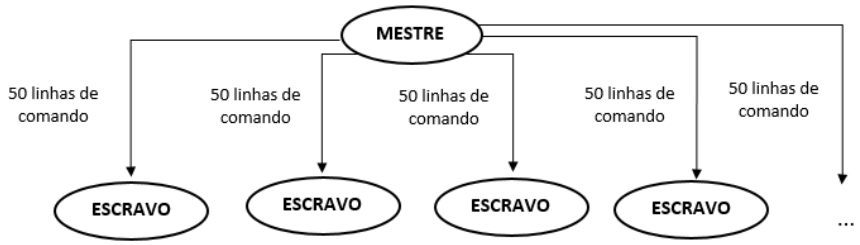

Figura 3: Modelo paralelo de execução do modelo CSM-Cropsim: Wheat

trabalho aos escravos.

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 infraestrutura para modalidades de computação paralela quando existe a necessidade de passar informações entre os vários processadores ou nodos de um *cluster* com memória distribuída. A interface do MPI foi estendida com funções para gerenciamento dinâmico de processos, entrada/saída paralela e acesso a memória remota, além de ser adaptada a arquiteturas com memória híbrida (compartilhada entre os núcleos e distribuída entre os nós).

Outra questão que afeta o desempenho da execução paralela é o aumento do número de processadores que, em um dado momento, pode tender a um aumento de tempo de execução em paralelo. Um exemplo desta situação é o trabalho descrito por Zhang et al. (2013), 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. Os autores descrevem o desenvolvimento de 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 este software desenvolvido alcançou 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.

4 Resultados e Discussão

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 mais desempenho computacional na divisão de trabalho entre os núcleos de processamento da máquina *Network of Workstations* (NOW).

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 (2013) e correspondem a quatro localidades do sul do Brasil. O tempo para a execução sequencial do modelo para estes dados foi de aproximadamente 43 minutos.

4.1 Coleta dos Resultados

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. Cabe salientar que havia mais usuários utilizando a rede da UPF. 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 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 de um comando 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 4 é demonstrado um exemplo do arquivo *shell script* do Linux para a execução dos testes.

execucao.sh ×

```
1 #/bin/bash
2
3 mpirun --hostfile hosts -np 8 mestre 200 >> lixo8
4 cp arq1 pasta1
5 rm pasta1/arq1
6 mpirun --hostfile hosts -np 16 mestre 200 >> lixo16
7 cp arq1 pasta1
8 rm pasta1/arq1
9 mpirun --hostfile hosts -np 32 mestre 200 >> lixo32
10 cp arq1 pasta1
11 rm pasta1/arq1
12 mpirun --hostfile hosts -np 8 mestre 200 >> lixo8
13 cp arq1 pasta1
14 rm pasta1/arq1
15 mpirun --hostfile hosts -np 16 mestre 200 >> lixo16
16 cp arq1 pasta1
17 rm pasta1/arq1
18 mpirun --hostfile hosts -np 32 mestre 200 >> lixo32
19 cp arq1 pasta1
20 rm pasta1/arq1
```

Figura 4: Script das linhas de execução do modelo CropSim-Wheat.

Na Figura 4, 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.

4.2 Resultados

A primeira coluna das tabelas 1 e 2, apresenta o número de processos, a segunda coluna é a quantidade de linhas de execução que está sendo enviada para cada processo. A terceira coluna contém os valores em segundos do tempo gasto na execução do modelo com dados distribuídos, a quarta 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.

Na Tabela 1 são apresentados os valores em segundos do tempo necessário para a execução paralela do modelo CropSim-Wheat em uma única máquina utilizando de 2 e 4 processos.

Tabela 1: Execução paralela do modelo CropSim-Wheat utilizando uma máquina com 4 núcleos físicos

Processos	Linhas	Dados Distribuídos	Dados Centralizados
2	50	5315,42	4326,14
2	100	5257,18	4469,89
2	200	5209,04	4536,01
2	400	4931,08	4931,77
2	800	4841,73	5015,64
2	1000	4919,14	5990,55
4	50	3403,03	2938,72
4	100	3428,60	3453,70
4	200	3431,12	3281,42
4	400	3454,07	2933,44
4	800	3417,33	3653,56
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 distribuídos), 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, pois está despendendo tempo de leitura e escrita de dados.

Na Tabela 2 são apresentados os valores em segundos do tempo necessário para a execução paralela do modelo CropSim-Wheat utilizando de 2 até 16 máquinas, fazendo uso de 8 até 64 processos.

Tabela 2: Execução paralela do modelo CropSim-Wheat utilizando máquinas com 4 núcleos físicos

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

Como pode ser observado na Tabela 2, há uma

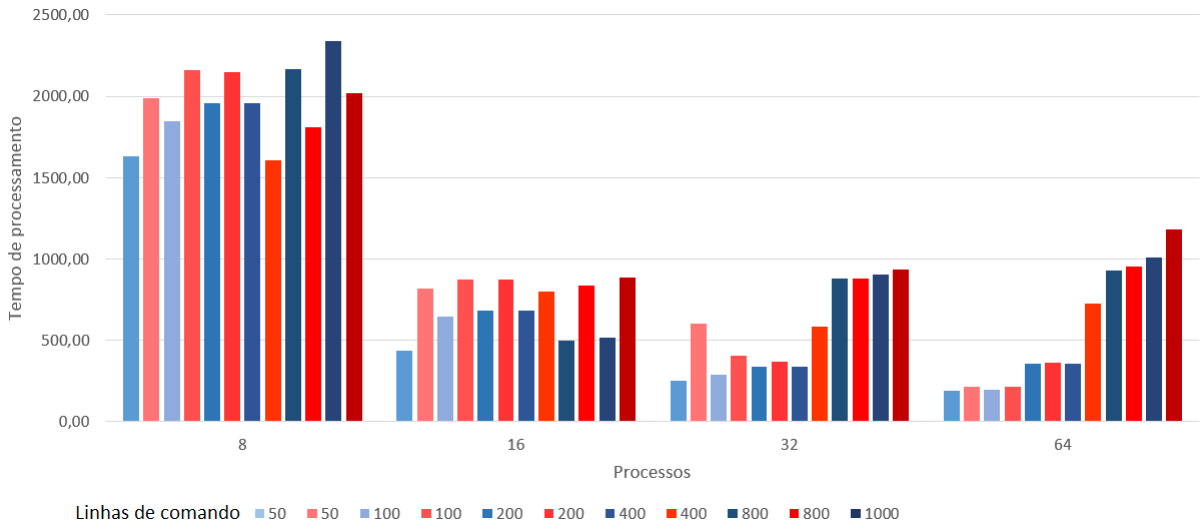

Figura 5: Comparação do desempenho obtido com os dados distribuídos e com os dados armazenados em um único local.

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 5) com dados centralizados. 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 5, é apresentado um gráfico que faz a comparação entre os valores de desempenho obtidos, com o armazenamento dos dados centralizados, em azul, e os dados armazenados distribuídos, em vermelho, mostrando o tempo de processamento enviando para cada processo quantidades diferentes de linhas de comando de execução.

Comparando os resultados das execuções de quando os dados foram centralizados em relação aos resultados 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 distribuídos houve uma melhora no desempenho, mas não na mesma proporção de quando os dados foram centralizados. Por exemplo, nos casos em que foram utilizados 8, 16 e 32 processos (Figura 5) obteve-se melhor resultado enviando 400 linhas de execuções para cada processo. Isto ocorreu, possivelmente, devido a redução de comunicação entre os processos e, também, porque os processos 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, ocasionando, consequentemente, uma perda de desempenho.

Após analisar qual o melhor caso, levando em consideração o desempenho obtido, observando o número das linhas e onde os dados foram armazenados, foi elaborada a Tabela 3, que visou a apresentação dos valores para quando os dados foram centralizados. Nesta tabela foram reunidos apenas os valores referentes as 50 linhas enviadas para cada processo. De um total de seis diferentes quantidades de processos, em quatro casos obteve-se melhor desempenho.

Levando em consideração o melhor resultado para o número de linhas de execução, conforme explicitado na Tabela 3, obteve-se um melhor desempenho quando foram utilizados 64 processos. Obteve-se um *speedup*(razão entre o tempo para executar o algoritmo sequencialmente e o tempo executando o algoritmo paralelo) de 13,85, indicando em quantas vezes foi acelerada a versão sequencial. Esta versão sequencial foi executada em, aproximadamente, 2592 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 6).

Na Tabela 4 são demonstrados os valores dos tempos de quando os dados foram centralizados em um único local.

Quando os dados estavam distribuídos, 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 nas observações anteriores, pode se verificar a influência de fatores referentes ao tamanho das mensagens e à quantidade de arquivos lidos para a execução do modelo e escritos após o processamento dos dados. Observando os valores obtidos de *speedup* e de eficiência, o melhor desempenho foi obtido com 64 processos. O desempenho computacional melhorou aproximadamente 90% comprovando que a paralelização do modelo de

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

Tabela 4: CropSim-Wheat: 400 linhas de execução e dados distribuídos

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

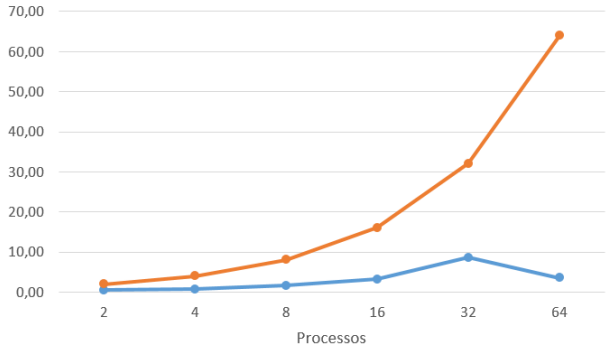

Figura 6: Gráfico de *speedup* dos dados distribuídos do CropSim-Wheat.

simulação CSM-Cropsim: Wheat mostrou ser uma opção viável para agilizar o processamento dos dados e aumentar a área de abrangência geográfica simulada por este modelo. Trabalhos futuros poderão investigar a possibilidade de melhora no desempenho que ainda pode ser realizado com outras abordagens computacionais e também com o aumento do número de processadores. Porém, sabe-se que, em alguns casos, uma quantidade maior de processadores poderia acarretar no aumento do tempo de processamento por motivos como tempo de troca de mensagens ou de leitura e escrita.

5 Conclusões

O modelo de simulação CSM-Cropsim: Wheat, utilizado pela Universidade de Passo Fundo e Embrapa Trigo, simula o crescimento do trigo e auxilia profissionais responsáveis pela tomada de decisão no campo com o intuito de melhorar a produtividade da cultura em questão. Observou-se que havia a necessidade da utilização de maiores quantidade de dados e, com isso, aumentar a área de abrangência geográfica e/ou também de dados históricos processados pelo modelo. Devido a estes fatores iniciou-se o estudo do funcionamento e da execução sequencial do modelo. Posteriormente, avaliou-se possíveis técnicas computacionais que proporcionassem o processamento de maiores quantidades de dados de forma efetiva e em tempo computacional hábil que favorecesse o uso destes dados dentro de um prazo útil. 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 a arquitetura computacionais composta de um sistema constituído de 16 máquinas do Laboratório Central de Informática da UPF.

Aplicando esta abordagem, obteve-se resultados satisfatórios na paralelização da execução do modelo, pois, com 64 processos, 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. Com isso, conclui-se que o desempenho computacional da execução paralela da execução de modelos de simulação depende de alguns fatores principais. Entre eles estão as quantidades de cálculos e de leitura/escrita (I/O) realizados pelo modelo. 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, obteve-se melhora no desempenho da execução do modelo.

Por fim, com este trabalho, otimizou-se 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.

Referências

Bryan, B. A., King, D. and Zhao, G. (2014). Influence of management and environment on australian wheat: information for

sustainable intensification and closing yield gaps, *Environmental Research Letters* **9**: 1–12. http://dx.doi.org/10.1088/1748-9326/9/4/044005.

Chou, S. C., Souza, C. R. d., Gomes, J. L., Evangelista, E. F., Osório, C. and Cataldi, M. (2007). Refinamento estatástico das previsões horárias de temperatura a 2m do modelo Eta em estações do nordeste do Brasil, *Revista Brasileira de Meteorologia* **22**: 287 – 296. http://dx.doi.org/10.1590/S0102-77862007000300001.

Del Ponte, E. M., Fernandes, J. M. C., Pavan, W. and Baethgen, W. E. (2009). A model-based assessment of the impacts of climate variability on fusarium head blight seasonal risk in southern brazil, *Journal of Phytopathology* **157**(11-12): 675–681. https://doi.org/10.1111/j.1439-0434.2009.01559.x.

Eugster, M., Knaus, J., Porzelius, C., Schmidberger, M. and Vicedo, E. (2011). Hands-on tutorial for parallel computing with R., *Computational Statistics* **26**(2): 219–239. http://dx.doi.org/10.1007/s00180-010-0206-4.

Hadka, D. and Reed, P. (2015). Large-scale parallelization of the Borg multiobjective evolutionary algorithm to enhance the management of complex environmental systems, *Environmental Modelling & Software* **69**: 353–369. https://doi.org/10.1016/j.envsoft.2014.10.014.

Hunt, L. A. and Pararajasingham, S. (1995). Cropsim-wheat: A model describing the growth and development of wheat., *Canadianjournal Plant Science* . http://pubs.aic.ca/doi/pdf/10.4141/cjps95-107.

Janssen, S. J., Porter, C. H., Moore, A. D., Athanasiadis, I. N., Foster, I., Jones, J. W. and Antle, J. M. (2017). Towards a new generation of agricultural system data, models and knowledge products: Information and communication technology, *Agricultural Systems* **155**: 200 – 212. https://doi.org/10.1016/j.agsy.2016.09.017.

Jiang, J., Wang, T., Chi, X., Hao, H., Wang, Y., Chen, Y. and Zhang, H. (2016). Sc-esap: A parallel application platform for earth system model1, *Procedia Computer Science* **80**: 1612–1623. https://doi.org/10.1016/j.procs.2016.05.493.

Jordi, A. and Wang, D. P. (2012). SbPOM: A parallel implementation of Princenton Ocean Model, *Environmental Modelling and Software* **38**: 59–61. http://dx.doi.org/10.1016/j.envsoft.2012.05.013.

Lazzaretti, A. T. (2013). *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.*, PhD thesis, UPF - Faculdade de Agronomia e Medicina Veterinária, Programa de Pós Graduação em agronomia. Disponível em www.ppgagro.upf.br/images/stories/alexandre-tagliari-lazzareti.pdf (Accesso 20 Agosto 2015).

Liu, Y. e. a. (2012). Optimization of agricultural bmps using a parallel computing based multi-objective optimization algorithm, *The International Journal of Environmental Resources Research* **1**(1): 39–50. http://dx.doi.org/10.22069/ijerr.2013.1685.

Luo, C., Mohsenimanesh, A. and Laguë, C. (2018). Parallel point-to-point tracking for agricultural wide-span implement carrier (wsic), *Computers and Electronics in Agriculture* **153**: 302 – 312. https://doi.org/10.1016/j.compag.2018.08.030.

Pavan, W. (2007). *Técnicas de Engenharia de Software Aplicadas à Modelagem e Simulação de Doenças de Plantas.*, PhD thesis, UPF - Faculdade de Agronomia e Medicina Veterinária, Programa de Pós Graduação em agronomia. Disponível em http://www.ppgagro.upf.br/download/Willingthon_Pavan.pdf (Accesso 15 Agosto 2016).

Prabhu, B. V. B. and Dakshayini, M. L. (2018). Performance analysis of the regression and time series predictive models using parallel implementation for agricultural data, *Procedia Computer Science* **132**: 198 – 207. https://doi.org/10.1016/j.procs.2018.05.187.

Ross, J. A., Richie, D. A., Park, S. J. and Shires, D. R. (2016). Parallel programming model for the epiphany many-core coprocessor using threaded mpi, *Microprocessors and Microsystems* **43**: 95–103. https://doi.org/10.1016/j.micpro.2016.02.006.

Schmidberger, M., Morgan, M., Eddelbuettel, D., Yu, H., Tierney, L. and Mansmann, U. (2009). State of the art in parallel computing with R., *Journal of Statistical Software.* **31**: 1–27. http://dx.doi.org/10.18637/jss.v031.i01.

Tiejian, L., Guangqian, W., Ji, C. and Hao, W. (2011). Dynamic parallelization of hydrological model simulations, *Environmental Modelling and Software* **26**(12): 1736–1746. http://dx.doi.org/10.1016/j.envsoft.2011.07.015.

van den Oord, G. and Bakhshi, R. (2017). Parallel post-processing of the earth climate model output, *Procedia Computer Science* **108**: 2473–2477. https://doi.org/10.1016/j.procs.2017.05.146.

Yalew, S., van Griensven, A., Ray, N., Kokoszkiewicz, L. and Betrie, G. (2013). Distributed computation of large scale {SWAT} models on the grid, *Environmental Modelling & Software* **41**(0): 223–230. http://dx.doi.org/10.1016/j.envsoft.2012.08.002.

Zhang, X., Beeson, P., Link, R., Manowitz, D., Izaurralde, R. C., Sadeghi, A., Thomson, A. M., Sahajpal, R., Srinivasan, R. and Arnold, J. G. (2013). Efficient multi-objective calibration of a computationally intensive hydrologic model with parallel computing software in Python, *Environmental Modelling and Software* **46**: 208–218. http://dx.doi.org/10.1016/j.envsoft.2013.03.013.

Zhao, G., Bryan, B. A., King, D., Luo, Z., Wang, E., Bende-Michl, U., Song, X. and Yu, Q. (2013). Large-scale, high-resolution agricultural systems modeling using a hybrid approach combining grid computing and parallel processing, *Environmental Modelling & Software* **41**(0): 231–238. http://dx.doi.org/10.1016/j.envsoft.2012.08.007.