UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE DARCY RIBEIRO

CENTRO DE CIÊNCIAS E TECNOLOGIAS AGROPECUÁRIAS

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA ANIMAL

KAMILA DA SILVA ALVARENGA

CONSTRUÇÃO DE UM CLUSTER PARA COMPUTAÇÃO PARALELA COM R APLICADO À ZOOTECNIA

Campos dos Goytacazes - RJ

Fevereiro/2020

KAMILA DA SILVA ALVARENGA

CONSTRUÇÃO DE UM CLUSTER PARA COMPUTAÇÃO PARALELA COM R APLICADO À ZOOTECNIA

Dissertação apresentada ao Centro de Ciências e Tecnologias Agropecuárias da Universidade Estadual do Norte Fluminense Darcy Ribeiro, como requisito parcial para obtenção do grau de Mestre em Produção Animal, na Área de Concentração de Produção Animal

ORIENTADOR: Prof. Leonardo Siqueira Glória

Campos dos Goytacazes

2020

KAMILA DA SILVA ALVARENGA

CONSTRUÇÃO DE UM CLUSTER PARA COMPUTAÇÃO PARALELA COM R APLICADO À ZOOTECNIA

Dissertação apresentada ao Centro de Ciências e Tecnologias Agropecuárias da Universidade Estadual do Norte Fluminense Darcy Ribeiro, como requisito parcial para obtenção do grau de Mestre em Produção Animal, na Área de Concentração de Produção e Reprodução Animal

Aprovada em 19 de fevereiro de 2020

BANCA EXAMINADORA

Dr. Jonas Henrique de Souza Motta (Doutor, Ciência Animal) – UENF

Dr. Matheus Lima Corrêa Abreu (Doutor, Ciência Animal) – UFMT

Prof Alberto Magno Fernandes (Doutor, Zootecnia) – UENF

Prof. Leonardo Siqueira Glória (Doutor, Genética e Melhoramento) – UENF (Orientador)

AGRADECIMENTOS

Primeiramente gostaria de agradecer a Deus, por ter me proporcionando mais uma oportunidade de agregar conhecimento em minha vida, agradecer à minha esposa Lucy Salvini por todo apoio, por acreditar em mim e por suportar minhas crises de estresse (risos). Agradeço ao meu orientador Leonardo Siqueira Glória por todo o auxílio e dedicação. Agradeço também a nossa equipe do NAD, Antônio vulgo “Tunico”, Beijiane, Bruna, Caio, Flávia, Gabriel e Jacyara. Galera vocês são top! Obrigada por me ajudar tantas vezes, por nossas risadas e lanchinhos no fim da tarde! Que todos possam realizar seus sonhos, seguir seus caminhos felizes e fazendo o que gostam!

“Procure ser um homem de valor, em vez de ser um homem de sucesso”

Albert Einstein

RESUMO

Tendo em vista o avanço da tecnologia hodiernamente, é de se esperar que existem inúmeros dados a serem processados, bem como, o processamento dos mesmos de forma mais rápida. Para isso, precisa-se cada vez mais de supercomputadores. A análise de uma grande quantidade de dados em tempo computacional hábil é uma necessidade atual de inúmeras organizações. Por este motivo, a utilização de ferramentas computacionais torna-se indispensável para a realização destas atividades. A utilização da linguagem R em conjunto com a computação de alta performance são ferramentas utilizadas para suprir estas necessidades. O R é uma linguagem de programação *open source* e um ambiente para computação estatística, modelação e visualização de dados. Tange-se de uma linguagem de programação totalmente desenvolvida para análise de dados. Ademais, está disponível para diversos sistemas operacionais, como *Linux, Unix, Windows e MacOS*. Portanto, o objetivo deste artigo foi descrever um tutorial explicando como montar um *cluster* de análise de dados para paralelização e sua aplicação nos métodos estatísticos Bayesianos aplicados em produção animal, com uso de sistema operacional e *software* livre. Foram utilizados 5 (cinco) *desktops* que vão desde *Cluster00* até *Cluster04* (cada um com 4GB RAM, 500 GB de HD e processador Core i5 segunda geração). O *Cluster00* sendo o mestre e os demais são os "nós escravos"). Foi usado o Linux – Ubuntu 18.04 *Desktop* no "nó mestre" e Ubuntu Server 18.04 LTS nos "nós escravos", ambos de 64 bits. Em todos os computadores utilizados foram instalados o *software* R, bem como os pacotes "snow" e "MCMCglmm". Nesse trabalho o tempo de análise com uso da cadeia dividida nos diferentes "nós" chega a reduzir em 80%. Enquanto a análise em apenas um computador é de aproximadamente 61 segundos, a de 5 computadores é de aproximadamente 12 segundos. A utilização de um *cluster* de análise de dados reduz exponencialmente o tempo de análise. A metodologia Bayesiana é otimizada com uso de um *cluster* de análise.

Palavras-chave: Alta performance, Bayesiana e *Cluster*

Abstract

In view of the advancement of current technologies, it is possible to expect that there are a greater number of data processed, as these are processed faster. For this reason, more and more supercomputers are needed. An analysis of a large amount of data, in appropriate computational time, nowadays, is demand for any enterprise. Therefore, the use of computational tools becomes indispensable for carrying out many activities. The use of the R language together with high performance is tool used to meet many businesses. In view of that, my objective was to describe a tutorial explaining how to set up a data analysis cluster for parallelization and its application in Bayesian statistical methods that use livestock production, using the operating system and free software. The R *software* is an open source programming language and an environment for statistical computing, modeling and data responses. It is a fully developed programming language for data analysis. In addition, it is available for several operating systems, such as Linux, Unix, Windows and MacOS. Five desktops computers were used, ranging from Cluster00 to Cluster04 (each with 4 GB of RAM, 500 GB of HD and Core i5 processor. Cluster00 is the master and the rest “slaves”). It was used on Linux - Ubuntu 18.04 Desktop on the "master node" and on Ubuntu Server 18.04 LTS on the "slaves", both 64-bit. On all the computers used, the R software and the “snow” and “MCMCglmm” packages were installed. In this study, the analysis time with the use of the chain divided into different “nodes” decrease approximately 80% i.e., in analysis with only one computer it is approximately 61 seconds and while with 5 computers it is approximately 12 seconds. The use of a data analysis cluster reduces the analysis time exponentially. Furthermore, the Bayesian methodology is optimized with the use of an analysis cluster.

Keywords: High performance, Bayesian and Cluster.

Sumário

1. INTRODUÇÃO

Tendo em vista o avanço da tecnologia hodiernamente, é de se esperar que existem inúmeros dados a serem processados, bem como, o processamento dos mesmos de forma mais rápida. Para isso, precisa-se cada vez mais de supercomputadores.

Os supercomputadores foram inicialmente a primeira tecnologia a desempenhar o alto poder de computação, contudo são extremamente onerosos e pouco escalonáveis. Para substituir os convencionais supercomputadores surgiu o *Cluster Beowulf*, que fornece computação de alto desempenho (HPC) a baixo custo, uma vez que utiliza *commodities* como componentes (STERLING et al. 1995).

A análise de uma grande quantidade de dados em tempo computacional hábil é uma necessidade atual de inúmeras organizações. Por este motivo, a utilização de ferramentas computacionais torna-se indispensável para a realização destas atividades.

A utilização da linguagem R em conjunto com a computação de alta performance são ferramentas utilizadas para suprir estas necessidades. O R é uma linguagem de programação *open source* e um ambiente para computação estatística, modelação e visualização de dados. Tange-se de uma linguagem de programação totalmente desenvolvida para análise de dados. Ademais, está disponível para diversos sistemas operacionais, como: *Linux, Unix, Windows e MacOS* (TORGO, 2009).

Entre as ferramentas da linguagem R está o paralelismo. A computação paralela tem desempenhado atualmente grande importância para análise de dados em várias áreas. O propósito principal do paralelismo é diminuir o tempo de processamento computacional através de ferramentas e recursos computacionais específicos. Existem pacotes em linguagem R específicos para esse tipo de aplicação, tais como: Rmpi, snow, snowFT, snowfall, foreach, doMC, doSNOW, doMPI e Rdsm (VERA e SUPPI, 2010).

O paralelismo tem sido aplicado nas áreas de melhoramento animal e vegetal com o intuito de ganhar maior desempenho em menor tempo de execução na análise dos dados, sendo inclusive eficiente à seleção genômica. (LAGROTTA et al., 2017).

2. OBJETIVO

O objetivo deste trabalho é apresentar um tutorial explicando a paralelização e sua aplicação nos métodos estatísticos Bayesianos aplicados em produção animal. O protótipo desta pesquisa é composto por 16 computadores totalizando 64 núcleos e 68 GB de memória RAM na distribuição Linux Ubuntu, e será disponibilizado para os pesquisadores utilizarem o *cluster* para análises estatísticas com o software R.

3. REVISÃO DE LITERATURA

3.1 SERVIDORES

Os servidores cumprem diversas tarefas em uma rede de computadores, entre elas, prover diferentes serviços aos clientes que acessam estes servidores, além de executar os serviços de: servidor de arquivos, aplicações, impressão, *e-mail, backup,* acesso remoto, dentre outros. Para que um servidor tenha um bom funcionamento para trabalhar com um grande número de requisições, é necessário que o mesmo disponha de *hardwares* específicos para esta finalidade (MENDES, 2007).

Os sistemas operacionais de rede necessitam de uma máquina *host* para trabalhar. Ainda que parte do sistema resida em cada computador-cliente, a maior parte trabalha em um servidor de rede. O servidor de rede, mais especificamente, é o computador que armazena recurso de *software*, tais como o próprio sistema operacional de rede, aplicações de computador, programas, conjuntos de dados e bancos de dados, além de permitir autorizações às estações conectas à rede para acessarem todos os recursos. Normalmente, os servidores de rede são constituídos por pequenos computadores a *mainframes* (computador de grande porte dedicado normalmente ao processamento de um volume enorme de informações). Comumente, o servidor é uma estação de microcomputador potente com componentes redundantes (WHITE, 2012).

Conforme explicam Lemos et al. (1999), o módulo servidor é uma entidade que torna disponível os recursos de uma estação aos usuários da rede. Entre os

recursos mais oferecidos, pode citar: armazenamento de arquivos, gerência de bancos de dados, suporte para impressão, tradução de nomes simbólicos em endereços físicos, concentração de terminais, monitoramento de redes, criptografia, correio eletrônico e serviços de comunicação.

3.2 CLUSTER

Segundo Pitanga (2008), a computação em *cluster* nasceu em 1962 com o projeto SAGE (*Semi – Automatic Ground Environment*), onde a IBM construiu um *cluster* para o NORAD (*North American Air Defense*). Esse projeto compreende diversos sistemas separados trabalhando cooperativamente para monitorar invasões aéreas no continente norte-americano. Na década de 80 o interesse pela computação em *cluste*r cresceu através de experimentos em pesquisas na indústria. Para a Agência de Segurança Nacional Americana foi criada uma coleção com 160 estações de trabalho denominada “Apollo” interconectadas como *cluster* para realizar tarefas computacionais complexas.

Pode perceber que atualmente há um grande número de aplicações que exigem cada vez mais uma grande quantidade de processamento de dados, computação gráfica, aplicações de mapeamento genético, previsões meteorológicas inclusive programas que exigem um amplo número de variáveis de entrada. Os supercomputadores são sistemas fortemente acoplados e foram criados para solucionar a demanda por poder computacional. Por possuir um custo elevado, os supercomputadores não são amplamente utilizados, tendo como alternativa mais acessível o uso de *cluster*, também conhecido como sistema fracamente acoplado. Os *clusters* de computadores processam as tarefas paralelas de forma transparente, ou seja, aparentando ser um único sistema para o usuário (BACELLAR, 2010).

Segundo Kindel et al. (2004), existem basicamente três tipos de *clusters*:

- Alta disponibilidade *(HA - High Availability)*: tem como função manter um determinado sistema rodando permanentemente;
- Balanceamento de carga (*HS – Horizontal Scaling*): não há processamento paralelo, mas sim distribuição de processos individuais entre os componentes

do sistema, portanto, se trata de um sistema onde a disponibilidade de um determinado serviço é grande;

- Computação de alto desempenho (*HPC-High Performance Computing*): o objetivo principal é atingir um alto desempenho para um determinado tipo de aplicação especialmente desenvolvido para processamento paralelo.

3.3 CLUSTER BEOWULF

Dentre os clusters de alto desempenho existe o *Cluster Beowulf.* Este tipo de cluster foi desenvolvido em 1994, pelos pesquisadores Donald Becker e Thomas Sterling do Centro de Voos Espaciais Goddard da NASA (*Goddard Space Flight Center).* Naquela época havia a necessidade de aumentar a capacidade de processamento para as pesquisas do *Goddard Space Flight Center*, entretanto os recursos para financiar os projetos estavam acabando (*STERLING*, 2002).

Foi então que esses pesquisadores decidiram interligar 16 computadores pessoais, cada um contendo um microprocessador 486, usando o sistema operacional com *kernel* Linux e uma rede padrão *Ethernet.* Todo esse conjunto conseguiu atingir a marca de 70 *megaflops*, o que para aquela época não era uma velocidade baixa comparada aos supercomputadores comerciais disponíveis (PITANGA, 2008).

O *Cluster Beowulf* consiste em um sistema de computação paralelo composto de vários computadores autônomos denominados “nós”, cada um executando seu próprio sistema operacional e que se comunicam com os demais através de uma rede de interconexão.

Dentre os “nós”, um deles é considerado mestre ou front-end, sua função é gerenciar os outros “nós” (escravos) e distribuir o processamento entre eles, entretanto, o nó mestre funciona também como *gateway* (*STERLING*, 2002).

Segundo Bacellar (2010), o sistema não requer arquitetura específica e máquinas homogêneas, apenas deve satisfazer às premissas abaixo:

- Conexão entre os nós, que pode ser feita por meio de *ethernet*;
- Deve haver um ou mais nós mestres (*front-end*) para realizar o controle dos nós escravos (*back-end*);

- O sistema operacional deve ser baseado em código aberto, bem como o mesmo deve conter todas as ferramentas necessárias para a configuração do *cluster*;
- É importante que haja um “nó mestre” (servidor) que realiza toda a distribuição das tarefas e o monitoramento do desempenho do cluster.

O *front-end* é responsável pelo monitoramento das falhas que eventualmente podem ocorrer assim como o direcionamento da carga de processamento, caso haja alguma indisponibilidade. Segue um esquema na figura 1 de como é a arquitetura de um *Cluster Beowulf.*

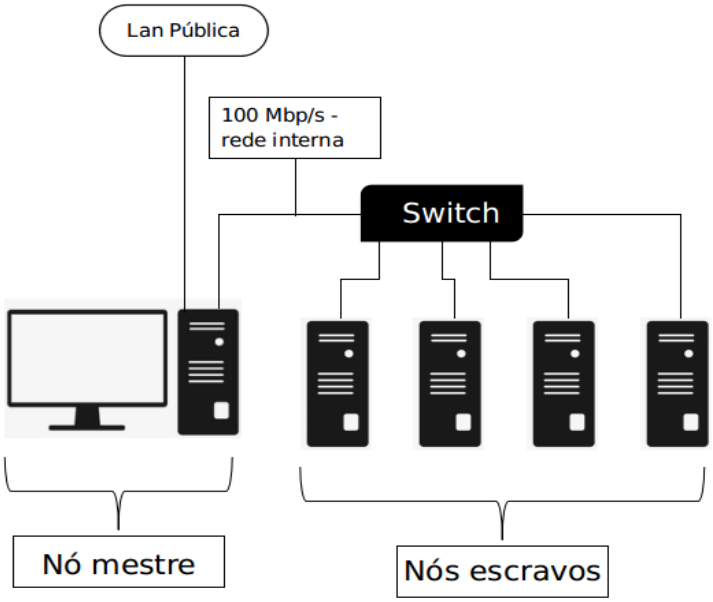


Figura 1: Esquema de um C*luster Beowulf*. Adaptado do site nicepng.

Conforme descreve Brown (2006), os campos de aplicações de um *Cluster Beowulf* são: meteorologia, astrofísica, simulações de mercado financeiro, exploração de petróleo, otimização, biotecnologia, tolerância a falhas, inteligência artificial, servidores de *internet*, pesquisa operacional, tratamento de imagens, jogos e até renderização de efeitos visuais para o cinema.

3.4 COMPUTAÇÃO PARALELA EM R

A computação paralela de maneira geral, lida com hardware e software para computação para que muitos cálculos sejam realizados simultaneamente. O principal

objetivo da computação paralela é a melhoria na capacidade do cálculo (SHMIDBERGER,2008).

R é uma linguagem de script para manipulação e análise de dados estatísticos. Essa linguagem foi inspirada e é compatível principalmente com a linguagem estatística denominada S, a qual foi desenvolvida pela AT & T. O nome S vem da palavra *statistics* em inglês, ela faz uma alusão a outra linguagem de programação com um nome de uma letra desenvolvido na AT & T, a famosa linguagem C. A linguagem S mais tarde foi vendida para uma empresa pequena, que adicionou uma interface gráfica de usuário (GUI) e nomeou o resultado de S-Plus. R por sua vez, se tornou mais popular que S ou S-Plus, porque trata-se de uma linguagem gratuita onde há uma contribuição de diversos programadores pelo mundo. Por vezes a linguagem R é chamada de GNU S, para refletir sua natureza de código aberto. (O Projeto GNU é uma grande coleção de software livre) (NORMAN MATLOFF, 2011, p.21).

Leanch (2014) diz que ao trabalhar com R, frequentemente encontrarão situações em que será necessário repetir um cálculo por diversas vezes, e para que isso seja feito é necessário o uso do loop for do R. No entanto, se houver um grande número de cálculos a ser realizado (ou seja, muitos milhares), ou se esses cálculos individuais são demorados (ou seja, muitos minutos), o loop for pode ser muito lento. Dito isso, quase todos os computadores agora têm processadores multicore, e enquanto esses cálculos não precisam comunicar (ou seja, eles são “embaraçosamente paralelos”), eles podem ser divididos por múltiplos núcleos e executados em paralelo, reduzindo o tempo de computação. Exemplos desses tipos de problemas incluem:

- Execução de modelo de simulação usando vários conjuntos de parâmetros diferentes;
- Execução de várias cadeias MCMC simultaneamente;
- Bootstrapping, validação cruzada, etc.

O R não é apenas um ambiente para computação estatística e gráficos, mas também uma linguagem de alto nível disponível gratuitamente para programação com um grande número de bibliotecas. Assim, desenvolvedores e estatísticos em

todo o mundo podem participar e fornecer código opcional para o ambiente base R. Desenvolver e implementar novos métodos de análise de dados pode, portanto, ser bastante fácil (HORNIK, 2007; R DEVELOPMENT CORE TEAM, 2012).

3.5 PACOTE *SNOW*

Rossini et al (2007) dizem que o pacote *snow* (Rede Simples de Estações de Trabalho) suporta computação paralela simples em R., que tem a interface simples e projetada para suportar vários mecanismos diferentes de comunicação de baixo nível. Quatro interfaces de baixo nível foram implementadas: PVM (via pacote rpvm), MPI (via Rmpi), NetWorkSpaces (via nws) e soquetes brutos que podem ser úteis se PVM, MPI ou NWS não estiverem disponíveis. Isso significa que é possível executar o mesmo código em um cluster com PVM, MPI ou NWS ou em um único computador com vários núcleos. O pacote snow inclui scripts para iniciar instâncias R nos escravos (cl <- makeCluster ()). As instâncias são executadas até serem fechadas explicitamente por meio do comando stopCluster (cl).

3.6 COMPUTAÇÃO PARALELA NA ESTATÍSTICA

A crescente busca por plataformas de processamento paralelo de dados extremamente escaláveis tem tido grande crescimento relacionado ao número de aplicações de grande volume de dados. É um grande desafio para os *softwares* de estatística e para a infraestrutura computacional atual executar computação estatística sobre os grandes bancos de dados existentes (XIE et al, 2012)

Kumar et al (2011) mostram que o exponencial crescimento na geração e coleta de dados desencadeou a inovação das técnicas de análise de dados e da extração de informações. Os processadores convencionais não possuem capacidade e velocidade computacional mediante os requisitos das massivas técnicas de mineração de dados. Processadores de alto desempenho têm grande potencial para trabalhar com grandes cargas de trabalho computacional. Para atender essa demanda a estrutura deve conter *software* e uma biblioteca de alto desempenho.

Stokely et. al. (2011) esclarecem a importância da computação paralela para estatística computacional utilizando o exemplo do *Map Reduce* para R. Tal

ferramenta possibilita que os cálculos sejam escritos em linguagem de alto nível e posteriormente divididos e distribuídos para que as tarefas sejam executadas por *datacenters* do Google, e.g. *Datacenters* são ambientes projetados para abrigar servidores e outros componentes tais como sistemas de armazenamento de dados (*storages*) e ativos de rede (*switches*, roteadores).

A computação paralela é atualmente imprescindível para a análise de dados em várias disciplinas. Com o propósito de diminuir o tempo de processamento computacional, ferramentas computacionais e recursos são vitais. Como solução para execução de *loops* paralelos em uma rede de computadores foram desenvolvidas soluções em linguagem R. Os pacotes existentes podem-se citar Rmpi, snow, snowFT, snowfall, foreach, doMC, doSNOW, doMPI e Rdsm (VERA e SUPPI, 2010).

Segundo Mazzonetto (2016), que realizou em seu trabalho estudos de caso com técnicas para computação paralela para realizar o processamento de grandes quantidades de dados de forma mais efetiva e em menor tempo, técnicas aplicadas em modelos de simulação utilizados pela Universidade de Passo Fundo e Embrapa Trigo, modelo conhecido como *CSM-Cropsim: Wheat.* Esse modelo simula o crescimento do trigo ajudando profissionais da área a tomar decisões com a finalidade de melhorar a produtividade.

Ainda nos estudos realizados por Mazzonetto (2016), observou-se que na execução do *CSM-Cropsim: Wheat,* os resultados obtidos utilizando a paralelização em sua execução foram extremamente satisfatórios. Com 64 processadores obteve-se um tempo de 187,13 segundos de *speedup de 13,85, enquanto que o tempo* sequencial para realizar a mesma quantidade de dados foi de 2593 segundos.

Em pesquisas de melhoramento genético a seleção genômica tem ganhado um crescente espaço. A utilização de ferramentas moleculares viabiliza a varredura uniforme do genoma para diversos marcadores da classe SNP. (*single nucleotide polymorphisms*). Isso torna possível a obtenção de mapas genômicos de grande interesse econômico (HABIER et al., 2011).

Entretanto, para execução dessas ferramentas existem muitos obstáculos na aplicação metodológica e computacional. (WU et al., 2011). Para superar esses obstáculos, métodos estatísticos sofisticados têm sido utilizados. Os modelos mais

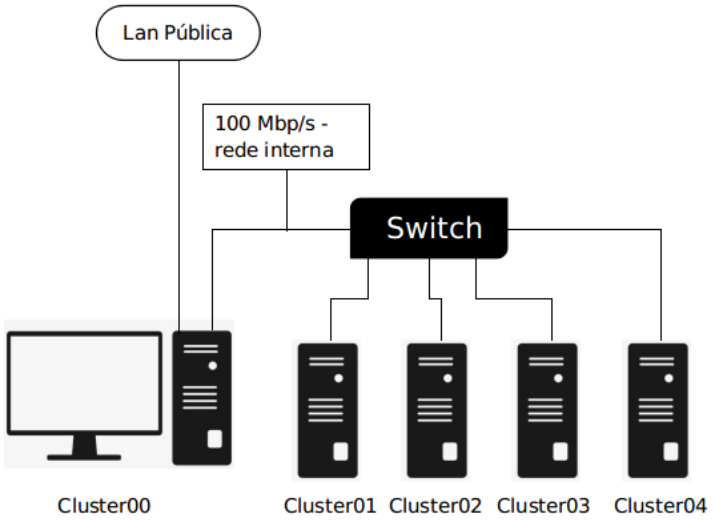
utilizados são baseados na Inferência Bayesiana através dos métodos de Monte Carlo Via Cadeias de Markov (MCMC).

Segundo Lagrotta et al (2017), esses métodos são complexos e computacionalmente intensivos e precisam de alta demanda computacional. Algumas análises na área de melhoramento animal e vegetal chegam a durar semanas ou meses em computadores de alto desempenho. Tendo em vista toda a grande demanda computacional, realizaram estudos comparando a eficiência de processamento do método BayesCπ programado em paralelo com o seu algoritmo sequencial padrão. Foram estudados dois métodos de paralelização. A primeira foi a análise de múltiplas cadeias MCMC em paralelo, e a segunda foi a paralelização de uma única cadeia MCMC. Foi utilizada a biblioteca MPI e o pacote OpenMPI associado ao compilador gfortran para execução em paralelo desses algoritmos. O algoritmo sequencial padrão foi processado em 77,29 horas. Na utilização de múltiplas cadeias em paralelo o processamento foi 77% mais rápido (17,75hs), enquanto que o método de paralelização de uma única cadeia apresentou um ganho de desempenho de 15% (65,37hs). Conclui-se então que a computação paralela é eficiente e pode ser aplicada à seleção genômica.

4. MATERIAL E MÉTODOS

4.1 HARDWARE UTILIZADO

Foram utilizados cinco computadores *desktops* que foram de *Cluster00* até *Cluster04* (cada um com 4GB RAM, 500 GB de HD e processador Core i5. O *Cluster00* sendo o mestre e os demais são os “nós escravos”). Um *Switch* TP-Link 8 portas foi utilizado para interligar os cinco nós do *cluster* na mesma rede. Além disso, um monitor Dell® que serviu para visualizar a interface de cada *desktop*, o qual permitiu que a interface do “nó mestre” fosse visualizada. Por fim, o sistema operacional utilizado no *cluster* foi o Linux – Ubuntu 18.04 *Desktop* no “nó mestre” e Ubuntu Server 18.04 LTS nos “nós escravos”, ambos de 64 bits. Uma ilustração do sistema em *clusters*, com a disposição dos cinco *hardwares* construídos, está apresentada na Figura 2.



.

Figura 2: Esquema de um C*luster Beowulf*. Adaptado do site nicepng.

4.2 CRIAÇÃO DE DRIVES USB INICIALIZÁVEIS

Nesta etapa foram criados os drives USB inicializáveis para instalação dos sistemas operacionais. Antes foi necessário realizar o *download* do Ubuntu 18.04.3 LTS (*Desktop*) e do Ubuntu Server 18.04.3 LTS. A seguir, a ordem de execução desta etapa:

Etapa 1: Foi realizado o download do Sistema Operacional Ubuntu 18.04.3 LTS (Desktop). Abaixo segue o link para download.

https://ubuntu.com/download/desktop

Etapa 2: Foi realizado o download do software Ubuntu Server 18.04.3 LTS. Abaixo segue o link para download.

https://ubuntu.com/download/server

Etapa 3: Foi realizado o download do software Rufus 3,6. Abaixo segue o link para download.

https://rufus.ie/pt_BR.html

Foram separados 3 (três) *pen-drives* de 8GB para torná-los inicializáveis, um para o Ubuntu 18.04.3 LTS e dois para Ubuntu Server 18.04.3 LTS, assim a instalação nos *desktops* tornou-se mais rápida.

Etapa 4: O pen-drive foi inserido na unidade USB, executando o programa Rufus 3,6, clicamos no botão "**Selecionar uma imagem ISO**", selecionamos o Ubuntu 18.04.3 LTS, clicamos no botão "**Abrir**" e no botão "**Iniciar**". Quando tudo estava finalizado, com a barrinha verde carregada, clicamos no botão "**Fechar**" e removemos o pen-drive da unidade USB.

O procedimento da etapa 4 foi repetido para o Ubuntu Server 18.04.3 LTS com dois pen-drives.

4.3 INSTALAÇÃO DO SISTEMA OPERACIONAL UBUNTU

Depois que a instalação do Hardware foi finalizada e os pen-drives inicializáveis prontos, foi iniciada a instalação do sistema operacional Ubuntu.

Para nosso sistema foram utilizados 5 (cinco) *desktops,* que vão de *Cluster00* até *Cluster04, sendo o Cluster00* o “nó mestre” e os demais “nós escravos”. Pode-se utilizar o número de *desktops* que preferir ou possuir para alcançar o poder computacional necessário.

Abaixo estão apresentadas as etapas para instalação do nó mestre (Cluster 00):

Etapa 1: Inserimos o pen-drive em que o Ubuntu 18.04.3 LTS foi gravado na unidade USB, ligamos o *desktop*, pressionamos constantemente a tecla de função <F11> (ou <F10> ou <ESC> ou <DEL> dependendo do seu *desktop.*) para ter acesso ao *Boot Device*;

Etapa 2: Com a seta do teclado, escolhemos a opção “Install Ubuntu” e pressionamos a tecla *ENTER*;

Etapa 3: Escolhemos o idioma Poruguês(Brasil) e depois clicamos em "continuar";

Etapa 4: Selecionamos o Layout do teclado, na opção "Português (Brasil)" e depois clicamos no botão "Continuar" (existe um campo que possibilita testar o teclado);

Etapa 5: Escolhemos a opção “Não quero conectar a uma rede sem fio agora” e depois clicamos em “Continuar”;

Etapa 6: Escolhemos a opção “Instalação normal”. No campo “Outras opções” deixamos todas as opções desmarcadas e depois clicamos em “Continuar”;

Etapa 7: Na janela “Tipo de instalação” escolhemos a opção: “Apagar disco e reinstalar o Ubuntu” (Nesse caso ele apagou o sistema operacional que estava instalado no *desktop* e instalou o Ubuntu 18.04.3 LTS), depois clicamos em “Instalar agora”;

Etapa 8: O computador perguntou: “Escrever as mudanças nos discos?”, clicamos em “Continuar”;

Etapa 9: Selecionamos a localização e clicamos em “Continuar”;

Etapa 10: Configuramos o nome do “nó mestre”. Nesta opção inserimos o nome de “Cluster00”, depois uma senha e em seguida clicamos em “Continuar”;

Etapa 11: Quando toda a instalação terminou, clicamos no botão "Reiniciar agora", em seguida removemos o pen-drive unidade USB.

Nó escravo (Cluster 01 ao n)

Etapa 1: Inserimos o pen-drive em que o Ubuntu Server 18.04.3 LTS foi gravado na unidade USB, ligamos o *desktop*, pressionando constantemente a tecla de função <F11> (ou <F10> ou <ESC> ou <DEL> dependendo do seu *desktop*) para ter acesso ao *Boot Device*;

Etapa 2: Com a seta do teclado escolhemos o idioma Português (Brasil) e depois pressionamos a tecla *ENTER*;

Etapa 3: Na tela seguinte, foi escolhida a opção “Instalar o Ubuntu Server” e depois *ENTER*;

Etapa 4: Apareceu um aviso informando que a tradução do instalador para nossa língua ainda não está completa. Com as setas do teclado, selecionamos a opção “Yes” e pressionamos a tecla *ENTER*;

Etapa 5: Nesta etapa foi configurado o teclado selecionando o país de origem para o layout do teclado utilizado e depois pressionamos a tecla *ENTER*;

Etapa 6: Escolhemos um layout que correspondesse ao do seu teclado físico (Portuguese Brazil) e pressionamos a tecla *ENTER;*

Etapa 7: Informamos um nome para identificar o computador na rede, neste caso colocamos “Cluster01”. Depois a opção “Continuar” e pressionamos a tecla *ENTER*;

Etapa 8: Nesta etapa foi digitado o nome real do usuário que usaria o servidor, neste caso “Cluster 01”. Depois a opção “Continuar” e pressione a tecla *ENTER;*

Etapa 9: Na próxima tela digitamos o nome “Cluster01” que seria utilizado para fazer login no servidor;

Etapa 10: No próximo passo, foi digitada a senha de nossa preferência. Na tela seguinte, digitamos novamente a senha e depois a opção “Continuar” e pressionamos a tecla *ENTER*;

Etapa 11: Quando foi questionado se desejávamos criptografar a pasta pessoal, respondemos “Não” e clicamos em continue;

Etapa 12: Nesta tela o instalador mostrou o fuso horário de nossa localização. Verificamos se tudo estava correto, confirmamos e clicamos em continue;

Etapa 13: Nesta etapa escolhemos a opção “Assistido – usar o disco inteiro” e pressionamos a tecla *ENTER*;

Etapa 14: Selecionamos o disco a ser particionado usando as setas do teclado e pressionamos a tecla *ENTER* para continuar;

Etapa 15: Foi mostrado o esquema de particionamento, confirmamos para continuar;

Etapa 16: Foram pedidas informações sobre o *proxie* de nossa rede, deixamos em branco e depois fomos para a opção “Continuar” e pressionamos a tecla *ENTER*;

Etapa 17: Foi perguntado se o servidor receberá as atualizações automaticamente ou não. Escolhemos a opção “Sem atualizações automáticas” e pressionamos a tecla *ENTER* para continuar;

Etapa 18: Marcamos todas as opções de pacotes para serem instalados no servidor e depois escolhemos a opção “Continuar’ e pressionamos a tecla *ENTER*;

Etapa 19: Na tela de instalação do GRUB, apenas escolhemos a opção “SIM” e pressionamos a tecla *ENTER*;

Etapa 20: Foi mostrada uma tela confirmando o fim da instalação. Escolhemos a opção “Continuar” e pressionamos a tecla *ENTER*. Quando o *desktop foi* reiniciado, retiramos o pen-drive da unidade USB;

Etapa 21: Repetimos a instalação do Ubuntu Server 18.04.3 LTS em todos os “nós” escravos constituintes do cluster.

4.4 CONFIGURAÇÃO DOS IPS ESTÁTICOS

Nó mestre (Cluster 00)

No *Cluster00* foi necessário a instalação de mais uma placa de rede para realizar a configuração da rede interna. Depois que todos os hardwares foram instalados e o Ubuntu em todas as máquinas foi preciso realizar a configuração dos IPs estáticos nas duas placas de rede do *Clustrer00* (“Nó mestre”).

Na placa enp3s2 (rede externa) foi configurado o IP da rede pública. Já na en01 placa a qual foi ligada à rede *Cluster* (rede interna) foi configurada com o endereço 10.0.1.100 com a mask 255.255.255.0.

Etapa 1: Entramos no terminal através do comando:

```
Ctrl + Alt + T
```

Etapa 2: Ativamos a conta root com o comando:

```
~$ sudo su
```

Etapa 3: Depois digitamos a senha de root que foi cadastrada.

Etapa 4: Instalamos o editor de texto nano com o comando abaixo:

 # apt-get install nano

Etapa 5: Entramos no arquivo hosts para configurar os IPs com o comando:

 # nano /etc/hosts

```
10.0.100            cluster00                cluster00
10.0.101            cluster01                cluster01
10.0.102            cluster02                cluster02
10.0.103            cluster03                cluster03
10.0.104            cluster04                cluster04
```

Etapa 6: Entramos no netplan para configurar os IPs estáticos com o comando:

 # cd /etc/netplan

 # ls

 # nano /etc/netplan/ 50-cloud-int.yaml

```
Network:
    version: 2
    render: network
    ethernets:
    en01:
            #dhcp4: yes
            #dhcp6: no
            #adress: [10.0.1.100/24]
            #nameservers:
            #adresses: [8.8.8.8, 8.8.4.4]
```

Nó escravo (*Cluster 01* ao n)

Depois que todos os *hardwares* foram instalados e o Ubuntu Server em todos os “nós escravos” foi necessário realizar a configuração dos IPs estáticos nas placas de rede do *Clustrer 01* ao n com endereço 10.0.1.100 com a mask 255.255.255.0.

Etapa 1: Entramos no terminal através do comando:

Ctrl + Alt + T

Etapa 2: Ativamos a conta root com o comando:

~$ sudo su

Etapa 3: Depois digitamos a senha de root que foi cadastrada;

Etapa 4: Instalamos o editor de texto nano com o comando abaixo:

 # apt-get install nano

Etapa 5: Entramos no arquivo hosts para configurar os IPs com o comando:

 # nano /etc/hosts

Etapa 6: Entramos no netplan para configurar os IPs estáticos com o comando:

 # cd /etc/netplan

 # ls

 # nano /etc/netplan/ 50-cloud-int.yaml

```
Network:
    version: 2
    render: netword
    ethernets:
        en01:
                #dhcp4: yes
                #dhcp6: no
                #adress: [10.0.1.101/24]
                #gateway: 172.20.34.1
                #nameservers:
                #adresses: [8.8.8.8, 8.8.4.4]
```

Repetimos as etapas acima em todos os “nós escravos” com os endereços apresentados abaixo. No caso deste trabalho, como o *Cluster* possuía 4 “nós escravos” essas etapas foram repetidas em todos eles apenas alterando o final do IP, como segue:

- Cluster02: 10.0.1.102;
- Cluster03: 10.0.1.103;
- Cluster04: 10.0.1.104;
- Cluster**n**: 10.0.1.10**n**.

4.5 INSTALAÇÃO DO R

Em todos os nós (*Cluster 00* ao n)

Etapa 1: Entramos no terminal através do comando:

```
Ctrl + Alt + T
```

Etapa 2: Ativamos a conta root com o comando:

```
~$ sudo su
```

Etapa 3: Depois digitamos a senha que foi cadastrada;

Etapa 4: Para instalar o R utilizamos o comando abaixo:

```
# apt-get install r-base r-base-dev
```

Etapa 5: Adicionamos o repositório:

```
# echo deb http://cran.rstudio.com/bin/linux/ubuntu xenial/ |
sudo tee -a /etc/apt/sources.li
```

Etapa 6: Atualizamos com o comando:

```
# apt-get update
```

4.6 ANÁLISE DE DADOS

Em todos os computadores utilizados foi instalado o software R e também os pacotes “snow” e “MCMCglmm”. O pacote “snow” é utilizado para a comunicação entre o computador *Host* e os “nós escravos”, segue um exemplo desse pacote para criar um *cluster* com 3 computadores (cluster01 “host” e o “cluster02” e 03 “nós escravos”):

```
cluster=makeCluster(c("cluster02@10.0.1.102","cluster03@10.0.1
.103",type="SOCK"))
```

Na linha de comando acima, foi criado um objeto com os endereços de IP, 10.0.1.102 e 103, sendo os usuários em cada computador nomeados como “cluster02” e “03”.

O pacote “MCMCglmm” foi utilizado para análise de modelos lineares com abordagem bayesiana. Para tanto, foi utilizada uma cadeia com 1.000.000 de interações, 100.000 de *burnin* e 3 de *thin.* Para análise simultânea em todos os “nós” foi utilizada a seguinte linha de comando:

```
analise=clusterCall(cluster,MCMCglmm,fixed=y1~trtf,random=~laa
qf,data=dados,nitt=1000000,burnin=100000,thin=3)
```

A função “clusterCall” envia para todos “nós” do objeto “cluster”, construído anteriormento com os IP dos “nós” e usuários, ajustarem o modelo e enviar para o “Host” quando as análises finalizarem. Os demais objetos são usualmente utilizados na função “MCMCglmm”, sendo “y1” a característica observada, “trtf” o efeito sistemático dos tratamentos e “laaqf” o efeito da unidade experimental.

O dados utilizados nessas análises foram referentes ao trabalho de Motta et al, 2019. Foram coletados pesos de artêmias (mg), no setor de piscicultura do laboratório de Zootecnia da Universidade Estadual do Norte Fluminense Darcy Ribeiro. O experimento foi em delineamento inteiramente casualizado, com 6 tratamentos e características de peso e comprimento.

Os tempos de todas as análises foram contabilizados, sendo o tempo da cadeia completa em um único computador e a cadeia dividida em todos os computadores e reunida no “Host” quando finalizada nos “nós”.

5. RESULTADOS E DISCUSSÃO

Com o aumento no número de dados e capacidade computacional, algumas técnicas estatísticas vêm sendo otimizadas para maior aproveitamento dos recursos computacionais. Dentre os métodos, o uso da abordagem Bayesiana para paralelismo computacional vem sendo utilizada mais recentemente (Wu 2011, Lagrotta 2017). O uso desses métodos se faz necessário para redução do tempo de ajuste de modelos estatísticos. Visto que algumas análises Bayesianas podem demorar horas e até dias.

Nesse trabalho o tempo de análise com uso da cadeia dividida nos diferentes “nós” chegou a reduzir em 80% (figura 1). Em que a análise em apenas um computador foi de aproximadamente 61 segundos e nos 5 computadores foi de aproximadamente 12 segundos.

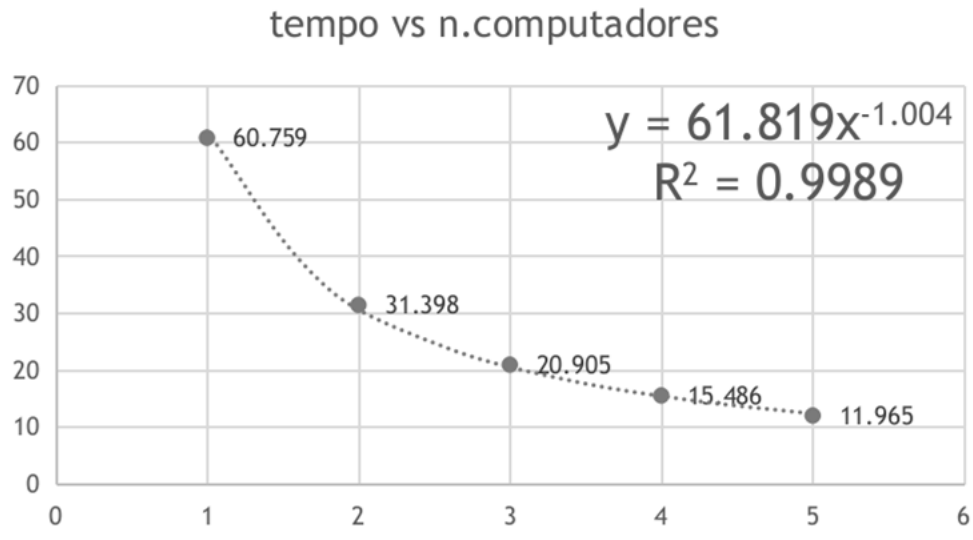


Figura 3: Tempo de análise com aumento do número de computadores (segundos).

Neste contexto, Wu et al. (2011) e Lagrotta (2017) demonstraram a importância da computação paralela e a possibilidade do uso de cadeias paralelas em seleção genômica com abordagem Bayesiana. Em Wu et al. (2011), a análise da

cadeia principal, com cem mil iterações, foi dividida em 10 menores com o mesmo número de interações. O *burn-in* de cada cadeia foi de mil interações. Os autores utilizaram 147 indivíduos genotipados para 50.000 SNPs,concluíram que o processamento via cadeias múltiplas foi 7,7 vezes mais rápido do que algoritmo baseado em paralelização de cadeia única. O mesmo foi observado por Lagrotta (2017), que observou tempo de 77,29 segundos com análise em uma única cadeia e 17,75 com a técnica de divisão da cadeia, uma redução de 77% aproximadamente.

Além do paralelismo do ajuste do modelo na abordagem Bayesiana, o intervalo da média geralmente é menor quando comparado com a abordagem frequentista (tabela 1). Nesse trabalho alguns intervalos chegaram a ser metade do intervalo ajustado com o médodo dos modelos mistos frequentistas utilizados por Motta et al. (2019). da Silva et al. (2020) em trabalho comparando métodos Bayesiana e frequentistas, observaram que os intervalos de credibilidade (Bayesiano) são menores em relação aos intervalos de confiança (frequentista).

Tabela 1: Médias e intervalo de confiança (95%) dos tratamentos para característica de peso dos animais (mg) calculados com abordagem Bayesiana e frequentista

	Bayesiana					Frequentista				
	Média	Inferior	Superior	MI	SM	Média	Inferior	Superior	MI	SM
Tratamento 1	14,9	12,0	17,7	2,9	2,8	14,1	11,1	17,9	3,0	3,8
Tratamento 2	10,1	7,3	12,9	2,9	2,8	13,3	10,5	16,8	2,8	3,5
Tratamento 3	11,2	8,4	14,0	2,8	2,8	12,6	10,0	15,9	2,6	3,3
Tratamento 4	17,6	14,8	20,4	2,8	2,9	23,1	18,3	29,3	4,9	6,2
Tratamento 5	20,5	17,6	23,3	2,8	2,8	24,9	19,7	31,4	5,2	6,5
Tratamento 6	17,0	14,2	19,9	2,9	2,8	21,7	17,2	27,5	4,6	5,8

MI- diferença entre o valor da média e o intervalo de confiança inferior; SM- diferença entre o intervalo de confiança superior e o valor da média.

6. CONCLUSÃO

A utilização de um *cluster* de análise de dados reduz exponencialmente o tempo de análise. A metodologia Bayesiana é otimizada com uso de um *cluster* de análise.

REFERÊNCIA BIBLIOGRÁFICAS

AUFFARTH, Benjamin. How to Build a Linux Cluster for Scientific Computing. **Institute for Bioengineering of Catalonia, Spain**, n. s 25, 2009.

BACELLAR, H. Viana. Cluster: Computação de alto desempenho. **Instituto de Computação, Universidade Estadual de Campinas. Campinas, São Paulo**, 2009.

BECKER, Donald J. et al. BEOWULF: A parallel workstation for scientific computation. In: **Proceedings, International Conference on Parallel Processing.** 1995. p. 11-14.

BRESHEARS, Clay P.; LUONG, Phu. Comparison of openmp and pthreads within a coastal ocean circulation model code. In: **Workshop on OpenMP Applications and Tools**. 2000. p. 2.

BROWN, R. G. **What makes a Cluster Beowulf?** Duke University Physics Department, 2006. Disponível em <www.beowulf.org/overview/howto.html>. Acesso em: 04 Abr. 2019.

CHAPPLE, Simon R. et al. **Mastering Parallel Programming with R**. Packt Publishing Ltd, 2016.

DAGUM, Leonardo; MENON, Ramesh. OpenMP: an industry standard API for shared-memory programming. **IEEE computational science and engineering**, v. 5, n. 1, p. 46-55, 1998.

FERNÁNDEZ, IGNACIO MARTÍNEZ. **Creacion y validacion de um cluster de computacion cientifica baseado em rocks.** Escuela Politecnica Superior, Leganes, Universidad Carlos III de Madrid, 2009. Disponível em: <https://e-archivo.uc3m.es/bitstream/handle/10016/5871/PFC_Ignacio_Martinez_Fernandez.pdf>

GABRIEL, Edgar et al. Open MPI: Goals, concept, and design of a next generation MPI implementation. In: **European Parallel Virtual Machine/Message Passing Interface Users' Group Meeting**. Springer, Berlin, Heidelberg, 2004. p. 97-104.

HABIER, David et al. Extension of the Bayesian alphabet for genomic selection. **BMC bioinformatics**, v. 12, n. 1, p. 186, 2011.

HAYES, B. J. et al. Prediction of total genetic value using genome-wide dense marker maps. **Genetics**, v. 157, n. 4, p. 1819-1829, 2001.

HORNIK, Kurt et al. The r FAQ. 2007.

KINDEL, Marcus. **Projeto e Implementação de um Cluster Beowulf. In:** Escola Regional de Alto Desempenho, 2004, Pelotas, ERAD 2004, p 249 - 252. Disponível

KNAUS, J.; PORZELIUS, C. Tutorial: Parallel computing using R package snowfall. **Knaus, J., Porzelius, C., Binder, H., & Schwarzer, G.(2009). Easier Parallel Computing**, 2009.

KUMAR, Bheshaj et al. **Parallel implementation for fast and efficient image compression in spatial domain.** In: **3rd International Conference on Machine Learning and Computing (ICMLC 2011)**. 2011. p. 378-381.

LAGROTTA, Marcos Rodrigues et al. Computação paralela aplicada à seleção genômica via inferência Bayesiana. **Embrapa Florestas-Artigo em periódico indexado (ALICE)**, 2017.

LEACH, Clint. Introduction to parallel computing in r. **Online tutorial, http://michaeljkoontz. weebly. com/uploads/1/9/9/4/19940979/parallel. pdf**, 2014.

LEMOS, Guido; SOARES, Luiz Fernando Gomes; COLCHER, Sérgio. Redes de Computadores: Das LANs, MANs e WANs às Redes ATM. 2. **Ed. Rio de Janeiro: Campus, 1995.**

MATLOFF, Norman. **The art of R programming: A tour of statistical software design**. No Starch Press, 2011.

MATLOFF, Norman. **Parallel computing for data science: with examples in R, C++ and CUDA**. Chapman and Hall/CRC, 2015.

MCCALLUM, Ethan; WESTON, Stephen. **Parallel R**. " O'Reilly Media, Inc.", 2011

MENDES, Douglas Rocha. **Redes de computadores** – Teoria e prática. 1. ed. São Paulo: Novatec, 2007.

MAZZONETTO, Angela et al. Uma abordagem para a execução paralela de modelos de simulação. 2016.

PINTANGA, Marcos. **Construindo supercomputadores com Linux**. 3. ed. Rio de Janeiro: Brasport, 2008.

ROSSINI, Anthony J.; TIERNEY, Luke; LI, Na. Simple parallel statistical computing in R. **Journal of computational and Graphical Statistics**, v. 16, n. 2, p. 399-420, 2007.

SCHMIDBERGER, Markus et al. State-of-the-art in Parallel Computing with R. **Journal of Statistical Software**, v. 47, n. 1, 2009.

SOARES, Luiz et al. **Rede de Computadores**: Das LANs MANs WANs às Redes ATM. 2º Edição. Rio de Janeiro: Elsevier, 1995.

STERLING, Thomas Lawrence. **Beowulf cluster computing with Linux**. MIT press, 2002.

STOKELY, Murray; ROHANI, Farzan; TASSONE, Eric. **Large-scale parallel statistical forecasting computations in R**. 2011.

SCHMIDBERGER M, Mansmann U (2008). "**Parallelized preprocessing algorithms for high-density oligonucleotide arrays**." In "Proc. IEEE International Symposium on Parallel and Distributed Processing IPDPS 2008," pp. 1–7.

TEAM, R. Core. **Writing R Extensions. R Foundation for Statistical Computing**, Vienna, Austria, 2012.

THEUSSL, Stefan Peter. **Applied high performance computing using R**. na, 2007.

TORGO, Luís. **Introdução à programação em R.** Disponível em: < https: //cran.r-project.org/doc/contrib/Torgo-ProgrammingIntro.pdf. >

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

XIE, X. et al. Jrbridge: **A framework of large-scale statistical computing for R**. In: Proceedings os the 2012 IEEE Asia-Pacific Services Computing Conference. Whashington, DC, USA: IEEE Computer Society, 2012 (APSCC '12), p. 27-347. ISBN 978-0-7695-4897-5. Disponível em: <http://dx.doi.org/10.1109/APSCC.2012.74>.

WHITE, Curt. **Rede de computadores e comunicação de dados.** 6ª. ed. São Paulo: Cengage, 2012.