MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
DE GOIÁS - CÂMPUS URUAÇU

GUILHERME EDUARDO ALVES SILVA FONSECA

IMPLEMENTAÇÃO DE UM CLUSTER COM COMPUTADORES SUBUTILIZADOS DOS LABORATÓRIOS DO INSTITUTO FEDERAL DE GOIÁS – CÂMPUS URUAÇU

URUAÇU
2022

INSTITUTO FEDERAL
Goiás

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
SISTEMA INTEGRADO DE BIBLIOTECAS

TERMO DE AUTORIZAÇÃO PARA DISPONIBILIZAÇÃO NO REPOSITÓRIO DIGITAL DO IFG - ReDi IFG

Com base no disposto na Lei Federal nº 9.610/98, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia de Goiás, a disponibilizar gratuitamente o documento no Repositório Digital (ReDi IFG), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, em formato digital para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IFG.

Identificação da Produção Técnico-Científica

[ ] Tese	[ ] Artigo Científico
[ ] Dissertação	[ ] Capítulo de Livro
[ ] Monografia – Especialização	[ ] Livro
[X] TCC - Graduação	[ ] Trabalho Apresentado em Evento
[ ] Produto Técnico e Educacional - Tipo: _____	

Nome Completo do Autor: Guilherme Eduardo Alves Silva Fonseca
Matrícula: 20181050100299
Título do Trabalho: Implementação de um cluster com computadores subutilizados do laboratórios do Instituto Federal de Goiás - Câmpus Uruaçu

Autorização - Marque uma das opções

1. (X) Autorizo disponibilizar meu trabalho no Repositório Digital do IFG (acesso aberto);
2. ( ) Autorizo disponibilizar meu trabalho no Repositório Digital do IFG somente após a data ___/___/_____ (Embargo);
3. ( ) Não autorizo disponibilizar meu trabalho no Repositório Digital do IFG (acesso restrito).

Ao indicar a opção **2 ou 3**, marque a justificativa:

( ) O documento está sujeito a registro de patente.
( ) O documento pode vir a ser publicado como livro, capítulo de livro ou artigo.
( ) Outra justificativa: _____

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O/A referido/a autor/a declara que:

i. o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;
ii. obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autor/a, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia de Goiás os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;
iii. cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia de Goiás.

Uruaçu, 08/08/2022
Local Data

Guilherme Eduardo Alves Silva Fonseca
Assinatura do Autor e/ou Detentor dos Direitos Autorais

GUILHERME EDUARDO ALVES SILVA FONSECA

IMPLEMENTAÇÃO DE UM CLUSTER COM COMPUTADORES SUBUTILIZADOS DOS LABORATÓRIOS DO INSTITUTO FEDERAL DE GOIÁS – CÂMPUS URUAÇU

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia de Goiás - Câmpus Uruaçu ligado ao Ministério da Educação, como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Área de concentração: Análise e Desenvolvimento de Sistemas

Linha de pesquisa: Sistemas Distribuídos

Orientador: Maurílio Humberto Rodrigues Miranda
Instituto Federal de Ciências e Educação de Tecnologia de Goiás

URUAÇU
2022

Dados Internacionais de Catalogação na Publicação (CIP)

Fonseca, Guilherme Eduardo Alves Silva.

F676i Implementação de um cluster com computadores subutilizados dos laboratórios do Instituto Federal de Goiás – Câmpus Uruaçu [manuscrito] / Guilherme Eduardo Alves Silva Fonseca. - 2022.
XXXII, 32 f.: il.

Orientador: Prof. Me. Maurílio Humberto Rodrigues Miranda.

Trabalho de Conclusão de Curso (Graduação) - Instituto Federal de Educação, Ciência e Tecnologia de Goiás: Campus Uruaçu, Tecnologia em Análise e Desenvolvimento de Sistemas, 2022.
Bibliografia. Anexos.
Inclui lista de figuras, lista de abreviaturas e siglas, figuras.

1. Computação. 2. Cluster (Sistema de computador). 3. Computadores. 4. Hardware - reutilização. I. Miranda, Maurílio Humberto Rodrigues (orientador). II. Instituto Federal de Educação, Ciência e Tecnologia de Goiás. III. Título.

CDD: 004.35

Ficha catalográfica elaborada pela Bibliotecária Sabrina Gisele da Silva Felix CRB1/2561

Guilherme Eduardo Alves Silva Fonseca

IMPLEMENTAÇÃO DE UM CLUSTER COM COMPUTADORES SUBUTILIZADOS DOS LABORATÓRIOS DO INSTITUTO FEDERAL DE GOIÁS – CAMPUS URUAÇU

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas ao Departamento de Áreas Acadêmicas do Instituto Federal de Educação, Ciência e Tecnologia de Goiás - Campus Uruaçu.

Aprovado em 10 de junho de 2022.

(Assinado eletronicamente)
Prof. Me. Maurílio Humberto Rodrigues Miranda
Instituto Federal de Goiás - Campus Uruaçu - Orientador

(Assinado eletronicamente)
Prof. Me. Alessandro Siqueira da Silva
Instituto Federal de Goiás - Campus Uruaçu

(Assinado eletronicamente)
Prof. Esp. Mateus Nunes dos Santos
Instituto Federal de Goiás - Campus Uruaçu

Documento assinado eletronicamente por:
- **Alessandro Siqueira da Silva, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 12/07/2022 21:07:19.
- **Mateus Nunes dos Santos, TEC DE TECNOLOGIA DA INFORMACAO**, em 12/07/2022 19:58:08.
- **Maurilio Humberto Rodrigues Miranda, CHEFE - CD4 - URU-DAA**, em 12/07/2022 14:58:38.

Este documento foi emitido pelo SUAP em 12/07/2022. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifg.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 302762
Código de Autenticação: cf8b1f5128

RESUMO

FONSECA, Guilherme Eduardo Alves Silva. IMPLEMENTAÇÃO DE UM CLUSTER COM COMPUTADORES SUBUTILIZADOS DOS LABORATÓRIOS DO INSTITUTO FEDERAL DE GOIÁS – CÂMPUS URUAÇU. 2022. 32 f. Trabalho de Conclusão de Curso – Tecnologia em Análise e Desenvolvimento de Sistemas, Instituto Federal de Educação, Ciência e Tecnologia de Goiás - Câmpus Uruaçu. Uruaçu, 2022.

O objetivo desse trabalho de conclusão de curso é implementar um cluster Beowulf com os computadores subutilizados dos laboratórios do Instituto Federal de Goiás – Câmpus Uruaçu, e através desse projeto evitar o descarte de máquinas úteis apesar de estarem defasadas a nível de hardware e por meio dessa implementação aproveitar o desempenho dessas máquinas em conjunto no cluster. O método usado no projeto foi o hipotético-dedutivo, onde foi unido a racionalização e a experimentação, com esse método solucionaremos o problema dos computadores dos laboratórios do IFG. Portanto, com a conclusão deste projeto teremos um cluster Beowulf que será usado para as pesquisas científicas das áreas de física, química e etc.
Palavras-chave:Cluster. Computadores subutilizados. Reutilização de hardware.

ABSTRACT

FONSECA, Guilherme Eduardo Alves Silva. IMPLEMENTATION OF A CLUSTER WITH UNDER-USED COMPUTERS FROM THE LABORATORIES OF INSTITUTO FEDERAL DE GOIAS - CÂMPUS URUAÇU . 2022. 32 f. Trabalho de Conclusão de Curso – Tecnologia em Análise e Desenvolvimento de Sistemas, Instituto Federal de Educação, Ciência e Tecnologia de Goiás - Câmpus Uruaçu. Uruaçu, 2022.

The objective of this course conclusion work is to implement a Beowulf cluster with underused computers from the laboratories of the Instituto Federal de Goiás - Câmpus Uruaçu, and through this project to avoid the disposal of useful machines despite being outdated at the hardware level and through this implementation take advantage of the performance of these machines together in the cluster. The method used in the project was the hypothetical-deductive, where rationalization and experimentation were joined, with this method we will solve the problem of computers in the IFG laboratories. Therefore, with the conclusion of this project, we will have a Beowulf cluster that will be used for scientific research in the areas of physics, chemistry, etc..

Keywords: Cluster. Underutilized computers. Hardware reuse.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
DECOM	Departamento de Computação
IFG	Instituto Federal de Educação, Ciência e Tecnologia de Goiás
PVFS2	Sistema de Arquivo Virtual Paralelo
MPI	Interface de Passagem de Mensagens
ONU	Organização das Nações Unidas
REE	Resíduos Eletroeletrônicos
SO	Sistema Operacional
BIOS	Sistema Integrado de Entrada e Saída
IP	Protocolo da Internet
SSH	Protocolo de rede criptográfico
NFS	Protocolo de sistema de arquivos distribuídos
ISO	Imagem de arquivos gravados
RAM	Memória de Acesso Aleatório
MB	Megabyte
IPv4	Protocolo de Internet versão 4
GNU	Sistema Operacional tipo Unix
LGPL	Licença de software livre e aprovada
TDP	Quantidade de calor gerado pelo processador ou placa mãe

SUMÁRIO

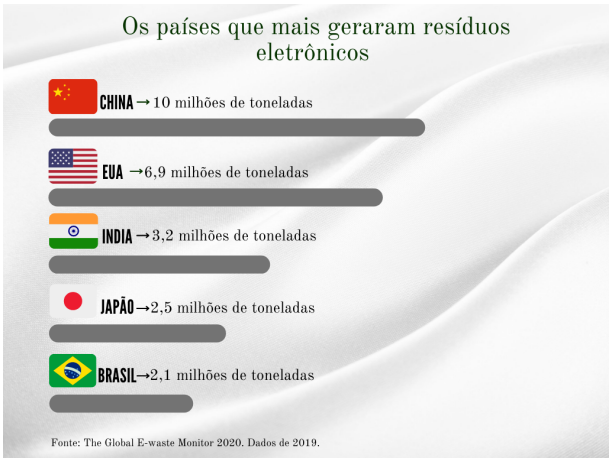
1 INTRODUÇÃO

Atualmente, podemos notar o grande número de aplicações que exigem um poder de processamento de dados e em paralelo cresce a quantidade de computadores que são descartados por seus usuários por não serem capazes de processar essas aplicações. O principal motivo do usuário efetuar o descarte é porque ele não tem o conhecimento do reaproveitamento do hardwares ali contido na máquina em questão. Essas máquinas são taxadas como inúteis e são descartadas por não terem nenhuma finalidade para o seu usuário, aumentando a porcentagem de lixo eletrônico que é descartado por ano no mundo.

A nossa sociedade recentemente está habituada com a economia linear que é insustentável porque ela se baseia em extrair os recursos naturais produzir os bens e depois de usados, eles podem ser descartados por ser motivos de não funcionamento ou por ficarem obsoletos. Uma vez que o equipamento descartado é um eletroeletrônico ele vai gerar resíduos eletroeletrônicos (REE), (AFONSO, 2018). Um REE pode variar de um rádio comunicador até um servidor que foram descartados por mal funcionamento ou porque ficaram obsoletos para desempenhar as suas funções, esse tipo de lixo pode ser descartado por residências, universidades, empresas e governos. (UFSC, 2021)

O relatório Global E-Waste Monitor 2017, lançado pela Universidade da ONU e pela Associação Internacional de Resíduos Sólidos mostra que o volume de lixo eletrônico tem aumentado. Em 2016, foram gerados 44,7 milhões de toneladas de lixo eletrônico, na América Latina o Brasil foi o país que mais descartou lixo eletrônico com 1,5 milhões de toneladas, um aumento de 8% em comparação com 2014. O gráfico da figura mostra que o Brasil ficou em 5° lugar entre no rank de países que geram mais resíduos eletrônico produzindo 2,1 milhões de toneladas conforme mostra o gráfico abaixo. (ELETRON, 2020).

Figura 1 – Países que mais geraram resíduos eletrônicos



Fonte: The Global E-Waste Monitor 2020. Dados de 2019.

O lixo eletrônico quando descartado de maneira incorreta pode causar riscos prejudiciais à saúde como "problemas respiratórios e danos ao sistema nervoso podendo ser desencadeados a partir da contaminação do organismo com mercúrio, chumbo e cádmio presentes nos eletrônicos", explica a professora da Poli-USP Tereza Cristina Carvalho (MARQUES, 2018).

Os resíduos eletrônicos que são descartados em lixões ou aterros contêm componentes internos com pequenas quantidades de materiais valiosos como cobre, ouro e chumbo. Diante disso a extração desses materiais podem trazer riscos para as pessoas que tentam extrair esses metais usando técnicas rudimentares sendo eles a queima, imersão em banho químico ou aquecimento. Nesse processo são lançados resíduos tóxicos no ar podendo contaminar o solo, água e também as pessoas que participam desse trabalho informal sendo elas na maioria crianças e mulheres, esse cenário é visto em países subdesenvolvidos. Os resíduos tóxicos podem causar alterações na função pulmonar, habilidades intelectuais, prejuízos à função da tireoide e podendo causar também câncer e doenças cardiovasculares, (ELETRON, 2021).

Locais como escolas e empresas passam por essa situação de não fazer a reutilização de computadores obsoletos, uma solução para evitar o descarte de máquinas que ainda podem ser úteis seria montar um cluster onde dependendo da necessidade do meio no qual essas máquinas se encontram seria uma opção viável para evitar o descarte da máquinas.

Considerando esse contexto de hardware obsoleto que não está sendo usado e que pode ser reutilizado, uma alternativa para reutilizar essas máquinas é através da clusterização, fazer a união dessas máquinas de forma lógica, formando através dessa conexão um super computador que pode ser usado de várias formas dependendo apenas do tipo de cluster que será implementado nas máquinas.

Um exemplo da utilização do cluster fazendo o reaproveitamento das máquinas, é o que acontece em instituições como IFG em específico no Câmpus Uruaçu, nos laboratórios de informática que contém computadores antigos que estão com um bom funcionamento, mas como não estão mais atendendo às necessidades dos alunos em suas aplicações, com isso foi efetuada a troca desses computadores por novos de alta performance, onde os antigos foram realocados para uma outra sala onde estão outros computadores que foram trocados por motivos semelhantes e os mesmos estão parados, apesar de que poderiam estar sendo utilizados em um cluster.

Na maioria das vezes o cluster é formado por computadores convencionais e se apresenta de forma transparente ao usuário, como sendo um único computador de grande porte, segundo (FUGI, 2015). Portanto o cluster pode ser formado por computadores que qualquer arquitetura, qualquer geração de processadores, isso não irá interferir em nada na configuração do cluster.

Pensando nas aplicações diárias do IFG o cluster escolhido foi o Beowulf, ele combinação de 2 tipos de clusters, sendo eles: cluster alto desempenho em conjunto com o

cluster de balanceamento de carga onde a sua principal característica é a escalabilidade, ou seja, pode ser adicionado um ou mais nós ao cluster quando for necessário, além da compatibilidade com qualquer arquitetura, o cluster irá ajudar em todos os cursos da instituição em questão do desempenho, pois se um aluno ou professor precisar de um computador que necessita que detêm de um poder computacional maior para executar códigos matemáticas que levariam dias ou horas em computadores convencionais, no cluster Beowulf será executado em minutos dependendo da quantidades de máquinas que será composta no cluster. O cluster Beowulf é o mais apto para a instituição porque além de ter um alto desempenho executar aplicações de forma rápida a sua outra característica é a divisão de carga entre os computadores fazendo com que não haja sobrecarga nas máquinas que serão usadas. Além da facilidade de administrar os nós, onde podemos adicionar, remover e dar manutenção sem atrapalhar o funcionamento do cluster, (FUGI, 2015).

Portanto os computadores que seriam destinados a ficar parados esses serão realocados para implementação do cluster Beowulf fazendo reaproveitamento das máquinas que iriam ficar inutilizáveis para evitar o descarte de máquinas úteis.

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolver um Cluster de Beowulf com computadores defasados a nível de hardware no IFG Câmpus Uruaçu.

1.1.2 Objetivos Específicos

- Aproveitar máquinas inutilizadas.
- Evitar descarte de máquinas úteis que não estão sendo mais usadas por não estarem atendendo às necessidades de seus usuários.
- Melhor utilização de recurso público.

1.2 Organização

Neste tópico é feita uma breve descrição de como o restante do trabalho está organizado, começando pelo Capítulo 2 que é o de fundamentação teórica, no tópico 2.1 falaremos sobre os conceitos sobre sistemas distribuídos. No subtópico 2.1.1 os tipos de sistemas distribuídos. No tópico 2 falará sobre o conceito de cluster. No subtópico 2.2.1 os tipos de cluster. No Capítulo 3, que é referente a metodologia, falaremos qual método foi utilizado para desenvolver o projeto. No Capítulo 4 que refere se a implementação, onde apresentará como foi implementado o cluster Beowulf. No tópico 4.1 será feito o detalhamento de como foi montado o cluster Beowulf voltado para parte de hardware e as

especificações das máquinas. No tópico 4.2 falaremos sobre a estrutura da rede, como foi elaborada para suportar o cluster e fazendo um breve explicação sobre como foi feita a montagem da parte de rede do cluster Beowulf. no tópico 4.3 falaremos sobre a implementação lógica, onde será relatado detalhadamente como foi realizado a implementação do cluster e as experiências vividas e os conhecimentos adquiridos ao decorrer desse trabalho de conclusão de curso. No Capítulo 5 realizaremos a análise dos resultados que foram coletados através dos códigos usado no cluster Beowulf. O tópico 5.1 falaremos sobre a análise dos resultados obtidos pela execução do código de números primos. O tópico 5.2 falaremos sobre a análise dos resultados obtidos pela execução do código de multiplicação de matrizes. No Capitulo 6 falaremos sobre a Conclusão do trabalho mediante aos desafios e resultados obtidos e também sobre os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo falaremos sobre o conceito de sistemas distribuídos, as vantagens do seu uso, tipos de sistemas distribuídos, falaremos também do conceito de cluster e os seus tipos. Assim teremos um bom embasamento teórico, sabendo os conceitos básicos para a implementação de um cluster.

2.1 Sistemas Distribuídos

Um sistema distribuído é formado por um conjunto de computadores independentes que se apresenta a seus usuários como um sistema único e coerente. Esse conceito tem vários aspectos importantes. O primeiro é que um sistema distribuído consiste componentes autônomos. Quando se diz componente refere-se tanto ao hardware quanto ao software, esse conceito se encaixa também a servidores ou máquinas conectadas pela rede. A construção de um sistema distribuído precisa garantir características como, (COULOURIS et al., 2013):

- Heterogeneidade: eles devem ser construídos a partir de uma variedade de redes, sistemas operacionais, hardware e linguagens de programação diferentes. Os protocolos de comunicação da Internet mascaram a diferença existente nas redes e o middleware pode cuidar das outras diferenças.
- Sistemas abertos: os sistemas distribuídos devem ser extensíveis, o primeiro passo é publicar as interfaces dos componentes, mas a integração de componentes escritos por diferentes programadores é um desafio real.
- Escalabilidade: um sistema distribuído é considerado escalável se o custo da adição de um usuário for um valor constante, em termos dos recursos que devem ser adicionados. Os algoritmos usados para acessar dados compartilhados devem evitar gargalos de desempenho, e os dados devem ser estruturados hierarquicamente para se obter os melhores tempos de acesso. Os dados acessados frequentemente podem ser replicados.
- Tratamento de falhas: qualquer processo, computador ou rede pode falhar, independentemente dos outros. Portanto, cada componente precisa conhecer as maneiras possíveis pelas quais os componentes de que depende podem falhar e ser projetado de forma a tratar de cada uma dessas falhas apropriadamente.
- Concorrência: a presença de múltiplos usuários em um sistema distribuído é uma fonte de pedidos concorrentes para seus recursos. Em um ambiente concorrente, cada recurso deve ser projetado para manter a consistência nos estados de seus dados.
- Transparência: o objetivo é tornar certos aspectos da distribuição invisíveis para o programador de aplicativos, para que este se preocupe apenas com o projeto de seu aplicativo em particular. Por exemplo, ele não precisa estar preocupado com sua

localização ou com os detalhes sobre como suas operações serão acessadas por outros componentes, nem se será replicado ou migrado. As falhas de rede e de processos podem ser apresentadas aos programadores de aplicativos na forma de exceções – mas elas devem ser tratadas.

- Qualidade do serviço: não basta fornecer acesso aos serviços em sistemas distribuídos. Em particular, também é importante dar garantias das qualidades associadas ao acesso aos serviços. Exemplos dessas qualidades incluem parâmetros relacionados ao desempenho, à segurança e à confiabilidade.

2.1.1 Tipos de Sistemas Distribuídos

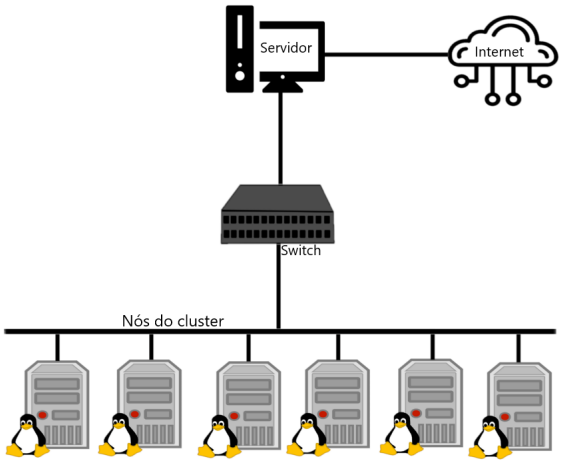
Os sistemas distribuídos são divididos por categorias: sistema de computação distribuídos, sistemas de informação distribuídos e sistema distribuídos pervasivos (TANENBAUM; STEEN, 2007).

- Sistema de computação distribuídos: essa categoria de sistemas é direcionada para computação de alto desempenho ela se subdivide em: computação em cluster e computação em grade.
  - Computação em Cluster: é um conjunto de nós de computação controlados e acessados por meio de um único nó mestre também chamado de servidor. As tarefas típicas do mestre são manipular a alocação dos nós a um determinado programa paralelo, manter uma fila de jobs apresentados e proporcionar uma interface para os usuários do sistema.
  - Computação em Grade: é um sistema que tem um alto grau de heterogeneidade, a questão fundamental dessa computação é que recursos de diferentes organizações são reunidos para permitir a colaboração de um grupo de pessoas ou instituições, essa colaboração pode ser chamada também de organização virtual. As pessoas pertencem as mesma organização virtual tem direito de acesso aos recursos fornecidos por aquela organização podendo ser um deles um servidor de computação, facilidades de armazenamento e acesso a um banco de dados. Também podem ser oferecidos acesso a equipamentos que estiverem em rede como telescópios ou sensores entre outros.
- Sistema de Informação Distribuídos: é um conjunto de processos concorrentes que estão acessando recursos distribuídos, eles podem ser compartilhados ou replicados, ele se subdivide em: sistema de processamento de transação, integração de aplicação empresarial.
- Sistema Distribuído Pervasivo: são equipamentos que são caracterizados por seu pequeno tamanho, pela alimentação por bateria, por sua mobilidade e por terem somente uma conexão sem fio, se bem que nem todas essas características se aplicam a todos os dispositivos. Um exemplo de sistemas pervasivos são: redes de sensores, sistemas domésticos e sistemas eletrônicos para tratamento de saúde.

2.2 Cluster

A palavra cluster é um termo em inglês onde sua tradução é a palavra aglomeração, esse termo pode ser aplicado em vários contextos. Mas neste trabalho o contexto que nos importa é da computação, ele define uma arquitetura de sistema onde pode se aglomerar vários computadores para que trabalham juntos. Cada computador é denominado com nó ou nodo, esses nodos combinados formam o cluster, eles podem ser referenciados de como supercomputadores ou computação em cluster, conforme mostrado na imagem abaixo.

Figura 2 – Exemplo esquemático do Cluster Beowulf



Fonte: Próprio autor.

2.2.1 Tipos de Cluster

O cluster pode atender às necessidades de uma instituição de ensino por meio das suas diferentes funcionalidades, que carregam os seus diferentes tipos, segundo (COULOURIS et al., 2013) são:

- Cluster de Alto Desempenho (*High Performance Computing Cluster*): esse cluster tem o foco principal em proporcionar um alto desempenho na execução das aplicações, ou seja, ele busca fornecer os resultados mais desejáveis em um tempo eficiente, independentemente do quão complexa seja a tarefa ele deverá executar no menor tempo possível. Por isso eles são recomendados em atividades que exigem o maior uso de processamento, um exemplo são os sistemas de pesquisas científicas. Esse tipo de é o que será implementado nesse trabalho para o uso no Instituto para pesquisas científicas de diferentes áreas.
- Cluster de Alta Disponibilidade (*High Availability Computing Cluster*): o cluster de alta disponibilidade tem como o objetivo manter uma aplicação funcionando sem ter nenhuma interrupção, mas se acontecer alguma interrupção por problemas gerados

por um nó, nada vai acontecer porque o mecanismo vai continuar funcionando como se não tivesse acontecido nada na interação com o usuário. Esse tipo de cluster é inferior no quesito de desempenho porque ele está preocupado em deixar aplicação rodando. Mesmo que um nó venha a sofrer com interrupções, a aplicação estará disponível ao usuário. Esse cluster pode ser usado em servidores de arquivos que não podem ficar indisponíveis.

- Cluster para Balanceamento de Carga (*Load Balancing*): esse cluster tem como objetivo, fazer a distribuição das tarefas de forma igualitária entre os nós, sendo assim cada nó vai executar a mesma quantidade de processos, cada nó irá receber e atender uma tarefa específica, se um nó ficar indisponível as tarefas são redistribuídas automaticamente de forma igualitária entre os nós, esse tipo de cluster deve ser monitorado continuamente porque se um nó não estiver com as mesma quantidades de processos o cluster estará desbalanceado, ou seja, a pessoa responsável pela manutenção do cluster terá que balancear ele e deixar esse nó com as mesma quantidades de processos que os outros nós. O cluster para balanceamento de carga é utilizado em torre de servidores que atendem uma grande demanda em site e que usam essas torres.

2.3 Cluster Beowulf

A computação em cluster originou-se com o projeto SAGE (*Semi-Automatic Ground Enrironment*), construído para o NORAD (*North American Air Defense*) pela IBM e entrou em operação no ano de 1962, seu objetivo consiste em diversos sistemas separados trabalhando de forma cooperativa para monitorar invasões aéreas no continente norte-americano. (PITANGA, 2008)

Figura 3 – SAGE- Sistema A

Fonte: (PITANGA, 2008)

Durante a década de 80 o interesse pela computação em cluster teve um aumento

significativo devido os vários experimentos de pesquisa e também por ser bastante usado na indústria. A primeira implementação que cunhou o nome cluster foi a *Digital Equipament Corporation*, que desenvolveu um sistema compreendido por computadores VAX 11/750. Em 1993 a NASA já utilizava um pequeno cluster de estações da IBM utilizado para simular partidas de naves espaciais, nesse mesmo ano foi implementado o projeto NOW da UC Berkeley sendo assim ranqueado como o primeiro cluster na lista do top500.org dos quinhentos supercomputadores mais rápidos do mundo. Em 1994 nasceu o cluster Beowulf, porque a agência espacial estava precisando de um equipamento que processasse na ordem de um gigaflop, ou seja, um bilhão de ponto de operações em ponto flutuante por segundo. Os cluster Beowulf foi formado por 16 computadores pessoais, cada um contendo um processador 486 usando o SO Linux e uma rede padrão Ethernet, ele foi usado para aplicações científica chegando a 70 megaflops. O nome Beowulf em si foi retirado de um poema antigo da língua inglesa, que conta a história de um cavaleiro inglês que detêm de uma força imensurável e valentia em sua saga pra derrotar o mostro de Grendel. [1]

[1] https://www.top500.org. Data de acesso: 20/12/21

3 METODOLOGIA

Metodologia é uma ciência que estuda os métodos utilizados no processo de conhecimento. Nesse projeto predominará a metodologia hipotético-dedutiva método que busca a união do método dedutivo acrescentando-o a racionalização e ao método indutivo a experimentação. Esse método foi desenvolvido por Karl R. Popper, consiste em escolher hipóteses das quais proporcionará a resolução para um determinado problema de natureza científica. Através desse método tentaremos solucionar o problema dos computadores que não estão sendo usados nos laboratórios do IFG Câmpus Uruaçu de forma racional através da experiência de implementar um cluster Beowulf de alto desempenho.

Nesse projeto foi realizado uma pesquisa bibliográfica sobre os conceitos que ajudaram a entender como de fato funciona esse tipo de cluster e buscar entender a função de cada linha de comando que será executada nas máquinas. Com a busca dos conceitos deu-se início á implementação com um mestre e um escravo, foram testadas as memórias, placas mães e processadores. Em seguida foi o momento de montar a rede física, cada computador tem o seu cabo de rede em que o mesmo é conectado a um *switch* que interliga todos, cada computador é configurado sendo eles os nós e o servidor. Logo foi realizado instalação do sistema operacional onde o escolhido foi o Lubuntu por ser mais leve mesmo usando interface gráfica, foi instalado também o pacote NFS que faz o compartilhamento e sincronização do diretório que foi usado no cluster sendo uma pasta compartilhada com todos os nós através do servidor, com essa etapa concluída foi realizada a configuração do pacote SSH onde ele foi usado para fazer gerar uma chave pública que foi compartilhada nos nós para que assim o mestre pudesse acessar os nós sem precisar digitar em todo o momento do acesso a senha do nó isso foi crucial para o funcionamento do cluster Beowulf.

Em seguida instalamos o programa OPENMPI que contém a biblioteca MPI responsável por fazer a troca de mensagens entre os nós e o servidor possibilitando assim o paralelismos no momento da execução dos códigos.

4 IMPLEMENTAÇÃO

A implementação foi realizada no Instituto Federal de Goiás - Câmpus Uruaçu, no bloco 300 no 1° andar na sala 106 que é um laboratório de informática, todas as máquinas usadas no cluster Beowulf já foram usadas em outros laboratórios onde foram trocadas por máquinas novas, as máquinas defasadas algumas delas foram destinadas a ficar em um laboratório onde talvez poderiam ser usadas e outras foram realocadas a ficar no subsolo onde elas estão amontoadas como se fossem lixo esperando para serem descartas, algumas delas não estão mais funcionando mas a maioria funciona normalmente. Algumas máquinas apresentaram dificuldades em inicializar o pendrive bootável porque tinham a senha de administrador para acessar a *BIOS*, esse problema logo foi resolvido fazendo o resete da bios pelo *jumper*.

As máquinas que não estavam ligando foram submetidas a limpeza da memória ram com uma borracha fazendo a remoção da ferrugem do contato da memória, e também algumas máquinas foram submetidas a troca da fonte de alimentação onde as fontes que foram trocadas eram de máquinas que não ligaram com os procedimentos ditos anteriormente fazendo assim o reuso de peças como a própria fonte de alimentação e também de memória ram aumentando as capacidade de algumas máquinas que estavam com 512MB apenas de ram e que foram para 2GB de memória ram por máquina aumentando a velocidade delas consideravelmente.

4.1 Implementação Física

Para a instalação física do Cluster Beowulf foi preciso uma estante para deixar as máquinas onde por partilheira foram armazenados duas máquinas deitadas sendo o total na estante por enquanto 12 máquinas escravas o mestre ficou na mesa ao lado. Todos os computadores tem uma cabo de rede conectado a um switch de 24 portas, os cabos foram organizados e foi feito o mapeamento de cada porta do switch já pensando em um futuro problema nos cabos de rede sabendo qual cabo pertence a qual máquina. Em sequência foi montando a mesa onde ficará o mestre colocando nela um monitor e um kit teclado e mouse com fio.

Cada máquina tem um processador Intel Core 2 Duo E8400, com 2 núcleos sendo o número de unidades de processamento central independentes em um único componente de computação (matriz ou chip), com uma frequência de 3.00 GHz a frequência baseada em processador descreve a frequência com que os transistores de um processador abrem e fecham. A frequência baseada em processador é o ponto operacional em que o TDP é definido e medida em gigahertz (GHz), ou bilhões de ciclos por segundo (INTEL,).

Foi usado um switch conectado a internet e no mesmo havia os cabos de rede

conectando as máquinas sendo ela os nós e o servidor. Para provar que o cluster funciona em arquiteturas diferentes as máquinas usadas são de gerações e arquiteturas diferentes sendo assim qualquer computador independente da sua capacidade de processamento pode ser usado em um Cluster Beowulf. A imagem abaixo mostra como ficaram organizados as máquinas que são compostas o cluster Beowulf.

Figura 4 – Implementação Física do cluster Beowulf

Fonte: Próprio autor.

4.2 Rede

Para montar a rede do Cluster foi usado um switch conectado a internet e no mesmo havia os cabos de rede conectando as máquinas sendo ela os nós e o servidor. As máquinas foram configuradas com *IP* estáticos para não gerar conflito caso o servidor ou o nó perdesse o *IP* que foi usado para configurar a pasta compartilhada em rede por exemplo ou até mesmo um *IP* dinâmico poderia ocasionar no erro de SSH sendo que o mestre iria fazer a conexão baseada no arquivo hosts que foi configurado com um *IP* X mas quando a máquina foi reiniciada pegou um *IP* Y sendo assim não iria conectar no nó podendo dar erro e não executar o código de execução paralela, por esse motivo foi usado

os *IP* estáticos conforme mostra na imagem abaixo o endereço do mestre e também dos escravos.

Figura 5 – Lista de IP do cluster

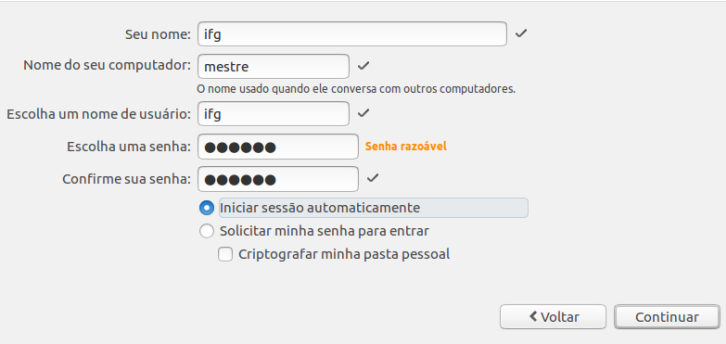
Nome	IP	Máscara de rede	Gateway
Mestre	192.168.0.130	255.255.255.0	192.168.0.1
No1	192.168.0.131	255.255.255.0	192.168.0.1
No2	192.168.0.132	255.255.255.0	192.168.0.1
No3	192.168.0.133	255.255.255.0	192.168.0.1
No4	192.168.0.134	255.255.255.0	192.168.0.1
No5	192.168.0.135	255.255.255.0	192.168.0.1
No6	192.168.0.136	255.255.255.0	192.168.0.1
No7	192.168.0.137	255.255.255.0	192.168.0.1

Fonte: Próprio autor.

4.3 Implementação lógica

A implementação do cluster começou com a coleta delas, algumas foram coletadas do laboratório 106 outras estavam no subsolo, todos as máquinas usadas no cluster estavam paradas por ter a capacidade de poder de processamento necessário para o uso dos alunos ou professores. Mas no cluster Beowulf qualquer hardware pode ser usado. Após a coleta das máquinas levamos pro laboratório 106 para começar a configuração delas o Sistema Operacional escolhido foi o Lubuntu por ser é um sistema Linux que foi baseado no Ubuntu, ele foi escolhido por ser fácil de usar e também por ser leve, sua ISO tem o tamanho de 900 MB e com os requisitos do hardware para suportar ele é qualquer máquina que tenha 512 MB de memória RAM, ou seja, qualquer máquina antiga consegue rodar o Lubuntu, a versão usada no cluster foi a 16.04.2 da arquitetura 32 bits usando a interface gráfica.

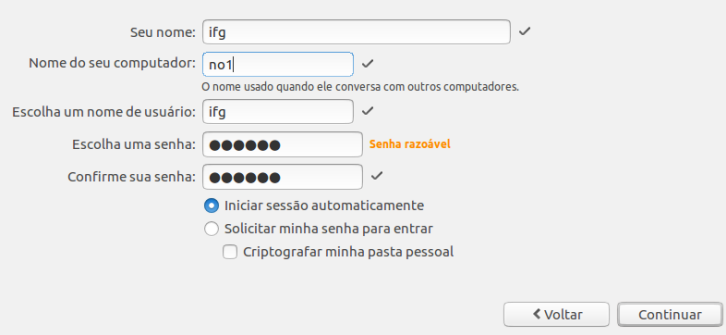
Figura 6 – Configuração de nomeação do nó Mestre



Fonte: Próprio autor.

Executando então a ISO do Lubuntu, o usuário deve se atentar a parte de: nomeação do computador, nome do usuário, senha, e o campo seu nome na imagem acima mostra como foi feita a nomeação do mestre e também do nó1 onde a palavra ”ifg”é usada tanto no campo seu nome quanto no campo da nomeação do usuário, esse nome foi o mesmo para simplificar a configuração do SSH e o NFS onde usando o mesmo nome pode

Figura 7 – Configuração de nomeação do nó1



Fonte: Próprio autor.

evitar a confusão usuário quando for implementar um cluster, a senha escolhida para o cluster foi 111111 pois é fácil de lembrar.

Com a instalação do Lubuntu concluída vamos para a configuração de redes dos nós, os IPs foram configurados de forma estática para facilitar a configuração do SSH e NFS, se colocar dinâmico quando uma máquina desligar e com isso pegar outro IP que não estiver cadastro e configurado no arquivos de host no caso do mestre ou *fstab* no caso dos nos pode dar problema na pasta que será compartilhada do mestre para todos os escravos. A configuração da placa de rede foi feita fora do terminal de comando por ser mais fácil de configurar, foi editado a configuração IPv4, o método escolhido foi o manual, a faixa de IP usada foi a mesma da rede do Câmpus Uruaçu. Seguindo a faixa de IP do mestre assim também foi feita a configuração dos demais nós do cluster.

Após a configuração da placa de rede usando o IP estático.

ifg@mestre: sudo apt-get update

Esse comando atualiza o repositório do SO e informa ao sistema se há pacotes novos ou não. Em sequência o usuário deverá atualizar os pacote instalados com o seguinte comando:

ifg@mestre: sudo apt-get upgrade

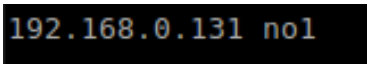
O próximo é para a configuração do arquivo $/etc/hosts $. Nesse arquivo foi mapeado os IP's de cada nó em todas as máquinas que, quando for conectar via SSH, não será preciso digitar o IP; apenas o apelido que foi dado para configurar esse arquivo, para começar basta digitar o seguinte comando:

ifg@mestre: sudo nano /etc/hosts

No arquivo de hosts do mestre ou servidor é necessário colocar apenas os IP's dos nós e o apelido, já no arquivo de hosts dos nós é necessário colocar apenas o Ip e o apelido do mestre ou servidor como mostra as imagens abaixo.

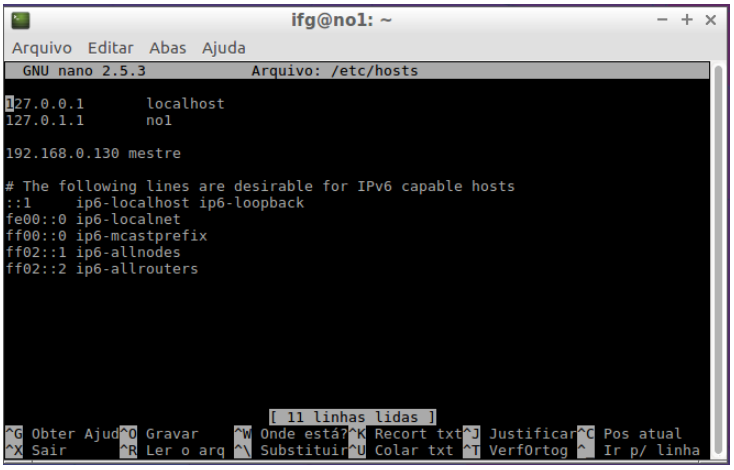
Com o arquivo editado para testar a conectividade entre nós e mestre basta digitar o seguinte comando:

Figura 8 – Configuração arquivo Hosts - Mestre



Fonte: Próprio autor.

Figura 9 – Configuração arquivo Hosts - nó1



Fonte: Próprio autor.

ifg@mestre: ping no1

Esse comando vai enviar pacotes ao host de destino e aguarda por uma resposta do host. Em sequência após testar edição do arquivo host, as máquinas estão preparadas para a instalação do SSH com o seguinte comando:

ifg@mestre: sudo apt-get install openssh-server

Com o término da instalação para verificar se o SSH está sendo executado basta executar o comando:

ifg@mestre: sudo systemctl status ssh

Se o status estiver como *active running* ele está sendo executado. Caso ele não estivesse deveria ser executado o comando:

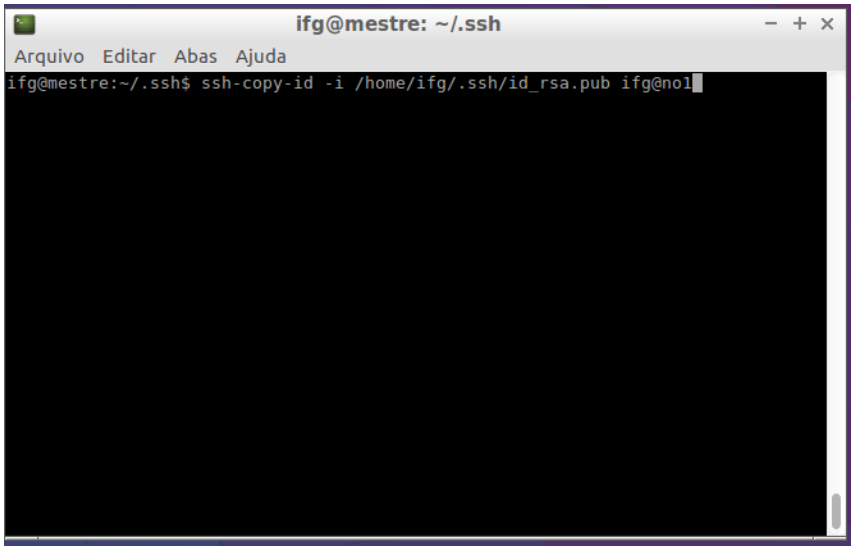
ifg@mestre: sudo systemctl enabled ssh

Com o SSH rodando em todas as máquinas foi digitado o seguinte comando nó mestre:

ifg@mestre: ssh-keygen

O comando gerará um par de chave, uma pública e outra privada, onde foi usada a chave pública para que o mestre tenha acesso aos nós sem que tenha que digitar a senha de cada nó toda vez que for fazer uma conexão remota. Esse procedimento vai ser útil

Figura 10 – Copiando e instalando a chave pública



Fonte: Próprio autor.

quando for executar os códigos em paralelo, para fazer a cópia da chave basta digitar no mestre o seguinte comando conforme a imagem abaixo.

Para verificar se foi feita a instalação da chave pública no nó1 basta testar a conexão do mestre nos nós do cluster com o seguinte comando.

ifg@mestre: ssh no1

Com a instalação e a configuração do SSH concluída o proximo passo é instalação e configuração do NFS, que fará o compartilhamento de arquivos em rede por meio de diretórios. A máquina de armazenamento desses diretórios será no mestre. Antes de começar a configuração do NFS foi preciso instalar os pacotes que serão necessários para a configuração do mesmo.

ifg@mestre: sudo apt-get install nfs-kernel-server portmap nfs-common

Com a instalação concluída em todos os nós foi preciso criar a pasta que será compartilhada em rede pelo nó mestre com o seguinte comando:

ifg@mestre: sudo mkdir Clusterdir

Os nós da rede terão acesso e permissões ao arquivos que estarão compartilhados no diretório Clusterdir. Com a criação do diretório é necessário configurar os arquivos de exportação do NFS por meio do comando:

ifg@mestre: sudo nano /etc/exports

Com ele, vai abrir o arquivo *exports* e inserir a seguinte linha conforme mostra a imagem abaixo.

Figura 11 – Configuração da pasta compartilhada

Fonte: Próprio autor.

A edição do arquivo segue o padrão NFS que define a pasta que será compartilhada *(/home/ifg/Clusterdir)* e depois, os computadores que poderão ter acesso a pasta sendo os IPs das máquinas que estiverem na faixa de *192.168.0.0/24*, e também permite usar a opção de controle *rw* que concede ao usuário fazer leitura e gravação dentro do diretório. O *no subtree check* é utilizado quando um subdiretório é exportado, com essa opção o NFS verifica o subdiretório que for exportado pelo cliente, *async* permite que o NFS faça transferências de arquivos de forma assíncrona sem precisar da resposta do cliente e com o comando *no root squash* aumenta a velocidade de transferência permitindo que os *roots* trabalham com privilégios de compartilhamento. Com a edição concluída é preciso reiniciar o serviço de NFS com o comando:

ifg@mestre: /etc/init.d/nfs-kernel-server restart

E para testar se a pasta esta distribuída basta o seguinte comando:

ifg@mestre: showmount -e 192.168.0.130

Com a pasta distribuída vamos criar o diretório Clusterdir nos escravos usando o mesmo comando:

ifg@no1: sudo mkdir Clusterdir

Para verificar se a pasta foi criada basta usar o comando.

ifg@no1: ls

Com a pasta criada e verificada é necessário usar dois comandos para montar o diretório NFS compartilhado com da máquina mestre. Esses comandos devem ser executados nos nós escravos:

ifg@no1: sudo mount -t nfs 192.168.0.130:home/ifg/Clusterdir /home/ifg/Clusterdir

ifg@no1: mount

O comando acima deve ser executado em seguida, lembrando que esses comandos devem ser digitados nos nós escravos:

ifg@no1: sudo mount -t nfs

O comando acima faz a montagem do diretório que foi distribuído para os nós escravos. O endereço de IP 192.168.0.130 é do servidor/mestre onde está a pasta origem do Clusterdir, o caminhado */home/ifg/Clusterdir* este é o diretório será onde será compartilhada o diretório. E para deixar mais fácil de fazer essa configuração de compartilhamento é utilizar o nome dos computadores sendo o mesmo. Para concluir a configuração da pasta compartilhada basta fazer a configuração no arquivo *fstab* que permitirá a montagem permanente do sistema de arquivo usando os comandos:

ifg@no1: sudo nano /etc/fstab

192.168.0.130:/home/ifg/Clusterdir /home/ifg/Clusterdir nfs rw,sync,hard,intr 0 0.

Neste comando, *192.168.0.130:/home/ifg/Clusterdir* indica o endereço IP do mestre e o diretório onde foi configurado a pasta compartilhada para ser disponibilizada em rede e em sequência */home/ifg/Clusterdir nfs rw,sync,hard,intr 0 0*, mostra o caminho configurado via NFS, com a opção de modificação (*rw*) que permite a leitura e gravação de forma sincronizada (*sync*) de forma que o usuário poderá interromper chamadas de sistemas que estão aguardando requisições (*hard* e *intr 0 0*). Com o arquivo *exports* no nó mestre e o *fstab* configurado em todos os nodes escravos e utilizando o comando nos escravo vai ser montado a pasta compartilhada entre os nodes.

ifg@no1: sudo mount -a

Com o SSH e NFS funcionando o último é a instalação e configuração da biblioteca de comunicação paralela que é o MPI, ela possui uma padronização para o para o paralelismo de memória distribuída além de fazer a otimização da comunicação com vários recursos de transmissão de mensagem e consequentemente terá um desempenho melhor. O download do MPI vai ser feito no site oficial **https://www.open-mpi.org/software/ompi/v4.1/** e a versão que será usada é a 4.1.2. Com o MPI baixado vamos para o próximo passa que é criar um diretório de instalação do MPI com o comando. [17]

ifg@mestre: mkdir $HOME/openmpi

O conduz a criação de pastas como o */openmpi* dentro do diretório pessoal. Próximo passo é copiar o arquivo baixado para esta pasta com o comando:

ifg@mestre: cp $HOME/Downloads/openmpi-4.1.2.tar.gz $HOME/openmpi

Com o pacote copiado ele deve ser descompactado com o seguinte comando:

ifg@mestre: tar -xzvf openmpi-4.1.2.tar.gz

Em seguida, é preciso que entrar no diretório descompactado com o comando:

[17] https://www.open-mpi.org/software/ompi/v4.1/

ifg@mestre: cd openmpi-4.1.2

Vamos instalar o *build-essential* é responsável pela criação e execução dos códigos que vão ser compilados no mestre.

ifg@mestre: sudo apt-get install build-essential

ifg@mestre: ./configure –prefix=$HOME/openmpi

Posteriormente são necessários os seguintes comandos:

ifg@mestre: make all

ifg@mestre: make install

O *make all* compila do software e executa uma série de rotinas para compilar um programa por meio de um código-fonte e o *make install* irá copiar o mesmo código e vai compilar junto com a sua documentação para um diretório específico. Agora vamos instalar mais dois pacotes sendo o pacote *openmpi-bin* para executar códigos em paralelo utilizando o comando *mpirun* e o *libopenmpi-dev* para desenvolver programas paralelos em MPI como mpicc. Usando os comandos

ifg@mestre: sudo apt-get install openmpi-bin

ifg@mestre: sudo apt-get install libopenmpi-dev

Com o término da instalação do restante dos comandos só serão executados no mestre daqui em diante. Os nós escravos já estão configurados para serem usados. Dando sequência, o comando *export* servirá para exportar variáveis de ambiente. *PATH* é o caminho que busca arquivos executáveis e através desse caminho o sistema operacional reconhece esses comandos diferentes. O *PATH* é armazenado em uma variável de ambiente *$PATH* digitando o comando:

ifg@mestre: export PATH=$PATH:$HOME/openmpi/bin

ifg@mestre: export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HOME/openmpi/lib.

O primeiro comando inclui a biblioteca de passagem de mensagem de alto desempenho binário *bin* na variável de ambiente $PATH. No segundo comando é inclui uma variável de ambiente *LD_LIBRARY_PATH* que informa aos programas do Linux a localização dessas bibliotecas compartilhadas por meio dos diretórios diferentes. A *lib* é uma biblioteca de passagem de mensagem de alto desempenho compartilhada. Para verificar se o MPI foi instalado com sucesso é só digitar:

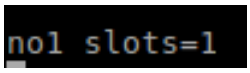
ifg@mestre: mpicc -v

A última etapa de configuração do MPI é informar os nodes que compõe o Cluster no servidor. Dentro do arquivo de configuração devemos incluir o node principal/servidor e os nodes escravos:

ifg@mestre: sudo nano .mpi_hostfile

Adicionando as linhas conforme a imagem abaixo.

Figura 12 – Arquivo .mpi_hostfile



Fonte: Próprio autor.

Na imagem acima mostra que o nó1 aloca 1 processador para executar um código que vai ser usado na sequência. Ao decorrer das configurações de outros nós, eles precisam ser adicionados no arquivo *mpi_hostfile* que especifica quantos processadores estarão disponíveis pra execução do código, o nó mestre também pode ser incluído nesta lista assim como foi feito no *Cluster Beowulf*. Cada host deve especificar o número de slots, ou seja, o número de processadores disponíveis e que vão ser usados por cada *host*. Nesse caso, foi utilizado 1 processador para execução de tarefas MPI, sendo 1 processador do nó1. Existem vários padrões para implementação de códigos paralelos, o MPI tem como padrão Fortran e C. A linguagem escolhida foi C, por ter vários outros trabalhos que usam códigos em C.

Essa última etapa explica como compilar um programa paralelo em C. O programa que vai ser compilado em paralelo deve estar dentro da pasta compartilhada, no caso *Clusterdir*, diretório que foi configurado pelo NFS. Então deve acessar a pasta Clusterdir pelo comando:

ifg@mestre: cd Clusterdir

Para compilar o código dentro da pasta foi usado o código:

ifg@mestre: mpic++ nomedoprograma.c++ -o nomedoprograma

E para testar se o executável foi criado é só acessar a pasta Clusterdir, e para verificar, o próximo passo é compilar o programa com o comando.

ifg@mestre: ./nomedoprograma

Esse código vai executar de forma paralela mas apenas no servidor, para a execução paralela em todos os nós é executado com o comando:

ifg@mestre: mpirun np 4 hostfile ../.mpi_hostfile ./nomedoprograma

Onde *mpirun* é um comando da biblioteca *bin* para executar em binário e *-np* é o número de hosts que serão executados. Nesse exemplo demonstrado o programa será executado no nó mestre mas o processador que será usado será o do nó1 como o parâmetro que foi inserido no arquivo *.mpi_hostfile*.

5 ANÁLISE DE RESULTADOS

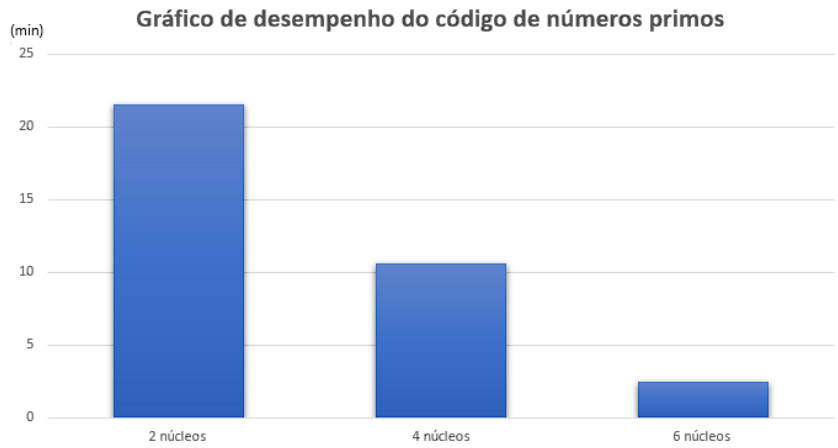
A análise de desempenho do Cluster Beowulf foi realizada através da execução de códigos paralelos com uso de MPI. Os algoritmos foram testados primeiro em sequencial, ou seja, sendo testada no nó mestre e depois em todo o Cluster constituído por três máquinas.

Os testes foram todos realizados no Cluster composto por três computadores Intel Core 2 Duo E8400 2 núcleos 3.0 GHz, 2 GB de memória principal e disco rígido de 40 GB, sendo uma máquinas com a função de nó mestre e os demais como nós escravos. Os dois algoritmos que foram testados seguiram o mesmo padrão, sendo sequencial para apenas uma máquina sendo ela o mestre e os nós 1 e 2 referem-se à execução em paralelo.

5.1 Teste código números primos

A primeira fase de testes utilizando o código paralelo, o código de números primos foi desenvolvido em linguagem C++, e pode ser encontrado na internet sob a licença GNU LGPL. O código realiza a contagem de números primos entre 1 e N, sendo N um inteiro maior que 1, e faz uso do MPI para realizar o cálculo em paralelo em um sistema que utiliza a memória distribuída. Para cada número inteiro i, é verificado se algum j menor o divide gerando resto igual a zero. A quantidade total de trabalho para um dado N é, portanto, aproximadamente proporcional a 1/2 * N $\hat{2}$.

Figura 13 – Gráfico de desempenho



Fonte: Próprio autor.

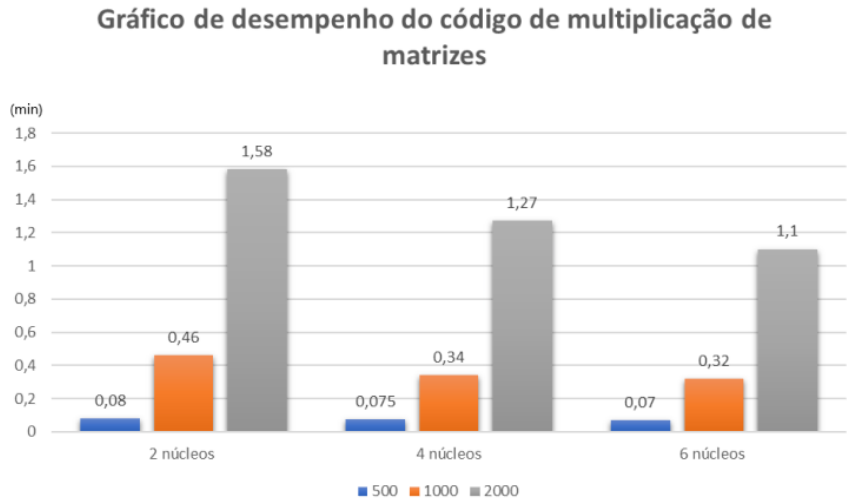
Conforme o gráfico acima podemos observar o desempenho do cluster calculando a quantidade números que tem entre 1 e 1 milhão que executa o código usando 2 núcleos com um tempo de 21 minutos e 51 segundos, ao decorrer que adicionamos mais 2 núcleos no cluster vemos a diminuição do tempo pela metade quando chegando ao tempo de execução de 10 minutos e 57 segundos usando 4 núcleos, com a adição dos últimos 2 núcleos chegando

a 6 núcleos de processamento o tempo obtido foi de 2 minutos e 5 segundos. Sendo assim, a cada adição de 2 núcleos o desempenho na execução do código dos números primos tem uma queda considerável no tempo considerando que tem que achar a quantidade de números entre 1 e 1 milhão.

5.2 Teste código matriz

A segunda fase de testes foi utilizado o código paralelo de multiplicação de matrizes, também utilizando a linguagem C++. Este código realiza a multiplicação de matrizes quadradas de tamanho n, como mostrado na imagem abaixo como n tendo o valor de 500, 1000 e 2000.

Figura 14 – Gráfico de desempenho



Fonte: Próprio autor.

O gráfico demonstra a execução do código sendo executado por 2, 4 e 6 núcleos com seus respectivos tempos sendo eles representados em minutos. Podemos notar que o tempo de execução da multiplicação de matrizes é diferente do tempo de execução do código dos números primos pelo fato simples fato do código de multiplicação de matrizes usar um tamanho significativo para exemplificar o paralelismo do código em execução. Mas apesar dos valores menores podemos notar um ganho de tempo com a execução com usando 4 e 6 núcleos.

6 CONCLUSÃO

O avanço tecnológico e o desenvolvimento de softwares que necessitam de um poder de hardware para serem executados têm gerado a produção de larga escala nas indústrias de computadores que por causa dessa necessidade teve que aumentar a produção, no entanto, ao decorrer da aquisição de computadores novos por parte de seus usuários as máquinas obsoletas estão sendo descartadas e isso tem gerado uma grande quantidade de REE por que elas ficaram obsoletas mediante a evolução dos hardwares e softwares causando o descarte de vários equipamentos que estão ótimo estado e em consequência ao descarte vem gerando o acúmulo de máquinas que ainda podem ser usadas mas para outras finalidades.

Esse trabalho teve como objetivos específicos, fazer uma melhor utilização de recurso público, aproveitar máquinas que estavam inutilizadas e assim evitando o descarte delas mas o principal objetivo era fazer a implementação de um cluster Beowulf que é formado por um aglomerado de computadores trabalhando juntos para executador um código em paralelo fazendo a divisão de tarefas de forma igualitária o cluster foi nomeado de Cluster Retail, pois a implementação foi feita com muita dificuldade pois não havia artigos para pesquisar sobre, então foi coletada toda informação que poderia ser usada na implementação do cluster, em alguns casos tive que entrar em contato com os autores via e-mail ou instagram para pedir ajuda de como poderia ser feito da melhor forma a implementação do cluster por isso foi dado o nome Retail, porque foi feito com varias partes de vários artigos que foi achado na Web. O público alvo do cluster Beowulf são professores e alunos que podem usar o alto desempenho do cluster para executar códigos que precisam de um poder computacional maior para serem executados no menor tempo possível e o cluster Beowulf é capaz de executar códigos em paralelo em poucos minutos. Apesar de ter não sido possível inserir um quantitativo mais expressivo de nós por estar dando um erro quando é cluster é submetido fazer a execução de algum código usando mais de 4 máquinas, o cluster Beowulf se mostrou eficiente perante os teste que o mesmo foi submetido. Mesmo usando apenas 3 máquinas contendo 2 núcleos de processamento cadas uma, os resultados foram satisfatórios pelo simples motivo que seria a velocidade dele de fazer cálculos. Por exemplo, fazer uma série de cálculos testando número por número entre 1 e mais de um milhão procurando a quantidade de números primos que contém nesse intervalo, obtendo o tempo de execução de 2 minutos e 5 segundos usando os 6 núcleos, executando o mesmo código com 2 núcleo foi preciso quase 22 minutos para concluir o teste, provando assim sua eficiência ao realizar cálculos que se fossem realizados em máquinas normais levariam mais tempo para serem executados.

Diante a implementação desse projeto foi possível aprofundar o conhecimento no funcionamento dos Sistemas Distribuídos, Sistema Operacional Linux, foi realizada

uma pesquisa relacionada ao uso do MPI que é a ferramenta que faz a paralelização e a distribuição da memória entre as máquinas, além dos estudos realizados para entender os protocolos SSH e NFS que foram usados na configuração do cluster Beowulf.

Foram encontrados diversos desafios no decorrer do trabalho como a adaptação de usar um ambiente Linux onde as dificuldades foram superadas através do decorrer da implementação do cluster, a falta de artigos para pesquisar a forma como seria feita a implementação e quais ferramentas usar para realizar a paralelização foi a parte mais difícil da implementação. Outra dificuldade foi de implementar as conexões via SSH e fazer o compartilhamento de umas pasta entre o mestre e os nós do cluster.

Como trabalhos futuros para o cluster Beowulf planeja-se:

- Resolver o problema que impede de executar qualquer código com mais de 4 máquinas ligadas ao cluster.
- Adicionar mais nós ao cluster.
- Realizar a implementação de uma ferramenta de monitoramento como o Zabbix ou Ganglia.
- Executar outros códigos paralelos para testar os desempenho do cluster Beowulf.

Referências

AFONSO, J. C. **Resíduos de Equipamentos Eletroeletrônicos: O Antropoceno Bate à Nossa Porta**. 5. ed. [S.l.]: Revista Virtual de Química, 2018. v. 10. ISSN 1984-6835. Citado na página 1.

COULOURIS, G. et al. **Sistemas Distribuídos: Conceitos e Projeto**. 5. ed. [S.l.]: Bookman, 2013. ISBN 978-85-8260-054-2. Citado 2 vezes nas páginas 5 e 7.

ELETRON, G. **Quais países produzem mais lixo eletrônico no mundo?**: Quais países geram mais lixo eletrônico no mundo? [S.l.], 2020. 1 p. Disponível em: <encurtador.com.br/puF45>. Acesso em: 14 de dezembro de 2021. Citado na página 1.

ELETRON, G. **Descarte indevido de lixo eletrônico é prejudicial para mulheres e crianças, segundo relatórios da OMS?**: Descarte indevido de lixo eletrônico é prejudicial para mulheres e crianças, segundo relatórios da oms? [S.l.], 2021. 1 p. Disponível em: <encurtador.com.br/fzQ08>. Acesso em: 10 de dezembro de 2021. Citado na página 2.

FUGI, D. M. **Montagem de Cluster Beowulf**: Artigos referente a montagem de um cluster. [S.l.], 2015. 1 p. Disponível em: <https://www.vivaolinux.com.br/artigo/Montagem-de-Cluster?pagina=1>. Acesso em: 25 de novembro de 2019. Citado 2 vezes nas páginas 2 e 3.

INTEL. **Processador Intel® Core™2 Duo E8400**: Especificações. [S.l.]. 46 p. Disponível em: <https://ark.intel.com/content/www/br/pt/ark/products/33910/intel-core2-duo-processor-e8400-6m-cache-3-00-ghz-1333-mhz-fsb.html>. Acesso em: 26 de abril de 2022. Citado na página 11.

MARQUES, P. **Conheça quais são os riscos do lixo eletrônico para a saúde**: Quanto lixo é consumido. [S.l.], 2018. 1 p. Disponível em: <https://noticias.r7.com/tecnologia-e-ciencia/conheca-quais-sao-os-riscos-do-lixo-eletronico-para-a-saude-13032018>. Acesso em: 09 de novembro de 2019. Citado na página 2.

PITANGA, M. **Construindo supercomputadores com Linux**. 3. ed. [S.l.]: Brasport, 2008. ISBN 978-85-7452-372-9. Citado na página 8.

TANENBAUM, A. S.; STEEN, M. V. **Sistemas Distribuídos: princípios e paradigmas**. 2. ed. [S.l.]: Sabrina Cairo, 2007. ISBN 978-85-7605-142-8. Citado na página 6.

UFSC. **O que são Resíduos Eletroeletrônicos?**: O que fazer com os resíduos eletroeletrônicos não patrimoniados gerados na ufsc ou pela comunidade? [S.l.], 2021. 1 p. Disponível em: <https://gestaoderesiduos.ufsc.br/residuos-eletroeletronicos/>. Acesso em: 10 de janeiro de 2022. Citado na página 1.

7 ANEXO - CÓDIGO DE NÚMEROS PRIMOS

Figura 15 – Código de números primos - Parte 1

ANEXO A – CÓDIGO PRIME_MPI

```
# include <cmath>
# include <cstdlib>
# include <ctime>
# include <iomanip>
# include <iostream>
# include <mpi.h>

using namespace std;

int main ( int argc, char *argv[] );
int prime_number ( int n, int id, int p );
void timestamp ( );

//****************************************************************************

int main ( int argc, char *argv[] )

//****************************************************************************
//
//  Purpose:
//
//    MAIN is the main program for PRIME_MPI.
//
//  Discussion:
//
//    This program calls a version of PRIME_NUMBER that includes
//    MPI calls for parallel processing.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    16 June 2016
//
//  Author:
//
//    John Burkardt
//
{
  int id;
  int ierr;
  int n;
  int n_factor;
  int n_hi;
  int n_lo;
```

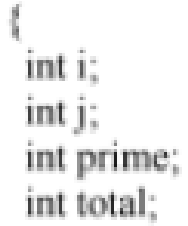
Figura 16 – Código de números primos - Parte 2

```
  int p;
  int primes;
  int primes_part;
  double wtime;

  n_lo = 1;
//  n_hi = 262144;
n_hi = 1048576;
n_factor = 2;
//
//  Initialize MPI.
//
  ierr = MPI_Init ( &argc, &argv );

  if ( ierr != 0 )
  {
    cout << "\n";
    cout << "PRIME_MPI - Fatal error!\n";
    cout << "  MPI_Init returned nonzero ierr.\n";
    exit ( 1 );
  }
//
//  Get the number of processes.
//
  ierr = MPI_Comm_size ( MPI_COMM_WORLD, &p );
//
//  Determine this processes's rank.
//
  ierr = MPI_Comm_rank ( MPI_COMM_WORLD, &id );

  if ( id == 0 )
  {
    timestamp ( );
    cout << "\n";
    cout << "PRIME_MPI\n";
    cout << "  C++/MPI version\n";
    cout << "\n";
    cout << "  An MPI example program to count the number of primes.\n";
    cout << "  The number of processes is " << p << "\n";
    cout << "\n";
    cout << "         N        Pi          Time\n";
    cout << "\n";
  }

  n = n_lo;

  while ( n <= n_hi )
  {
    if ( id == 0 )
    {
```

Figura 17 – Código de números primos - Parte 3

```
    wtime = MPI_Wtime ( );
  }
  ierr = MPI_Bcast ( &n, 1, MPI_INT, 0, MPI_COMM_WORLD );

  primes_part = prime_number ( n, id, p );

  ierr = MPI_Reduce ( &primes_part, &primes, 1, MPI_INT, MPI_SUM, 0,
    MPI_COMM_WORLD );

  if ( id == 0 )
  {
    wtime = MPI_Wtime ( ) - wtime;

    cout << "  " << setw(8) << n
         << "  " << setw(8) << primes
         << "  " << setw(14) << wtime << "\n";
  }
  n = n * n_factor;
 }
//
//  Terminate MPI.
//
 MPI_Finalize ( );
//
//  Terminate.
//
 if ( id == 0 )
 {
  cout << "\n";
  cout << "PRIME_MPI - Master process:\n";
  cout << "  Normal end of execution.\n";
  cout << "\n";
  timestamp ( );
 }

 return 0;
}
//****************************************************************************
**80

int prime_number ( int n, int id, int p )

//****************************************************************************
**80
//
//  Purpose:
//
//    PRIME_NUMBER returns the number of primes between 1 and N.
//
//  Discussion:
```

Figura 18 – Código de números primos - Parte 4



```
{
  int i;
  int j;
  int prime;
  int total;
```

Figura 19 – Código de números primos - Parte 5

```
  total = 0;

  for ( i = 2 + id; i <= n; i = i + p )
  {
    prime = 1;
    for ( j = 2; j < i; j++ )
    {
      if ( ( i % j ) == 0 )
      {
        prime = 0;
        break;
      }
    }
    total = total + prime;
  }
  return total;
}
//****************************************************************************

void timestamp ( )

//****************************************************************************
//
//  Purpose:
//
//    TIMESTAMP prints the current YMDHMS date as a time stamp.
//
//  Example:
//
//    31 May 2001 09:45:54 AM
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    24 September 2003
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    None
//
{
# define TIME_SIZE 40
```

8 ANEXO - CÓDIGO MULTIPLICAÇÃO DE MATRIZES

```
#include <time.h>
#include <stdbool.h>
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>
#include <mpi.h>
#include "mpi.h"
MPI_Status status;
double a[N][N],b[N][N],c[N][N];
main(int argc, char **argv)
{
int numtasks,taskid,numworkers,source,dest,rows,offset,i,j,k;
double t1, t2;
struct timeval start, stop;
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &taskid);
MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
numworkers = numtasks-1;
if (taskid == 0) {
t1 = MPI_Wtime();
for (i=0; i<N; i++) {
for (j=0; j<N; j++) {
a[i][j]= 1.0;
b[i][j]= 2.0;
}
}
rows = N/numworkers;
offset = 0;
for (dest=1; dest<=numworkers; dest++)
{
MPI_Send(&offset, 1, MPI_INT, dest, 1, MPI_COMM_WORLD);
MPI_Send(&rows, 1, MPI_INT, dest, 1, MPI_COMM_WORLD);
MPI_Send(&a[offset][0], rows*N, MPI_DOUBLE,dest,1, MPI_COMM_WORLD);
MPI_Send(&b, N*N, MPI_DOUBLE, dest, 1, MPI_COMM_WORLD);
offset = offset + rows;
}
```

```
for (i=1; i<=numworkers; i++)
{ source = i;
MPI_Recv(&offset, 1, MPI_INT, source, 2, MPI_COMM_WORLD, &status);
MPI_Recv(&rows, 1, MPI_INT, source, 2, MPI_COMM_WORLD, &status);
MPI_Recv(&c[offset][0], rows*N, MPI_DOUBLE, source, 2, MPI_COMM_WORLD,
&status);
}
t2 = MPI_Wtime();
printf("Here is the result matrix:\n");
for (i=0; i<N; i++) {
for (j=0; j<N; j++)
printf("%6.2f ", c[i][j]);
printf ("\n"); }printf("Elapsedtimeis
}
if (taskid > 0) {
source = 0;
MPI_Recv(&offset, 1, MPI_INT, source, 1, MPI_COMM_WORLD, &status);
MPI_Recv(&rows, 1, MPI_INT, source, 1, MPI_COMM_WORLD, &status);
MPI_Recv(&a, rows*N, MPI_DOUBLE, source, 1, MPI_COMM_WORLD,
&status);
MPI_Recv(&b, N*N, MPI_DOUBLE, source, 1, MPI_COMM_WORLD, &status);
for (k=0; k<N; k++)
for (i=0; i<rows; i++) {
c[i][k] = 0.0;
for (j=0; j<N; j++)
c[i][k] = c[i][k] + a[i][j] * b[j][k];
}
MPI_Send(&offset, 1, MPI_INT, 0, 2, MPI_COMM_WORLD);
MPI_Send(&rows, 1, MPI_INT, 0, 2, MPI_COMM_WORLD);
MPI_Send(&c, rows*N, MPI_DOUBLE, 0, 2, MPI_COMM_WORLD);
}
MPI_Finalize();
}
```