

Ferramentas de Clusterização Para o Ubuntu

Carlos Alberto Vaz de Moraes Junior

5 de julho de 2025

Sumário

1	Configurações Iniciais	9
1.1	Instalação do ubuntu via pendrive	9
1.1.1	Configuração do servidor, nodos e usuário administrador	9
1.2	Configuração da Rede Pública e Local	10
1.3	Instalação do servidor e cliente ssh	12
1.4	Configuração do arquivo /etc/hosts	12
1.5	Trocar o nome do host	14
1.5.1	Script para automatização	15
1.6	Acesso a rede externas nos nodos	15
1.6.1	Servidor	15
1.6.2	Nodos	15
2	Configuração do usuário admin	17
2.1	Instalação do servidor e cliente ssh	17
2.2	Instalação do sshpass	17
2.3	Uso de script para ssh sem senha admin nos nodos	17
2.4	Script para automatização	18
3	Instalação de pacotes	19
3.1	Instalação no servidor	19
3.2	Instalação nos nodos - rede local	19
4	Configuração da partição NFS	21
4.1	Instalação e configuração do servidor (CONTROLADOR)	21
4.2	Instalação e configuração dos nodos (CLIENTES)	21
4.2.1	Uso de scripts para os nodos	22
5	Configuração dos usuários	23
5.1	Criação dos usuários	23
5.2	Atualizar e criar novos usuários em lote	23
5.2.1	Sobre o comando newusers	23
5.2.2	Formato do Arquivo	23
5.2.3	Execução do Comando	24
5.3	Criando Usuários em Lote	24
5.4	Atualizando Senhas a partir de um Arquivo	24
5.4.1	Passo 1: Crie um Arquivo de Senhas	24

5.4.2	Passo 2: Liste os Usuários e Senhas	24
5.4.3	Passo 3: Salve e Saia	25
5.4.4	Passo 4: Verifique o Conteúdo do Arquivo	25
5.4.5	Passo 5: Atualize as Senhas	25
5.5	Adicionar usuários e configurar senhas em lote nos nós	25
5.6	Uso de script para ssh sem senha nos usuários dos nós	25
6	Configuração do gerenciamento de Filas - Pacote Slurm	27
6.1	MUNGE - Introdução	27
6.2	Nó Controlador	27
6.2.1	Instalação	27
6.2.2	Verificação	27
6.2.3	Geração da Chave	27
6.2.4	Permissões	28
6.2.5	Serviço	28
6.3	Nós clientes	28
6.3.1	Instalação	28
6.3.2	Cópia da Chave	28
6.3.3	Permissões	28
6.3.4	Serviço	28
6.3.5	Teste de Conexão	28
6.4	Solução de Problemas	29
6.5	Munge - Considerações Finais	29
6.6	Slurm - Introdução	29
6.7	Instalação Básica	29
6.8	Nó Controlador	29
6.8.1	Configuração do Arquivo slurm.conf	29
6.8.2	Criação do Arquivo de Configuração	32
6.8.3	Inicialização do Serviço	32
6.8.4	Verificação	32
6.9	Nós clientes	32
6.9.1	Configuração	32
6.9.2	Inicialização do Serviço	32
6.9.3	Verificação	32
6.10	Script para Configuração do slurm.conf no controlador e clientes	32
6.11	Solução de Problemas	33
6.12	Teste do Cluster	33
6.13	Slurm - Considerações Finais	33
7	Configurações extras	35
7.1	Desabilitar modo gráfico	35
7.2	Pacote para checagem de temperatura - lm-sensors	35
7.3	Script para checar a temperatura do Servidor e dos clientes	35
7.4	Configuração de funcionalidade de reencaminhar mensagens postmail	35

<i>SUMÁRIO</i>	5
8 Apêndice	37

Lista de Códigos Fonte:

8.1	Descoberta de ip dos nós e autoconfiguração do /etc/hosts	37
8.2	Atualização do hostname nos nós	42
8.3	Configuração de ssh Admin Sem Senha	43
8.4	Instalação de Pacotes em Lote	44
8.5	Adição NFS nos nós	47
8.6	Adição de usuários em Lote nos nós	49
8.7	Configuração ssh de usuários sem senha nos nós	50
8.8	Configuração Munge nos nós	51
8.9	Instalação e configuração do Slurm no controlador e nós	53
8.10	Configuração do slurm.conf no controlador e nós	54
8.11	Checar slurm no controlador e nós	56
8.12	Desabilitar modo gráfico nos nós	57
8.13	Configuração do lm-sensor nos nós	58
8.14	Script para monitoramento da temperatura do controlador e nós	59

Configuração do cluster ubuntu 2025.

Capítulo 1

Configurações Iniciais

1.1 Instalação do ubuntu via pendrive

Passos:

- Faça o download da ISO do Ubuntu 20.04 no endereço eletrônico <https://www.ubuntu.com/download/desktop>
- Utilize a ferramenta usb-creator-gtk para criar um pendrive bootável do sistema Ubuntu
- Após finalizado o processo, coloque o pendrive na máquina a ser instalada, ligue e selecione o Ubuntu Live na inicialização (via DEL, F2, F8, F12 dependendo da máquina).
- Siga com o processo de instalação a partir das etapas (pressionando continue para finalizar a etapa):
 - Instalar Ubuntu
 - Escolher teclado: BR
 - Tipos de Instalação: Normal
 - Apagar disco e instalar Ubuntu
 - Selecionar Zona de Horário
 - Fornecer as credenciais, a seguir.

1.1.1 Configuração do servidor, nodos e usuário administrador

No servidor, escolher:

- seu nome: administrador
- o nome do computador: cluster
- senha: utilize uma senha segura que desejar

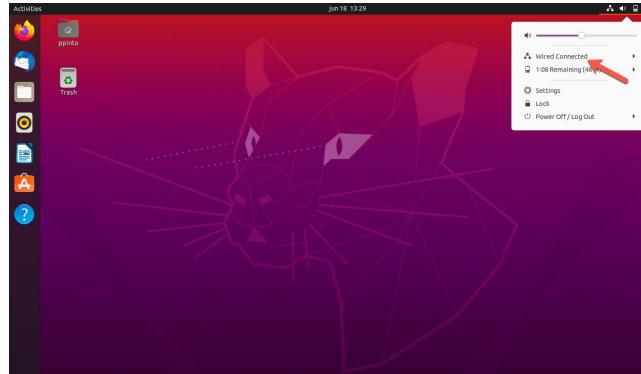
Nos nodos, escolher

- seu nome: administrador
- o nome do computador: nodo0X até nodoXX
- senha: utilize a MESMA senha definida no servidor

1.2 Configuração da Rede Pública e Local

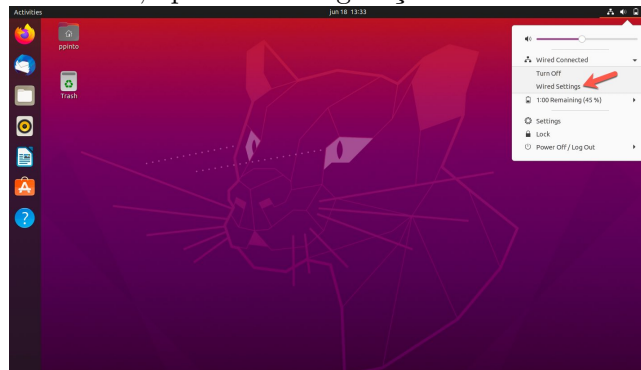
Para configuração do cluster, duas placas de rede devem estar disponíveis no computador servidor, uma para rede externa (internet), e outra disponível para rede local (intranet). Para mais ou menos cpus acrescentar ou retirar as linhas com os nodos. Na próxima etapa, deve-se definir um IP estático (para este guia a GUI/Área de Trabalho do Ubuntu foi utilizada). Aqui estão os passos:

Figura 1.1: Acesso a Configuração manual da internet e intranet via ambiente gráfico. Para configuração do cluster, duas conexões devem estar disponíveis: a de INTERNET e a INTRANET. A INTERNET usualmente usa uma faixa de IP começando com 200.X.X.X. Já a INTRANET pode ter uma faixa de IP começando 10.X.X.X.



Fonte: Internet, (2025)

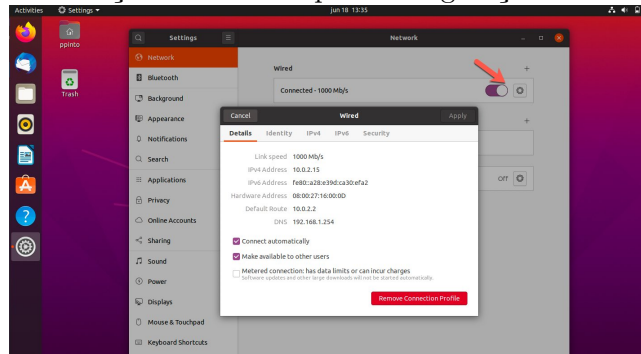
Figura 1.2: Escolha da conexão para configuração manual via ambiente gráfico. Tipicamente para duas placas, como no caso do servidor, serão observadas as configurações: **Conexão com Fio 1** e **Conexão com Fio 2**. Para os nós, apenas a configuração **Conexão com Fio 1** é observada.



Fonte: Internet, (2025)

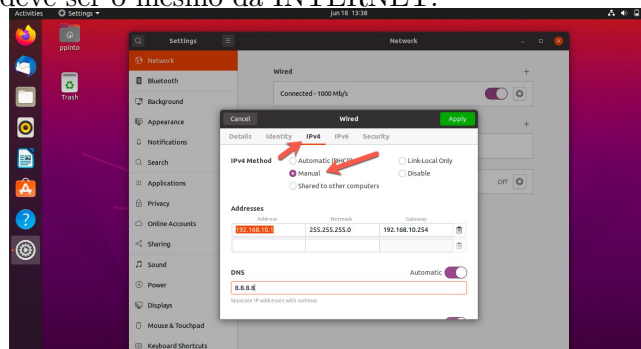
- Pesquise por configurações (servidor e nós) (Figura 1.1).
- Clique na aba Rede ou Wi-Fi, dependendo da interface que você gostaria de modificar (Figura 1.2).
- Para abrir as configurações da interface, clique no ícone de engrenagem ao lado do nome da interface (Figura 1.3).
- Selecione “Manual” na aba IPV4 e insira seu endereço IP estático, Máscara de rede e Gateway (Figura 1.4).

Figura 1.3: Menu de edição da conexão para configuração manual via engrenagem.



Fonte: Internet, (2025)

Figura 1.4: Janela onde parâmetros manuais são inseridos para configuração manual. Recomenda-se usar Address começando com 10.1.1.0, Netmask 255.255.255.0, Gateway 10.1.1.0 para a INTRANET. O DNS da INTRANET deve ser o mesmo da INTERNET.



Fonte: Internet, (2025)

- Clique no botão Aplicar (Figura 1.4).

Resumidamente

– SERVIDOR

- * Conexão com Fio 1: Endereço começando com 200.X.X.X, netmask 255.255.255.0, gateway: 200.X1.X1.X1, DNS: 200.X2,X2,X2. Obs: essas informações são obtidas com o provedor de internet.
- * Conexão com Fio 2: Endereço começando com 10.1.1.0, netmask 255.255.255.0, gateway: 10.1.1.0, DNS: 200.X2,X2,X2. Obs: o DNS é obtido com o provedor de internet.

– NÓS

- * Conexão com Fio 1: Endereço começando com 10.1.1.1 (nodo01) até 10.1.1.X (nodoX), netmask 255.255.255.0, gateway: 10.1.1.0, DNS: 200.X2,X2,X2. Obs: o dns é obtido com o provedor de internet.

1.3 Instalação do servidor e cliente ssh

Após configurar a internet, para garantir que a comunicação entre controlador e clientes, os seguintes pacotes devem ser instalados no servidor

prompt de terminal

```
administrador@cluster:~$ sudo apt-get install openssh-server openssh-client
```

Como ainda não existe configuração de internet nos nodos, baixe os pacotes

prompt de terminal

```
administrador@cluster:~$ sudo apt download openssh-server openssh-client openssh-sftp-server ssh
```

para instalação desses pacotes de forma offline. Utilize um pendrive para instalar os pacotes em cada um dos nodos com

prompt de terminal

```
administrador@cluster:~$ sudo dpkg -i openssh-* ssh*
```

1.4 Configuração do arquivo /etc/hosts

Finalmente, no servidor, deve-se configurar o arquivo

prompt de terminal:

```
administrador@cluster:~$ sudo nano /etc/hosts
```

com o conteúdo

127.0.0.1	localhost
127.0.1.1	cluster
10.1.1.0	nodo00
10.1.1.1	nodo01
10.1.1.2	nodo02
10.1.1.3	nodo03
10.1.1.4	nodo04
10.1.1.5	nodo05
10.1.1.6	nodo06
10.1.1.7	nodo07
10.1.1.8	nodo08
10.1.1.9	nodo09
10.1.1.10	nodo10
10.1.1.11	nodo11
10.1.1.12	nodo12

```
10.1.1.13      nodo13
```

```
# The following lines are desirable for IPv6 capable hosts
::1          ip6-localhost ip6-loopback
fe00::0      ip6-localnet
ff00::0      ip6-mcastprefix
ff02::1      ip6-allnodes
ff02::2      ip6-allrouters
```

No servidor, configura-se a rede interna com

```
administrador@cluster:~$ ifconfig
enp7s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 10.1.1.0  netmask 255.255.255.0  broadcast 10.1.1.255
    inet6 fe80::2e8f:98f3:ec44:da9b  prefixlen 64  scopeid 0x20<link>
    ether 00:1a:3f:c1:12:98  txqueuelen 1000  (Ethernet)
    RX packets 2167869  bytes 719150686 (719.1 MB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 2724010  bytes 2712002129 (2.7 GB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Para a rede externa, utilizar

```
administrador@cluster:~$ ifconfig
enp8s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 200.13X.11X.XX  netmask 255.255.255.XXX  broadcast 200.132.11X.XXX
    inet6 fe80::7014:8efb:1c20:28ad  prefixlen 64  scopeid 0x20<link>
    ether d4:5d:64:34:86:38  txqueuelen 1000  (Ethernet)
    RX packets 2551356  bytes 2215367005 (2.2 GB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 2269029  bytes 814171212 (814.1 MB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
administrador@cluster:~$ nmcli dev show | grep DNS
IP4.DNS[1]:                200.13X.11X.XXX
IP4.DNS[2]:                8.8.8.8
```

A configuração do ip dos nodos pode ser manual, na própria ferramenta de rede do ubuntu (na GUI/Área de Trabalho do Ubuntu), em que cada máquina assume um valor de ip no alcance 10.1.1.0 até 10.1.1.XXX, como a seguir:

```
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 10.1.1.1  netmask 255.255.255.0  broadcast 10.1.1.255
    inet6 fe80::f651:917e:148c:b69  prefixlen 64  scopeid 0x20<link>
    ether 10:7c:61:a5:a9:ef  txqueuelen 1000  (Ethernet)
    RX packets 642153  bytes 664423353 (664.4 MB)
```

```

RX errors 0   dropped 0   overruns 0   frame 0
TX packets 448908   bytes 184307516 (184.3 MB)
TX errors 0   dropped 0   overruns 0   carrier 0   collisions 0

```

É preciso realizar a configuração via gui em todos os nodos. A seguir, editar o arquivo hosts para adicionar o ip do cluster

prompt de terminal: editar hosts nos nodos

```
administrador@nodo01:~$ sudo nano /etc/hosts
```

com o conteúdo

```

127.0.0.1      localhost
127.0.1.1      nodo01
10.1.1.0       cluster

# The following lines are desirable for IPv6 capable hosts
::1           ip6-localhost ip6-loopback
fe00::0       ip6-localnet
ff00::0       ip6-mcastprefix
ff02::1       ip6-allnodes
ff02::2       ip6-allrouters

```

Terminada esta etapa, testar o acesso à rede externa com

```

administrador@cluster:~$ ping www.google.com
PING www.google.com (172.217.29.196) 56(84) bytes of data.
64 bytes de gru10s03-in-f4.1e100.net (172.217.29.196): icmp_seq=1 ttl=54 tempo=21.2 ms
64 bytes de gru10s03-in-f4.1e100.net (172.217.29.196): icmp_seq=2 ttl=54 tempo=21.2 ms

```

e a rede interna com

```

administrador@cluster:~$ ping nodo02
64 bytes de nodo02 (10.1.1.2): icmp_seq=1 ttl=64 tempo=1.13 ms
64 bytes de nodo02 (10.1.1.2): icmp_seq=2 ttl=64 tempo=1.16 ms

```

O ping indicará se as configurações foram corretamente implementadas.

Para automatizar a adição dos nós no arquivo /etc/hosts, utiliza-se a descoberta de IP via script [8.1](#).

1.5 Trocar o nome do host

prompt de terminal

```
administrador@cluster:~$ sudo hostnamectl set-hostname new-hostname
```

1.5.1 Script para automatização

Para atualizar o hostname em múltiplos nodos, utilizar o script 8.2. A consistência nos nomes de host é crucial para a organização do cluster. Este script é responsável por atualizar os hostnames de cada nó para um padrão uniforme (ex: nodo02, nodo03, etc.), o que facilita a identificação e o gerenciamento dentro do ambiente de cluster.

1.6 Acesso a rede externas nos nodos

1.6.1 Servidor

Para configurar o servidor, deve-se usar a sequência identificando a interface à internet (INTERNET_INTERFACE) e a interface local (LAN_INTERFACE). Em seguida, configurar o roteamento com:

```
INTERNET_INTERFACE=enp8s0
LAN_INTERFACE=enp7s0

sudo iptables -t nat -A POSTROUTING -o $INTERNET_INTERFACE -j MASQUERADE
sudo iptables -A FORWARD -i $LAN_INTERFACE -o $INTERNET_INTERFACE -m \
state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i $INTERNET_INTERFACE -o $LAN_INTERFACE -j ACCEPT

sudo sysctl -w net.ipv4.ip_forward=1
```

OBS: Esta configuração é perdida se o servidor é desligado.

1.6.2 Nodos

Para configuração de acesso externo à internet pelos nodos, duas etapas devem ser realizadas.

Configuração do DNS

Identifica-se o DNS do servidor, injetando este valor na configuração dos nodos com:

```
nmcli -g NAME connection show | grep "Conexao_cabeada" | \
while read CONN_NAME; do
sudo nmcli connection modify "$CONN_NAME" ipv4.dns "XXX.XXX.XXX.XXX"
done
```

OBS: Para descobrir as informações acerca do nome da conexão, alterando o grep "Conexão cabeada" conforme a necessidade e também o DNS, digite apenas nmcli (no nodo e servidor).

Configuração do Cliente

Para ativar o roteamento no nodo, deve-se executar o comando

```
sudo ip route add default via 10.1.1.1
sudo systemctl restart NetworkManager.service
```

OBS: o comando deve ser executado em cada nó para a conexão ser ativada. Esta configuração é perdida se o nó é desligado.

Capítulo 2

Configuração do usuário admin

2.1 Instalação do servidor e cliente ssh

Para garantir que a comunicação entre controlador e clientes, os seguintes pacotes devem ser instalados

prompt de terminal

```
administrador@cluster:~$ sudo apt-get install openssh-server openssh-client
```

Vale salientar que estes pacotes (principalmente o openssh-server) devem estar disponíveis nos nós para que a execução dos scripts em lote possa ocorrer corretamente. Neste caso, a instalação o openssh-server **deve** ser realizada no momento da instalação do Ubuntu nos nós.

2.2 Instalação do sshpass

A instalação do pacote sshpass facilitará a instalação da configuração dos nodos, permitindo a execução do ssh com senha no terminal. Para instalar, digitar no terminal

prompt de terminal

```
administrador@cluster:~$ sudo apt-get install sshpass
```

2.3 Uso de script para ssh sem senha admin nos nodos

Use ssh-keygen para gerar um par de chaves consistindo de uma chave pública e uma chave privada no computador cliente. Este comando pode ser executado em qualquer distribuição de cliente Linux moderna, via terminal.

prompt de terminal: Gerando as chaves

```
administrador@cluster:~$ ssh-keygen -t rsa -N -f ~/.ssh/id_rsa <<< y
```

Carregue sua chave pública para o cliente Linux. Em um cliente Linux, use ssh-copy-id para propagar a chave pública para o servidor, assim:

prompt de terminal: Transfira a chave pública para o servidor remoto

```
administrador@cluster:~$ ssh-copy-id administrador@nodo01
```

Certifique-se de substituir user por um nome de usuário válido do servidor e somedomain pelo IP ou domínio válido do servidor.

2.4 Script para automatização

Para automatizar a configuração do admin sem senha nos nós e controlador, utilize o script [8.3](#).

Capítulo 3

Instalação de pacotes

A instalação de pacotes segue a estrutura convencional de instalação de pacotes do Ubuntu, via utilização do apt.

3.1 Instalação no servidor

Aqui, segue uma lista de pacotes essenciais para a execução dos cálculos numéricos e monitoramento do Cluster.

- openssh-server, openssh-client e sshpass: para conexão remotas (conforme mencionado anteriormente, estes pacotes já devem estar instalados logo após a configuração da rede local) (OBRIGATÓRIO).
- nfs-kernel-server e nfs-common: servidor para montagem NFS e cliente para os nós (OBRIGATÓRIO).
- gcc: compilador para C, disponível no pacote build-essential (OBRIGATÓRIO).
- gfortran: compilador para Fortran, disponível no pacote gfortran (OBRIGATÓRIO).
- openmpi: para execução do código em paralelo (OBRIGATÓRIO).
- munge: pacote suporte para automatização de distribuição de cálculos, disponível nos pacotes munge libmunge2 libmunge-dev (OBRIGATÓRIO).
- slurm: pacote gerenciador de filas para automatização de distribuição de cálculos, disponível no pacote slurm-wlm (OBRIGATÓRIO).
- intel-oneapi-base-toolkit: pacote com compiladores ifx e icx e mpi (paralelo) para cálculo numérico e simulações (OPCIONAL).
- Postfix: Utilização de relé postfix para comunicação dos status do cluster e filas Slurm via e-mail no pacote postfix (OPCIONAL).

3.2 Instalação nos nodos - rede local

Após a implementação da configuração vista na seção 1.6, é possível baixar os aplicativos via apt nos nodos. Para a instalação de programas em modo lote, utilizar o script 8.4.

Capítulo 4

Configuração da partição NFS

4.1 Instalação e configuração do servidor (CONTROLADOR)

prompt de terminal: INSTALAÇÃO

```
administrador@cluster:~$ sudo apt-get install nfs-kernel-server
```

Após editar o arquivo

prompt de terminal: edição do arquivo exports

```
administrador@cluster:~$ sudo nano /etc/exports
```

Adicionando o conteúdo a seguir, dependendo do número de nodos.

```
/home 10.1.1.1(rw,sync,no_root_squash,no_subtree_check)
/home 10.1.1.2(rw,sync,no_root_squash,no_subtree_check)
/home 10.1.1.3(rw,sync,no_root_squash,no_subtree_check)
/home 10.1.1.4(rw,sync,no_root_squash,no_subtree_check)
...
/home 10.1.1.xx(rw,sync,no_root_squash,no_subtree_check)
```

prompt de terminal: execução do servidor nfs

```
administrador@cluster:~$ sudo exportfs -a
administrador@cluster:~$ sudo systemctl restart nfs-kernel-server
```

4.2 Instalação e configuração dos nodos (CLIENTES)

prompt de terminal: instalação dos clientes

```
administrador@cluster:~$ sudo apt-get install nfs-common
```

Para instalar os programas em lote, utilize o script [3.2](#). Após editar o arquivo

prompt de terminal: edição do arquivo exports

```
administrador@cluster:~$ sudo nano /etc/fstab
```

acrescentando no seguinte conteúdo:

```
10.1.1.1:/home /home nfs auto,noatime,nolock,bg,intr,tcp,actimeo=1800 0 0
```

Para finalizar

prompt de terminal: instalação dos clientes

```
administrador@cluster:~$ sudo mount -a
```

ou

```
echo senha | sudo -S mount -a
```

para entrada automática de senha.

4.2.1 Uso de scripts para os nodos

No script 8.5, a linha

```
10.1.1.1:/home /home nfs auto,noatime,nolock,bg,intr,tcp,actimeo=1800 0 0
```

é automaticamente adicionada em cada nó, sendo o nfs recarregado em todos os nós após a edição do arquivo `/etc/fstab`. Caso a linha exista no arquivo `fstab`, a adição é pulada.

Capítulo 5

Configuração dos usuários

5.1 Criação dos usuários

5.2 Atualizar e criar novos usuários em lote

O comando `newusers` lê um arquivo de pares de nome de usuário e senha em texto simples e usa essas informações para:

- Atualizar um grupo de usuários existentes
- Criar novos usuários

Cada linha está no mesmo formato do arquivo de senha padrão.

5.2.1 Sobre o comando `newusers`

- Destinado a ambientes de sistema grandes
- Permite atualizar muitas contas simultaneamente (modo em lote)
- Como os dados são armazenados em texto simples:
 - Garanta que apenas o `root` tenha acesso de leitura/escrita
 - Use o comando `chmod` para proteger o arquivo:

```
chmod 600 /root/batch-user-add.txt
```

5.2.2 Formato do Arquivo

Cada linha deve seguir o formato:

```
username:password:UID:GID:user_info:home_directory:shell
```

Exemplo:

```
usuario1:senha123:1001:1001:Usuario 1:/home/usuario1:/bin/bash
usuario2:senha456:1002:1002:Usuario 2:/home/usuario2:/bin/bash
```

5.2.3 Execução do Comando

Para processar o arquivo:

```
sudo newusers /root/batch-user-add.txt
```

prompt de terminal: newusers

```
administrador@cluster:~$ touch /root/batch-user-add.txt
administrador@cluster:~$ chmod 0600 /root/batch-user-add.txt
```

Crie uma lista de usuários como segue. Primeiramente, abra o arquivo:

prompt de terminal: newusers

```
administrador@cluster:~$ nano /root/batch-user-add.txt
```

Acrescente username e password:

```
user1:password:1001:513:Student Account:/home/user1:/bin/bash
user2:password:1002:513:Sales user:/home/user2:/bin/bash
user100:password:1100:513:Sales user:/home/user100:/bin/bash
tom:password:1110:501:Guest Account:/home/guest:/bin/menu
jerry:password:1120:501:Guest Account:/home/guest:/bin/menu
```

5.3 Criando Usuários em Lote

Para criar usuários em lote, use o comando `newusers` com um arquivo contendo os detalhes dos usuários:

```
# newusers /root/batch-user-add.txt
```

5.4 Atualizando Senhas a partir de um Arquivo

Outra forma de atualizar senhas em massa com `chpasswd` é criar um arquivo com os nomes de usuário e as novas senhas. O comando lê os dados do arquivo, e não da entrada padrão.

Para atualizar as senhas dessa forma, siga os passos:

5.4.1 Passo 1: Crie um Arquivo de Senhas

Abra o `vim` para criar um arquivo chamado `mypasswords.txt`:

```
vim mypasswords.txt
```

5.4.2 Passo 2: Liste os Usuários e Senhas

Dentro do arquivo, liste os usuários e suas respectivas senhas no formato:

```
pnapuser1:newpassword01
pnapuser2:newpassword02
pnapuser3:newpassword03
```

5.4.3 Passo 3: Salve e Saia

Salve o arquivo e saia do vim.

5.4.4 Passo 4: Verifique o Conteúdo do Arquivo

Verifique o conteúdo do arquivo usando `cat`:

```
cat mypasswords.txt
```

5.4.5 Passo 5: Atualize as Senhas

Execute `chpasswd` redirecionando a entrada do arquivo:

```
sudo chpasswd < mypasswords.txt
```

O comando será executado silenciosamente sem exibir nenhuma saída.

5.5 Adicionar usuários e configurar senhas em lote nos nós

Segue script [8.6](#) para adicionar usuários em lote em diversos nós. Essencial para a criação e manutenção de ambientes multiusuário. Este script permite adicionar múltiplos usuários em todos os nós do cluster em um processo de lote. Ele utiliza arquivos (`new_users.txt` e `newpass.txt`, disponíveis via NFS) para criar as contas e definir suas senhas. De forma inteligente, ele pula a criação de usuários que já existem e garante que as senhas sejam definidas apenas para contas válidas.

5.6 Uso de script para ssh sem senha nos usuários dos nós

Segue script [8.7](#) para configurar ssh sem senha em lote dos usuários em diversos nós. Complementando o acesso administrativo, este script configura o acesso SSH sem senha para usuários comuns entre o nó de controle e os nós de computação. Isso é vital para que os usuários possam submeter e monitorar seus trabalhos nos nós sem a necessidade de autenticação por senha para cada conexão.

Capítulo 6

Configuração do gerenciamento de Filas - Pacote Slurm

O pacote Slurm é o sistema robusto de gerenciamento de filas para execução de cálculos computacionais. Munge é um serviço de autenticação essencial para o Slurm. Este script se encarrega de instalar e configurar o serviço Munge nos nós do cluster. Ele garante que os componentes do Slurm possam se comunicar de forma segura e autenticada. Para sua implementação no cluster, devem ser configurados o pacote Munge e o Slurm, os quais serão discutidos a seguir. Todos os comandos foram implementados via utilização do pacote disponível no apt do sistema Ubuntu 20.04.

6.1 MUNGE - Introdução

O Munge é um sistema de autenticação utilizado pelo Slurm para comunicação segura entre os nós do cluster. Este documento apresenta a configuração passo a passo para nós controlador e worker.

6.2 Nó Controlador

6.2.1 Instalação

Execute os seguintes comandos para instalar os pacotes necessários:

```
sudo apt update
sudo apt install munge libmunge2 libmunge-dev
```

6.2.2 Verificação

Teste a instalação com:

```
munge -n | unmunge | grep STATUS
```

Saída esperada: STATUS: SUCCESS

6.2.3 Geração da Chave

Se necessário, gere a chave manualmente:

```
sudo /usr/sbin/mungekey
```

6.2.4 Permissões

Configure as permissões corretamente:

```

sudo chown -R munge: /etc/munge/ /var/log/munge/ /var/lib/munge/ /run/munge/
sudo chmod 0700 /etc/munge/ /var/log/munge/ /var/lib/munge/
sudo chmod 0755 /run/munge/
sudo chmod 0700 /etc/munge/munge.key
sudo chown -R munge: /etc/munge/munge.key

```

6.2.5 Serviço

Habilite e inicie o serviço:

```

sudo systemctl enable munge
sudo systemctl restart munge

```

6.3 Nós clientes

6.3.1 Instalação

Instale os mesmos pacotes do controlador:

```

sudo apt install munge libmunge2 libmunge-dev

```

6.3.2 Cópia da Chave

Copie o arquivo `/etc/munge/munge.key` do controlador para cada nó cliente.

6.3.3 Permissões

Configure as permissões (mesmo comando do controlador):

```

sudo chown -R munge: /etc/munge/ /var/log/munge/ /var/lib/munge/ /run/munge/
sudo chmod 0700 /etc/munge/ /var/log/munge/ /var/lib/munge/
sudo chmod 0755 /run/munge/
sudo chmod 0700 /etc/munge/munge.key
sudo chown -R munge: /etc/munge/munge.key

```

6.3.4 Serviço

Habilite e inicie o serviço:

```

sudo systemctl enable munge
sudo systemctl restart munge

```

6.3.5 Teste de Conexão

Verifique a comunicação com o controlador:

```

munge -n | ssh <controlador> unmunge

```

6.4 Solução de Problemas

Verifique o status do serviço:

```
systemctl status munge
```

Consulte os logs:

```
sudo journalctl -u munge -f  
sudo cat /var/log/munge/munged.log
```

6.5 Munge - Considerações Finais

- A chave Munge deve ser **idêntica** em todos os nós
- O usuário `munge` é criado automaticamente - não crie manualmente
- Verifique sempre as permissões dos arquivos

A seguir, é apresentado o script 8.8 para automatização da instalação do MUNGE nos nós

6.6 Slurm - Introdução

O Slurm é um sistema de gerenciamento de cluster e fila de jobs. Este documento apresenta a configuração para nós controlador e trabalhadores. O script 8.9 é o passo para transformar seus nós em parte de um cluster de computação de alto desempenho. Ele instala o Slurm Workload Manager (`slurm-wlm`), que é o sistema de gerenciamento de filas e recursos responsável por agendar e distribuir as tarefas de computação entre os nós.

6.7 Instalação Básica

Em todos os nós (controlador e clientes), instale o pacote necessário:

```
sudo apt update  
sudo apt install slurm-wlm
```

A seguir, o script 8.9 para instalação do pacote Slurm no controlador e nós é apresentado.

6.8 Nó Controlador

6.8.1 Configuração do Arquivo `slurm.conf`

Utilize o gerador de configuração do Slurm:

```
firefox /usr/share/doc/slurmctld/slurm-wlm-configurator.html
```

Preencha os campos essenciais:

- `ClusterName`: NomeDoSeuCluster
- `SlurmctldHost`: NomeDoControlador

- `NodeName`: NomeDoWorker[1-4] (para 4 nós worker)
- Configure `CPUs`, `Sockets`, `CoresPerSocket` e `ThreadsPerCore` conforme saída do `lscpu`
- `ProctrackType`: `LinuxProc`

Segue exemplo de configuração do arquivo `slurm.conf`:

```
# slurm.conf file generated by configurator easy.html.
# Put this file on all nodes of your cluster.
# See the slurm.conf man page for more information.
#
SlurmctldHost=cluster
#
#MailProg=/bin/mail
MpiDefault=none
#MpiParams=ports=-#
ProctrackType=proctrack/cgroup
ReturnToService=2
SlurmctldPidFile=/run/slurmctld.pid
#SlurmctldPort=6817
SlurmdPidFile=/run/slurmd.pid
#SlurmdPort=6818
SlurmdSpoolDir=/var/lib/slurm-llnl/slurmd
SlurmUser=slurm
#SlurmdUser=root
StateSaveLocation=/var/lib/slurm-llnl/slurmctld
SwitchType=switch/none
TaskPlugin=task/affinity
#
#
# TIMERS
#KillWait=30
#MinJobAge=300
#SlurmctldTimeout=120
#SlurmdTimeout=300
#
#
# SCHEDULING
FastSchedule=1
SchedulerType=sched/backfill
SelectType=select/cons_res
SelectTypeParameters=CR_Core
#
#
# LOGGING AND ACCOUNTING
AccountingStorageType=accounting_storage/none
```

```

#AccountingStorageType=accounting_storage/slurmdbd
ClusterName=cluster
#JobAcctGatherFrequency=30
JobAcctGatherType=jobacct_gather/none
#SlurmctldDebug=info
SlurmctldLogFile=/var/log/slurm-llnl/slurmctld.log
#SlurmdDebug=info
SlurmdLogFile=/var/log/slurm-llnl/slurmd.log
#
#
# COMPUTE NODES
#NodeName=nodo01 State=UNKNOWN

NodeName=nodo02 CPUs=32 Boards=1 SocketsPerBoard=1 CoresPerSocket=16
ThreadsPerCore=2 State=UNKNOWN
NodeName=nodo03 CPUs=32 Boards=1 SocketsPerBoard=1 CoresPerSocket=16
ThreadsPerCore=2 State=UNKNOWN
NodeName=nodo04 CPUs=32 Boards=1 SocketsPerBoard=1 CoresPerSocket=16
ThreadsPerCore=2 State=UNKNOWN
NodeName=nodo05 CPUs=8 Boards=1 SocketsPerBoard=1 CoresPerSocket=4
ThreadsPerCore=2 State=UNKNOWN
NodeName=nodo06 CPUs=8 Boards=1 SocketsPerBoard=1 CoresPerSocket=4
ThreadsPerCore=2 State=UNKNOWN
NodeName=nodo07 CPUs=8 Boards=1 SocketsPerBoard=1 CoresPerSocket=4
ThreadsPerCore=2 State=UNKNOWN
NodeName=nodo08 CPUs=8 Boards=1 SocketsPerBoard=1 CoresPerSocket=4
ThreadsPerCore=2 State=UNKNOWN
NodeName=nodo09 CPUs=8 Boards=1 SocketsPerBoard=1 CoresPerSocket=4
ThreadsPerCore=2 State=UNKNOWN
NodeName=nodo10 CPUs=8 Boards=1 SocketsPerBoard=1 CoresPerSocket=4
ThreadsPerCore=2 State=UNKNOWN
NodeName=nodo11 CPUs=8 Boards=1 SocketsPerBoard=1 CoresPerSocket=4
ThreadsPerCore=2 State=UNKNOWN
NodeName=nodo12 CPUs=8 Boards=1 SocketsPerBoard=1 CoresPerSocket=4
ThreadsPerCore=2 State=UNKNOWN
NodeName=nodo13 CPUs=8 Boards=1 SocketsPerBoard=1 CoresPerSocket=4
ThreadsPerCore=2 State=UNKNOWN

#filas

PartitionName=geral Nodes=nodo[02-11] Default=YES MaxTime=INFINITE State=UP
PartitionName=mcest Nodes=nodo[12-13] MaxTime=INFINITE State=UP

```

6.8.2 Criação do Arquivo de Configuração

Copie o texto gerado para:

```
sudo nano /etc/slurm-llnl/slurm.conf
```

6.8.3 Inicialização do Serviço

```
sudo systemctl enable slurmctld
sudo systemctl restart slurmctld
```

6.8.4 Verificação

```
systemctl status slurmctld    # Verifica status do servico
sinfo                          # Mostra informacoes do cluster
srun hostname                  # Lista todos os nos disponiveis
```

6.9 Nós clientes

6.9.1 Configuração

Copie o arquivo `slurm.conf` do controlador para cada nó trabalhador:

```
sudo nano /etc/slurm-llnl/slurm.conf
```

Adicionalmente, a pasta nos nós

```
/etc/slurm-wln
```

deve conter o arquivo `cgroup.conf`. Neste arquivo, o seguinte conteúdo deve ser adicionado:

```
CgroupMountpoint=/sys/fs/cgroup
```

6.9.2 Inicialização do Serviço

```
sudo systemctl enable slurmd
sudo systemctl restart slurmd
```

6.9.3 Verificação

```
systemctl status slurmd
```

6.10 Script para Configuração do `slurm.conf` no controlador e clientes

Como o arquivo `slurm.conf` deve ser idêntico em todas as máquinas, é conveniente automatizar o processo de cópia do arquivo via script [8.10](#).

6.11 Solução de Problemas

Verifique os logs em caso de erros:

```
sudo cat /var/log/slurm-llnl/slurmd.log # Para nos worker
sudo cat /var/log/slurm-llnl/slurmctld.log # Para controlador
```

6.12 Teste do Cluster

No nó controlador, execute:

```
sinfo # Mostra estado dos nos
srun -N4 hostname # Executa em 4 nos (ajuste conforme necessario)
```

Para verificar o status dos clientes em lote, utilize o script 8.11.

6.13 Slurm - Considerações Finais

- O arquivo `slurm.conf` deve ser **idêntico** em todos os nós
- Verifique sempre os logs após reiniciar serviços
- A configuração de hardware (CPUs, cores, etc.) deve refletir sua infraestrutura real.

Capítulo 7

Configurações extras

Neste capítulo, algumas otimizações podem ser aplicadas.

7.1 Desabilitar modo gráfico

A partir da desativação do modo gráfico, mais recursos computacionais ficam disponíveis para os cálculos numéricos. A desativação do modo gráfico pode ser feita em lote via script [8.12](#).

OBS: a definição do modo multi-user pode modificar a habilidade do sistema de inicializar tarefas de rede, como subir placas de rede. Em situações em que o sistema não reconhece placas de rede com drivers compilados, favor reinstalar os drivers. Adicionalmente, utilize telinit 3 para entrar no modo multi-user com modo texto, internet e nfs.

7.2 Pacote para checagem de temperatura - lm-sensors

prompt de terminal: instalação dos clientes

```
administrador@cluster:~$ sudo apt-get install lm-sensors
```

A instalação do aplicativo em modo lote pode ser feita a partir do script visto na seção [3.2](#). A implementação do lm-sensors nos nós em modo lote pode ser feita com o script [8.13](#).

7.3 Script para checar a temperatura do Servidor e dos clientes

O script [8.14](#) apresenta os dados de CPU, temperatura e execução em formato de coluna. Duas rotinas foram implementadas: uma para máquinas AMD e outra para máquinas INTEL.

7.4 Configuração de funcionalidade de reencaminhar mensagens post-mail

Crie o arquivo `/etc/postfix/sasl_passwd` no seu editor de texto preferido e preencha com o seguinte em uma linha:

```
smtp.gmail.com UsuárioGmail:SenhaGmail
```

Caso utilize o e-mail institucional, o servidor smtp da UFPel está disponível na página <https://wp.ufpel.edu.br/cti/servicos/email/>. Em seguida, precisaremos criar o hash desse arquivo para torná-lo mais seguro. Primeiro, certifique-se de que `/etc/postfix` seja propriedade do usuário `postfix`, ou você receberá uma mensagem de erro informando `postmap: fatal: open database /etc/postfix/sasl_passwd.db: Permission denied`. Após editar o arquivo, execute a sequência

```
sudo chown postfix /etc/postfix
postmap hash:/etc/postfix/sasl_passwd
```

Os arquivos `/etc/postfix/sasl_passwd` e `/etc/postfix/sasl_passwd.db` continuarão existindo no seu computador mesmo após a criação do hash. Certifique-se de que ambos sejam propriedade do usuário e grupo `root`. Defina as permissões dos arquivos para 640 em ambos.

Após, edite o arquivo de configuração `/etc/postfix/main.cf` com

prompt de terminal: instalação dos clientes

```
administrador@cluster:~$ sudo nano /etc/postfix/main.cf
```

e adicione as seguintes linhas no final:

```
relayhost = [smtps.ufpel.edu.br]:587 #exemplo

smtp_use_tls = yes
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_sasl_security_options = noanonymous

smtp_generic_maps = hash:/etc/postfix/generic

compatibility_level = 2
inet_interfaces = all
```

Para concluir a instalação, execute o comando

prompt de terminal

```
administrador@cluster:~$ sudo systemctl restart postfix
```

Capítulo 8

Apêndice

Algoritmo 8.1: Descoberta de ip dos nós e autoconfiguração do /etc/hosts

```
#!/bin/bash

# Enhanced with error handling and progress reporting
set -euo pipefail # Exit immediately if a command exits
                  # with a non-zero status,
                  # exit if an undeclared variable is
                  # used, and propagate pipe errors.

# Check for root privileges
if [ "$EUID" -ne 0 ]; then
    echo "Please run as root (required for /etc/hosts
    and /etc/exports modification)" >&2
    exit 1
fi

# Validate arguments and set defaults
if [ $# -lt 2 ]; then
    echo "Usage: $0 [<network_prefix>] <start_ip> <end_ip>" >&2
    echo "Example: $0 10.1.1. 1 100" >&2
    echo "Example (using default prefix 10.1.1.): $0 1 100" >&2
    exit 1
fi

# Determine prefix, start, and end based on number of arguments
if [ $# -eq 3 ]; then
    prefix="$1"
    start="$2"
    end="$3"
elif [ $# -eq 2 ]; then
    prefix="10.1.1." # Default network prefix
    start="$1"
```

```

end="$2"
fi

timestamp=$(date +%Y%m%d_%H%M%S)

# --- Variables and Flags for NFS Export Configuration ---
# Define the base NFS export path. This should be your shared directory.
NFS_EXPORT_PATH="/home"
# Flag to track if any NFS export lines were added/modified
NFS_EXPORTS_CHANGED=0

# Backup original /etc/hosts file with error checking
backup_hosts_file="/etc/hosts.backup_${timestamp}"
if ! cp /etc/hosts "$backup_hosts_file"; then
    echo "Failed to create /etc/hosts backup! Exiting." >&2
    exit 1
fi
echo "Created hosts backup: $backup_hosts_file"

# Backup original /etc/exports file with error checking
backup_exports_file="/etc/exports.backup_${timestamp}"
if ! cp /etc/exports "$backup_exports_file"; then
    echo "Failed to create /etc/exports backup! Exiting." >&2
    exit 1
fi
echo "Created exports backup: $backup_exports_file"

# Function to update /etc/exports for a single IP
# Arguments: $1 = IP address to add to exports
update_nfs_export_entry() {
    local ip_address="$1"
    # Construct the full export line for the specific IP
    local export_line="${NFS_EXPORT_PATH} ${ip_address}(rw,sync,no_root_squash

    echo "Checking /etc/exports for: ${export_line}"

    # Check if the line already exists in /etc/exports
    if ! grep -qF "${export_line}" /etc/exports; then
        echo "Line not found for ${ip_address}. Adding to /etc/exports..."
        echo "${export_line}" | sudo tee -a /etc/exports > /dev/null
        if [ $? -eq 0 ]; then
            echo "Successfully added line for ${ip_address}
            to /etc/exports."
            NFS_EXPORTS_CHANGED=1 # Set flag as a change was made

```

```

        else
            echo "░░ERROR:░Failed░to░add░line░for░${ip_address}
            to░/etc/exports." >&2
        fi
    else
        echo "░░Line░for░${ip_address}░already░exists
        in░/etc/exports.░Skipping."
    fi
}

# Function to update server hosts file with progress reporting
update_server_hosts() {
    echo "Starting░server░hosts░file░update░and░NFS░exports░configuration..."

    # Read the current /etc/hosts content BEFORE any modifications.
    local original_hosts_content
    original_hosts_content=$(cat /etc/hosts)

    # Create temporary file safely
    tmp_hosts_file=$(mktemp /tmp/hosts_tmp.XXXXXX)
    # Ensure temporary file is removed on script exit
    trap 'rm -f "$tmp_hosts_file"' EXIT

    # Write standard header to the temporary file.
    echo -e "127.0.0.1\tlocalhost" > "$tmp_hosts_file"
    echo -e "127.0.1.1\tcluster" >> "$tmp_hosts_file"

    # Use an associative array to quickly check if an IP is
    within the current processing range.
    declare -A current_range_ips

    # Process all IPs in current range (start to end) with progress counter.
    total_nodes=$((end - start + 1))
    current_node=0

    for i in $(seq "$start" "$end"); do
        current_node=$((current_node + 1))
        ip="${prefix}${i}"
        hostname="nodo$(printf "%02d" "$i")" # Ensure two-digit formatting
        entry="${ip}\t${hostname}"

        printf "\rProcessing░node░%02d/%d:░%-15s" "$current_node"
        "$total_nodes" "$ip" # Updated progress message
    done
}

```

```

# Mark this IP as part of the current processing range.
current_range_ips["$ip"]=1

# Test connection with timeout.
if timeout 1 ping -c 1 "$ip" &>/dev/null; then
    echo -e "$entry" >> "$tmp_hosts_file"
    echo "  Discovered  active  node:  $entry" # Print discovery
    message on a new line.

    # --- CALL NFS EXPORT UPDATE FUNCTION FOR THIS ACTIVE IP ---
    update_nfs_export_entry "$ip"
    # -----
else
    # If offline, add as commented.
    echo -e "#${entry}" >> "$tmp_hosts_file"
    echo "  Node  offline:  $entry" # Print offline
    message on a new line.
fi
done

echo -e "\nAppending  remaining  configuration  from  original  /etc/hosts..."

while IFS= read -r line; do
    if echo "$line" | grep -qE "^127\.0\.0\.1|^127\.0\.1\.1|^::1\|
    00000000|^fe00::0|^ff00::0|^ff02::1|^ff02::2"; then
        continue
    fi

    local extracted_ip=""
    if echo "$line" | grep -qE "^${prefix}"; then
        extracted_ip=$(echo "$line" | awk '{print $1}')
    elif echo "$line" | grep -qE "^#${prefix}"; then
        extracted_ip=$(echo "$line" |
            sed -E "s/^#(${prefix}[0-9\.]*)\.*/\1/")
        # Adjusted regex for IP extraction
    fi

    if [[ -n "$extracted_ip" && ${current_range_ips}
    ["$extracted_ip"]+set == "set" ]]; then
        continue
    fi

    echo "$line" >> "$tmp_hosts_file"

```

```

done <<< "$original_hosts_content"

# Add standard IPv6 block cleanly and only once at the end of the file.
cat <<EOF >> "$tmp_hosts_file"

# Standard IPv6 Configuration
::1\tip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
EOF

# Set correct permissions on the temporary file BEFORE moving it.
chmod 644 "$tmp_hosts_file"

if [ -s "$tmp_hosts_file" ]; then
    if ! mv "$tmp_hosts_file" /etc/hosts; then
        echo "Failed to update /etc/hosts! Restoring backup..." >&2
        cp "$backup_hosts_file" /etc/hosts
        exit 1
    fi
else
    echo "Error: Generated empty hosts file! Aborting." >&2
    exit 1
fi

echo "Server hosts file updated successfully."
}

# Main execution block with error trapping and logging.
{
    echo "Starting IP discovery and server hosts
    file configuration at $(date)"
    echo "===== "

    # Update the server's hosts file and implicitly configure NFS exports
    update_server_hosts

    # --- NFS Service Reload and Restart (Conditional) ---
    if [ "$NFS_EXPORTS_CHANGED" -eq 1 ]; then
        echo -e "\nChanges detected in /etc/exports.
        Reloading NFS exports and restarting nfs-kernel-server..."
        sudo exportfs -a
    fi
}

```

```

    if [ $? -ne 0 ]; then
        echo "Warning: exportfs -a command failed." >&2
    fi
    sudo systemctl restart nfs-kernel-server
    if [ $? -ne 0 ]; then
        echo "ERROR: Failed to restart nfs-kernel-server.
        Please check logs." >&2
        exit 1
    else
        echo "nfs-kernel-server restarted successfully."
    fi
else
    echo -e "\nNo changes detected in /etc/exports.
    NFS service not restarted."
fi
# -----

echo -e "\nOperation completed at $(date)"
echo "===== "
echo "Summary:"
echo "- Server hosts file updated with active \
and discovered node entries within the specified range."
echo "- Original /etc/hosts backed up to $backup_hosts_file"
echo "- Original /etc/exports backed up to $backup_exports_file"
echo "Note: This script now also configures /etc/exports
for active discovered nodes."
} | tee -a "/var/log/cluster_config_$timestamp.log" # Log all
output to a timestamped file.

exit 0

```

Fonte: o Autor, (2025)

Algoritmo 8.2: Atualização do hostname nos nós

```

#!/bin/bash
echo ""
echo "##### "
echo "Atualizar hostname nos NODOS CLUSTER"
echo "##### "

echo ', '
echo 'Atualizar hostname:'
echo ', '
echo ""

```

```

if [[ -z "$1" || -z "$2" ]] ; then
    echo 'uso: ./03_hostname_update.sh 0X Y'
    echo 'exemplo: ./03_hostname_update.sh 04 15'
    exit 1
fi

echo "Intervalo de instala o selecionado"
echo $1 'at' $2

for i in `eval echo {$1..$2}`
do
if ping -c 1 nodo$i > /dev/null
then
    echo "Echo update Nodo $i cpu:"
    echo ''

    ssh -t nodo$i "

./admin_mode.sh

sudo hostnamectl set-hostname nodo$i
hostname

"
    echo ''
else
    # 100% failed
    echo "Host: nodo$i is down at $(date)"
    echo ''
fi
done

echo "END CLUSTER UPDATE ."

```

Fonte: o Autor, (2025)

Algoritmo 8.3: Configuração de ssh Admin Sem Senha

```

#!/bin/bash
echo ""
echo "#####"
echo "SEM SENHA ADMIN CLUSTER"
echo "#####"
echo ''

```

```

echo 'Configurar_acesso_sem_senha(admin):'
echo 'criar_chave:'
echo ""

if [[ -z "$1" || -z "$2" ]] ; then
    echo 'uso: ./02_accessadmin.sh OX Y'
    echo 'exemplo: ./02_accessadmin.sh 02 17'
    exit 1
fi

echo "Intervalo_de_instalacao_selecionado"
echo $1 'at' $2

echo ''
echo 'Install_package:'
echo ''
echo ""

ssh-keygen -t rsa -N "" -f ~/.ssh/id_rsa <<< y

for i in `eval echo {$1..$2}`

do

echo "administrador"

if ping -c 1 nodo$i > /dev/null
then
    echo "Configurando_ssh-copy-id_Nodo$i_cpu:"
    sshpass -p xxxxxxxx ssh-copy-id -o StrictHostKeychecking=no \
    administrador@nodo$i
    echo ''
else
    # 100% failed
echo "Host: nodo$i is down at $(date)"
echo ''
fi
done

echo "END_ADMIN_INSTALL."

```

Fonte: o Autor, (2025)

Algoritmo 8.4: Instalação de Pacotes em Lote

```
#!/bin/bash
```

```

echo ""
echo "#####"
echo "Instalar pacotes nos NODOS CLUSTER"
echo "#####"
echo ""

# Check for minimum arguments
if [[ $# -lt 3 ]]; then
    echo "Uso: ./c_install_package.sh <inicio_intervalo> <fim_intervalo> \
    \<pacote1> [pacote2] [pacote3] ..."
    echo "Exemplo: ./c_install_package.sh 02 15 build-essential htop nmon"
    exit 1
fi

# Extract arguments
start_node=$1
end_node=$2
shift 2 # Remove the first two arguments
packages=("$@") # Remaining arguments are packages

echo "Intervalo de instala o selecionado: \
nodo$start_node at nodo$end_node"
echo "Pacotes selecionados: ${packages[@]}"
echo ""

# Verify node range format
if ! [[ $start_node =~ ^[0-9]+$ ]] || ! [[ $end_node =~ ^[0-9]+$ ]]; then
    echo "Erro: 0 intervalo deve conter apenas n meros"
    echo "Exemplo correto: ./c_install_package.sh 02 15 pacote1 pacote2"
    exit 1
fi

if [ $start_node -gt $end_node ]; then
    echo "Erro: 0 primeiro n mero deve ser menor ou igual ao segundo"
    exit 1
fi

echo "Iniciando instala o ..."
echo ""

success_count=0
fail_count=0
failed_nodes=()

```

```

for i in $(seq -w $start_node $end_node); do
    node="nodo$i"

    if ping -c 1 -W 1 $node > /dev/null 2>&1; then
        echo "===== "
        echo "Instalando em $node..."
        echo "===== "

        # Join packages with spaces for the install command
        package_list="${packages[@]}"

        # SSH command with all package installations
        if ssh -t $node "
            ./admin_mode.sh;
            sudo apt-get update;
            sudo apt --fix-broken install -y;
            sudo apt-get install -y $package_list;
            echo '';
            echo 'Pacotes instalados com sucesso em $node';
            " 2>&1; then

            ((success_count++))
            echo "SUCESSO: $node"
        else
            ((fail_count++))
            failed_nodes+=("$node")
            echo "FALHA: $node"
        fi
    else
        ((fail_count++))
        failed_nodes+=("$node")
        echo "Host $node est offline - $(date)"
    fi

    echo ""
done

echo "===== "
echo "RESUMO DA INSTALACAO"
echo "===== "
echo "N s com sucesso: $success_count"
echo "N s com falha: $fail_count"

```

```

if [ $fail_count -gt 0 ]; then
    echo ""
    echo "Nos que falharam:"
    printf '%s\n' "${failed_nodes[@]}"
fi

echo ""
echo "INSTALACAO_NO_CLUSTER_CONCLUIDA."

```

Fonte: o Autor, (2025)

Algoritmo 8.5: Adição NFS nos nós

```

#!/bin/bash
echo ""
echo "#####"
echo "CONFIGURAR_NFS_NODOS_CLUSTER"
echo "#####"

echo ''
echo 'Install:'
echo ''
echo ""

if [[ -z "$1" || -z "$2" ]] ; then
    echo 'uso: ./d_nfsreload.sh 0X Y'
    echo 'exemplo: ./d_nfsreload.sh 02 17'
    exit 1
fi

echo "Intervalo de instalacao selecionado"
echo "$1 at  $2" # Quote variables for robustness

echo ''
echo 'Update_nfs:'
echo ''
echo ""

# Define the NFS mount line you want to add
NFS_MOUNT_LINE="10.1.1.1:/home/home_nfs
auto,noatime,nolock,bg,intr,tcp,actimeo=1800 0 0"

for i in `eval echo ${1..$2}`
do

```

```

if ping -c 1 nodo$i > /dev/null
then
    echo "Processing_Nodo_$i:"
    echo ''

    ssh -t nodo$i "
        ./admin_mode.sh#_Execute_your_admin_mode_script_first

        #_Check_if_the_NFS_mount_line_already_exists_in_/etc/fstab
        if !_grep -qF "$NFS_MOUNT_LINE" /etc/fstab; then
            echo "NFS mount line not found in /etc/fstab on \$(hostname).
                Adding it...\\"
            echo "$NFS_MOUNT_LINE" | sudo tee -a /etc/fstab > /dev/null
            echo "Line added successfully. Trying to mount...\\"
            sudo mount -a #_Attempt_to_mount_all_entries_in_fstab
            if [_ $? -eq 0]; then
                echo "NFS mounts updated and mounted successfully on \$(hostname).\"
            else
                echo "Warning: mount -a failed after adding line on \$(hostname).
                    Check mount status manually.\\"_>&2
            fi
        else
            echo "NFS mount line already exists in
                /etc/fstab on \$(hostname). Skipping addition.\\"
            sudo mount -a
            #_Still_run_mount -a_to_ensure_all_fstab_entries_are_mounted
            if [_ $? -eq 0]; then
                echo "NFS mounts refreshed successfully on \$(hostname).\"
            else
                echo "Warning: mount -a failed on \$(hostname).
                    Check mount status manually.\\"_>&2
            fi
        fi
    "

    echo ''
else
    # 100% failed
    echo "Host:_nodo$i_is_down_at_$(date +%H:%M:%S %d/%m)"
    echo ''
fi
done

echo "END_CLUSTER_INSTALL."

```

Fonte: o Autor, (2025)

Algoritmo 8.6: Adição de usuários em Lote nos nós

```
#!/bin/bash
echo ""
echo "#####"
echo "Adicionar_usuarios_nos_NODOS_CLUSTER"
echo "#####"

echo ''
echo 'Adi o_usu rios:'
echo ''
echo ""

if [[ -z "$1" || -z "$2" ]] ; then
    echo 'uso: ./04_user_batch_nodes.sh_0X_Y'
    echo 'exemplo: ./04_user_batch_nodes.sh_02_09'
    exit 1
fi

echo "Intervalo_de_instala o_selecionado"
echo $1 'at ' $2

for i in `eval echo {$1..$2}`
do
if ping -c 1 nodo$i > /dev/null
then
    echo "Echo_update_Nodo_$i_cpu:"
    echo ''

    ssh -t nodo$i "

./admin_mode.sh;

sudo_newusers_new_users.txt;

sudo_chpasswd_<_newpass.txt

"
    echo ''
else
    # 100% failed
    echo "Host: _nodo$i_is_down_at_$(date)"
```

```

echo ''
fi
done

echo "END_CLUSTER_UPDATE."

```

Fonte: o Autor, (2025)

Algoritmo 8.7: Configuração ssh de usuários sem senha nos nós

```

#!/bin/bash
echo ""
echo "#####"
echo "ACESSAR_NODOS_CLUSTER"
echo "#####"

for j in {02..30}
do

echo ''
echo 'Configurar_acesso_sem_senha:'
echo 'criar_chave:'
echo ""

ssh-keygen -t rsa

./for i in {02..XX}
do

echo "User:" $j

if ping -c 1 nodo$i > /dev/null
then
    echo "Echo_ssh-copy-id_Nodo_$i_cpu:"
    sshpass -p pwd/pass_file$j ssh-copy-id \
-o StrictHostKeychecking=no user$j@nodo$i

    echo ''
else
    # 100% failed
echo "Host_:nodo$i_is_down_at$(date)"
echo ''
fi
done

```

```
echo "END_User$j_INSTALL."
```

```
done
```

```
echo "END_INSTALL."
```

Fonte: o Autor, (2025)

Algoritmo 8.8: Configuração Munge nos nós

```
#!/bin/bash
echo ""
echo "#####"
echo "Instalar SLURM nos NODOS"
echo "#####"

echo ''
echo 'Install Munge SLURM:'
echo ''
echo ""

if [[ -z "$1" || -z "$2" ]] ; then
    echo 'uso: ./06_install_munge.sh 0X Y'
    echo 'exemplo: ./06_install_munge.sh 14 17'
    exit 1
fi

echo "Intervalo de instala o selecionado"
echo $1 'at' $2

echo ''
echo 'Configurando MUNGE:'
echo ''
echo ""

for i in `eval echo {$1..$2}`
do
if ping -c 1 nodo$i > /dev/null
then
    echo "INSTALL_munge_Nodo_$i_cpu:"
    echo ''

    ssh -t nodo$i "
```

```

./admin_mode.sh;

cd_slurm/;

pwd;

echo "Checando conectividade da internet..."
if ping -c 1 www.google.com > /dev/null; then
    echo "Conexao detectada. Continuando"

    sudo apt update;
    sudo apt install munge libmunge2 libmunge-dev;

else
    echo "Sem internet. Instalando deb local na pasta slurm.";
    sudo dpkg --get-selections | grep munge;
fi

munge -n | unmerge | grep STATUS;
#a_chave_gerada_no_controlador_esta_salva_na_pasta_slurm
#sendo_copiada_aos_nodos
sudo cp /etc/munge/munge.key /etc/munge/munge.key.bak;

sudo cp munge.key /etc/munge/;

#no_arquivo_permissions_o_seguinte_conteudo_e_adicionado
#echo 'permissions'
#sudo chown -R munge: /etc/munge/ /var/log/munge/
#sudo chown munge: /var/lib/munge/ /run/munge/
#sudo chown munge: /etc/munge/munge.key
#sudo chown -R munge: /etc/munge/munge.key

sudo ./permissions;

sudo systemctl enable munge

sudo systemctl restart munge

"
else
    # 100% failed

```

```

echo "Host: _nodo$i_is_down_at_$(date)"
echo ', '
fi
done

```

Fonte: o Autor, (2025)

Algoritmo 8.9: Instalação e configuração do Slurm no controlador e nós

```

#!/bin/bash
echo ""
echo "#####"
echo "Instalar SLURM nos NODOS GTCMC"
echo "#####"

echo ', '
echo 'Install SLURM:'
echo ', '
echo ""

if [[ -z "$1" || -z "$2" ]] ; then
    echo 'uso: ./07_install_slurm-wlm.sh OX Y'
    echo 'exemplo: ./07_install_slurm-wlm.sh 14 17'
    exit 1
fi

echo "Intervalo de instala o selecionado"
echo $1 ' at ' $2

echo ', '
echo 'Configurando SLURM:'
echo ', '
echo ""

for i in `eval echo {$1..$2}`
do
if ping -c 1 nodo$i > /dev/null
then
    echo "INSTALL SLURM-WLM Nodo $i cpu:"
    echo ', '

    ssh -t nodo$i "

~/admin_mode.sh;
cd slurm-wlm/;

```

```

pwd;

####echo "Checando conectividade da internet..."
####if ping -c 1 www.google.com > /dev/null; then
#####echo "Conexao detectada. Continuando"

#####sudo apt update -y;
#####sudo apt install slurm-wlm -y;

####else
#####echo "Sem internet. Instalando deb local na pasta slurm-wlm.";
#####sudo dpkg -i *.deb;
####fi

sudo cp cgroup.conf /etc/slurm-llnl/;

./slurm-worker;

"
    echo ' '

else
    # 100% failed
    echo "Host: _nodo$i_ is down at $(date)"
    echo ' '
fi
done

echo "END_CLUSTER_INSTALL."

```

Fonte: o Autor, (2025)

Algoritmo 8.10: Configuração do slurm.conf no controlador e nós

```

#!/bin/bash
echo ""
echo "#####"
echo "Instalar SLURM nos NODOS_CLUSTER"
echo "#####"

echo ' '
echo 'Atualizar SLURM config <CONTROLLER>:'
echo ' '
echo ""

```

```

sn="02"; #valor do nodo inicial
en="14"; #altere em acordo com o numero de maquinas

echo "Intervalo de instala o selecionado"
echo $sn 'at ' $en

# exit 1;

~/admin_mode.sh;
cd slurm.config/;
# o slurm.conf editavel fica salvo na pasta slurm.config
# e apos editado, e copiado para o controlador e clientes
sudo cp /etc/slurm-llnl/slurm.conf /etc/slurm-llnl/slurm.conf.bak;
sudo cp slurm.conf /etc/slurm-llnl/

#script slurm-controller contem
#sudo systemctl enable slurmctld;
#sudo systemctl restart slurmctld;
#systemctl status slurmctld;
#sinfo

./slurm-controller;

echo ''
echo 'Configurando SLURM:'
echo ''
echo ""

for i in `eval echo {$sn..$en}`
do
if ping -c 1 nodo$i > /dev/null
then
    echo "Atualizar SLURM config <WORKER> Nodo $i cpu:"
    echo ''
    ssh -t nodo$i "

~/admin_mode.sh;

cd slurm.config/;
sudo cp slurm.conf /etc/slurm-llnl/;

#script slurm-worker contem
#sudo systemctl enable slurmd;

```

```
#sudo_systemctl_restart_slurmd;
#systemctl_status_slurmd

./slurm-worker

"
    echo 'finalizando'

pwd

else
    # 100% failed
echo "Host: _nodo$i_is_down_at_$(date)"
echo ', '
fi
done

echo "END_update_slurm.conf"
```

Fonte: o Autor, (2025)

Algoritmo 8.11: Checar slurm no controlador e nós

```
#!/bin/bash
echo ""
echo "#####"
echo "Check_Slurmd_cluster_"
echo "#####"

echo ', '
echo 'Checando_Status:'
echo ', '
echo ""

for i in {02..17}
do
if ping -c 1 nodo$i > /dev/null
then
    echo "Check_Slurmd_Nodo_$i_cpu:"
    echo ', '
    ssh -t nodo$i "systemctl--no-pager-l_status_slurmd"
    echo ', '
else
    # 100% failed
echo "Host: _nodo$i_is_down_at_$(date)"
```

```

echo  ' '
fi
done

echo  "END_CLUSTER_INSTALL."

```

Fonte: o Autor, (2025)

Algoritmo 8.12: Desabilitar modo gráfico nos nós

```

#!/bin/bash
echo  ""
echo  "#####"
echo  "ACESSAR_CLUSTER"
echo  "#####"

echo  ' '
echo  'Disable_gui:'
echo  ' '
echo  ""

if [[ -z "$1" || -z "$2" ]] ; then
    echo  'uso: ./a_disable_gui.sh 0X Y'
    echo  'exemplo: ./a_disable_gui.sh 09 17'
    exit 1
fi

echo  "Intervalo de instala o selecionado"
echo  $1 'at' $2

echo  ' '
echo  'Disable_gui:'
echo  ' '
echo  ""

for i in `eval echo {$1..$2}`
do
    if ping -c 1 nodo$i > /dev/null
    then
        echo  "Echo_INSTALL_Nodo_$i_cpu:"
        echo  ' '
        ssh nodo$i "

        ./admin_mode.sh;

        sudo systemctl set-default multi-user
    fi
done

```

```

"""
    echo  ' ,

else
    # 100% failed
    echo "Host_:_nodo$i_is_down_at_$(date)"
    echo  ' ,
fi

done

echo "END_CLUSTER_CONFIG."
```

Fonte: o Autor, (2025)

Algoritmo 8.13: Configuração do lm-sensor nos nós

```

#!/bin/bash
echo  ""
echo  "#####"
echo  "HABILITAR_LM-SENSORS_CLUSTER"
echo  "#####"

echo  ' ,
echo  'Instalando_: '
echo  ' ,
echo  ""

if [[ -z "$1" || -z "$2" ]] ; then
    echo 'uso: ./e_installsensor.sh_OX_Y'
    echo 'exemplo: ./e_installsensor.sh_09_17'
    exit 1
fi
echo "Intervalo_de_instala_o_selecionado"
echo $1 'at' $2

echo  ' ,
echo  'Habilitar_lm-sensors: '
echo  ' ,
echo  ""

for i in `eval echo {$1..$2}`

do
```

```

if ping -c 1 nodo$i > /dev/null
then

    echo "setup_lm-sensors_in_Nodo_$i:"
    echo ''

    ssh -t nodo$i "

        ./admin_mode.sh;
        sudo sensors-detect --auto

    "
    echo ''

else
    # 100% failed
    echo "Host: nodo$i is down at $(date)"
    echo ''
fi
done

echo "END_CLUSTER_INSTALL."

```

Fonte: o Autor, (2025)

Algoritmo 8.14: Script para monitoramento da temperatura do controlador e nós

```

#!/bin/bash
echo ""
echo "##### "
echo "RELATORIO_PING_NODOS_GTCMC"
echo "##### "

echo ''
echo 'Teste_ethernet_nodos_na_data_e_hora:'
echo ''
date "+%H:%M:%S %d/%m/%y"
echo ""

echo "TR: tempo_de_resposta"
echo "Perda_Pac: perda_de_pacotes"

echo ""

echo '##### '

```

```

echo '    Headnode CPU Temp:    '
echo '#####'
echo ''
# Assuming headnode processor type is known or handled separately
temp=$(sensors | grep "+.*C" | tail -n 2 | head -n 1 | awk '{print $NF}')
echo "temp: $temp"

frequencia=$(lscpu | grep 'MHz' | tail -n 3 | head -n 1 | awk '{print $NF}')
echo "CPU MHz: $frequencia"
echo ''
# Coleta e formata os dados da CPU
printf "Uso da CPU\n"

CPU_LINE=$(top -bn1 | grep "%CPU(s)")
CPU_DATA=$(echo "$CPU_LINE" | awk '{
    gsub(/,/ , , ". ", $0)
    # Substitui virgulas por pontos

    printf "Usuario: %s%\n", $2
    printf "Sistema: %s%\n", $4
    printf "Nice: %s%\n", $6
    printf "Ocioso: %s%\n", $8
}')

echo ''

echo "$CPU_DATA"

echo ""

echo '#####'
echo '    Node CPU Celsius Temps and MHz usage:    '
echo '#####'
echo ''

# Funcao para nodos AMD
get_amd_node_info() {
    local node=$1
    if ping -c 1 "$node" > /dev/null; then
        echo "=== $node ==="
        echo "CPU: $(ssh "$node" 'top -bn1 | grep "%CPU(s)" | \
    | awk '{printf "%.1f%\n", $2}' )"
        echo "LIVRE: $(ssh "$node" 'top -bn1 | grep "%CPU(s)" | \
    | awk '{printf "%.1f%\n", $8}' )"
        # Comando de sensor AMD (e.g., k10temp for AMD CPUs)

```

```

# Ajuste este comando se 'k10temp-*' nao bater com \
a saida de sensor AMD
echo "Temp:$(ssh "$node" 'sensors | grep "+.*C" | head -n 1 \
| awk '{print \$2}' | tr -d " + C "')"
echo "MHZ:$(ssh "$node" 'lscpu | grep "MHz" | tail -n 3 | head -n 1 \
| awk '{print \$NF}')"
else
echo "=== $node ==="
echo "status: down"
echo "$(date +%H:%M:%S %d/%m)"
fi
}

```

```

# funcao para nodos intel
get_intel_node_info() {
local node=$1
if ping -c 1 "$node" > /dev/null; then
echo "=== $node ==="
echo "CPU:$(ssh "$node" 'top -bn1 | grep "%CPU(s)" \
| awk '{printf "%.1f%%\n", \$2}')"
echo "LIVRE:$(ssh "$node" 'top -bn1 | grep "%CPU(s)" \
| awk '{printf "%.1f%%\n", \$8}')"
# Comando de sensor especifico Intel (e.g., coretemp for Intel CPUs)
# Ajuste este comando 'coretemp-*' se nao bater
\com a saida intel
echo "Temp:$(ssh "$node" 'sensors coretemp-* | grep "+.*C" \
| tail -n 3 | head -n 1 | awk '{print \$3}' | tr -d " + C
"')"
echo "MHZ:$(ssh "$node" 'lscpu | grep "MHz" \
| tail -n 3 | head -n 1 | awk '{print \$NF}')"
else
echo "=== $node ==="
echo "status: down"
echo "$(date +%H:%M:%S %d/%m)"
fi
}

```

```

# funcao ajuda para determinar tipo de processador do nodo
get_node_processor_type() {
local node_num_str=$1 # e.g., "02", "15"
# Convert to integer for numeric comparison
local node_num=$((10#$node_num_str)) # Use 10# to ensure base 10 conversion

if (( node_num >= 2 && node_num <= 4 )); then

```

```

        echo "amd"
    elif (( node_num >= 5 && node_num <= 13 )); then
        echo "intel"
    elif (( node_num >= 14 && node_num <= 17 )); then
        echo "amd"
    else
        echo "unknown" # Fallback for unknown node numbers
    fi
}

# List of all nodes (02 to 17)
nodes=$(seq -w 02 17)

# Loop to process all nodes in groups of 3
for ((i = 0; i < ${#nodes[@]}; i += 3)); do
    node_name1="nodo${nodes[i]}"
    info1=""

    node_type1=$(get_node_processor_type "${nodes[i]}")
    if [ "$node_type1" == "amd" ]; then
        info1=$(get_amd_node_info "$node_name1")
    elif [ "$node_type1" == "intel" ]; then
        info1=$(get_intel_node_info "$node_name1")
    else
        info1=$(echo "===_${node_name1}_==\nError:_Unknown_processor_type")
    fi

    info2=""
    if (( i + 1 < ${#nodes[@]} )); then
        node_name2="nodo${nodes[i+1]}"
        info2=""
        node_type2=$(get_node_processor_type "${nodes[i+1]}")
        if [ "$node_type2" == "amd" ]; then
            info2=$(get_amd_node_info "$node_name2")
        elif [ "$node_type2" == "intel" ]; then
            info2=$(get_intel_node_info "$node_name2")
        else
            info2=$(echo "===_${node_name2}_==\nError:_Unknown_processor_type")
        fi
    fi

    info3=""
    if (( i + 2 < ${#nodes[@]} )); then

```

```

node_name3="nodo${nodes[i+2]}"
info3=""
node_type3=$(get_node_processor_type "${nodes[i+2]}")
if [ "$node_type3" == "amd" ]; then
    info3=$(get_amd_node_info "$node_name3")
elif [ "$node_type3" == "intel" ]; then
    info3=$(get_intel_node_info "$node_name3")
else
    info3=$(echo "===_ $node_name3 _===\nError: _Unknown_processor_type")
fi
fi

# Uses 'paste' to print the information side-by-side
paste <($(printf %s "$info1") <($(printf %s "$info2") <($(printf %s "$info3")
echo ', '
done

echo '#####'
echo '      Sinfo_and_Slurm_Queue_Status:      '
echo '#####'
echo ', '
sinfo
echo ', '
echo '#####'
echo ', '
squeue
echo '#####'

```

Fonte: o Autor, (2025)