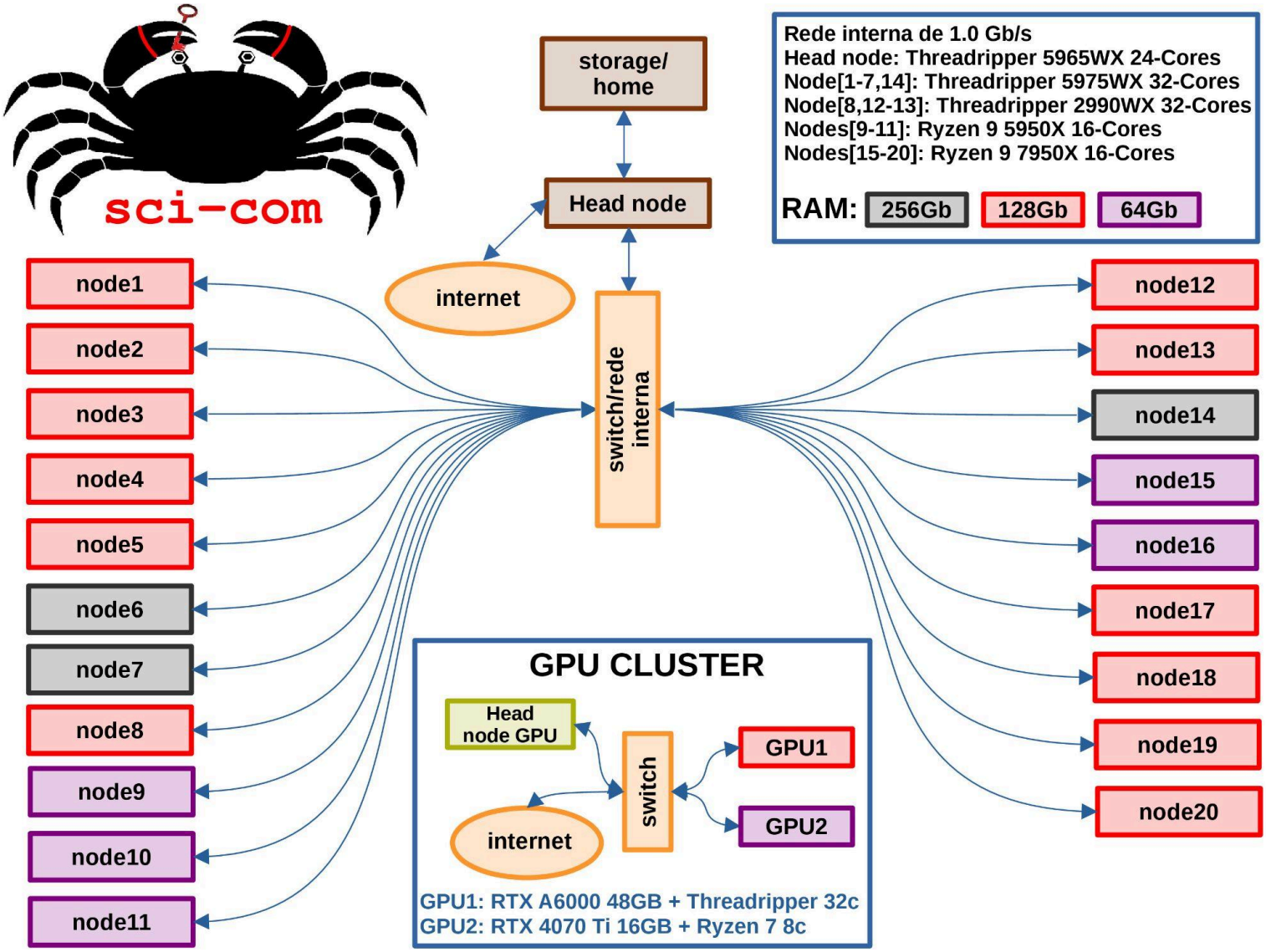
Manual `sci-com`

A Máquina

O sci-com conta com um head node (`scicom`) e 20 compute nodes. Há um storage HDD de 96TB e um scratch local SSD de 1TB em cada nó. O sistema operacional é Rocky Linux e o sistema de gerenciamento de fila é SLURM. A conexão de rede é de 1.0Gbps e há um nobreak de 11KVA e outro de 6KVA.

O sci-com conta também com uma expansão com GPU cuda (`gpu`) que possui 2 compute nodes. Há um storage HDD de 4TB e um scratch local de 1TB (GPU1) e 0.5TB (GPU2). O sistema operacional é Ubuntu server 24.04 e o sistema de gerenciamento de fila é SLURM. A conexão de rede é de 100Mbps.



Agradecimento

Lembrem-se de agradecer em artigos, pôsteres etc o uso do sci-com. Segue uma sugestão:

The authors would like to acknowledge the use of the computational resources provided by the Sci-Com Lab of the Department of Physics at UFES, which was funded by FAPES and CNPq.

Contas

Inicialmente, apenas professores terão contas próprias, e eles compartilharão o acesso com seus alunos. Cada aluno deverá utilizar uma subpasta específica dentro da pasta principal do professor.

Mailing list

Para receber notificações sobre paradas programadas e atualizações do sistema, por favor, inscrevam-se na lista de usuários do sci-com. Acessem o seguinte link para solicitar acesso:
https://groups.google.com/g/usuarios_scicom

Acesso ao head node `scicom`

- Da UFES (mais rápido):
  `ssh conta_scicom@172.20.76.10`

- De fora da UFES, mas no Brasil (IP externo):
  `ssh conta_scicom@200.137.67.4`

- De fora da UFES, mas no Brasil (sshgate):
  `ssh -J conta_ufes@sshgate.ufes.br:22222 conta_scicom@172.20.76.10 -p 22`
  para usar o sshgate.ufes.br precisa botar a chave ssh na sua conta ufes em https://git.ufes.br/

- De fora do Brasil (VPN da UFES, gdrive):
  `ssh conta_scicom@172.20.76.10`

Há bibliotecas já prontas para as arquitetura Zen, Zen2 e Zen3. As bibliotecas Zen3 são compatíveis com Zen4. Fazer `module avail` para ver as bibliotecas e `module load` para carregá-las.

Acesso aos compute nodes

Durante a execução de uma tarefa em um nó específico, é possível acessar este nó para monitorar o uso da CPU e visualizar os arquivos no diretório `local-scratch`. Primeiramente, identifique o nó em que sua tarefa está sendo executada utilizando o comando `squeue -u conta_scicom`.

Suponha que sua tarefa esteja sendo executada no `node3`. Para acessar este nó, execute:

```
ssh node3
```

Após o acesso, você pode monitorar o uso da CPU com o comando:

```
htop
```

Cada nó possui um SSD (/local-scratch) que deve ser usado preferencialmente para executar os jobs. Dessa forma, lembre-se de incluir a linha de comando apropriada no script de submissão. Para visualizar os arquivos no diretório /local-scratch, navegue até este diretório com:

```
cd /local-scratch/
```

Dessa forma, você poderá acompanhar em tempo real o desempenho da CPU e os arquivos sendo gerados ou modificados durante a execução de sua tarefa.

Acesso ao `gpu` (sci-com GPU) e Wolfram Mathematica

- Da UFES (mais rápido):
  `ssh conta_gpu@172.20.76.203`

- De fora da UFES, mas no Brasil (IP externo):
  `ssh conta_scicom@200.137.67.4`
  e depois
  `ssh gpu`

- De fora da UFES, mas no Brasil (sshgate):
  `ssh -J conta_ufes@sshgate.ufes.br:22222 conta_gpu@172.20.76.203 -p 22`
  para usar o sshgate.ufes.br precisa botar a chave ssh na sua conta ufes em https://git.ufes.br/

- De fora do Brasil (VPN da UFES, gdrive):
  `ssh conta_gpu@172.20.76.203`

Uma vez conectado ao `gpu`, é possível utilizar o SLURM do `gpu` para submeter trabalhos para as filas gpu-1 e gpu-2. A fila gpu-1 dispõe de uma GPU com maior capacidade de memória, embora seja mais lenta que a GPU disponível na fila gpu-2 (detalhes na tabela). O head node `gpu` possui um armazenamento de 2TB no `/storage`.

Para o funcionamento do SLURM é necessário poder acessar os nós `gpu1` e `gpu2`, sem senha, via ssh-key. É necessário configurar isso só uma vez para a conta do professor responsável. A partir do `gpu`, fazer:

```
ssh-keygen -t rsa -b 4096
ssh-copy-id conta_gpu@gpu1
ssh-copy-id conta_gpu@gpu2
```

Uma vez configurado isso, é possível acessar `gpu1` e `gpu2` fazendo (sem senha):

```
ssh gpu1
ssh gpu2
```

No sci-com GPU também está disponível o Wolfram Mathematica. Para utilizar o Mathematica, basta executar o comando `math`. Os links simbólicos para os executáveis estão localizados em `/usr/local/bin` e os binários podem ser encontrados em `/usr/local/Wolfram/Mathematica/14.0/Executables`. Abaixo, seguem as informações necessárias para registrar o Mathematica na primeira utilização:

gpu1
MathID: 6524-45398-96445
Activation key: 3523-8649-Y6E7Q8
Password: 6409-893-680:8,8,64,64:9:20240908

gpu2
MathID: 6509-97311-40790
Activation key: 3523-8649-XG5THV
Password: 2958-189-100:8,8,16,16:9:20240908

Transferência de dados

Para transferir dados do seu computador para o scicom é conveniente usar rsync:

```
rsync -avh --progress --partial --inplace $VSOUR $VDEST >$VOUT 2>$VERR
```

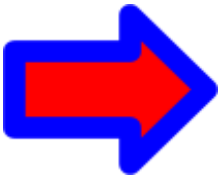
onde foram definidas, eg:

```
VOUT=test.out
VERR=test.err
VSOUR=test
VDEST=conta_scicom@172.20.76.131:/home/conta_scicom/fulano/
```

Vejam mais opções aqui.

SLURM

Fila	Tempo	Nós	Hardware	Memória	Local scratch	Geração
z3-long	14 dias	node[6-7, 14]	Threadripper 5975WX 32c	**256**GB	1TB	Zen 3
z3-short	3 dias	node[1-7, 14]	Threadripper 5975WX 32c	128/**256**GB	1TB	Zen 3
z1-short	3 dias	node[8, 12-13]	Threadripper 2990WX 32c	128GB	1TB	Zen 1
ry-short	3 dias	node[10-11]	Ryzen 9 5950X 16c	64GB	1TB	Zen 3
ry-short	3 dias	node[15-16]	Ryzen 9 7950X 16c	64GB	1TB	Zen 4
ry-short	3 dias	node[17-20]	Ryzen 9 7950X 16c	128GB	1TB	Zen 4
dev	5 horas	node9	Ryzen 9 5950X 16c	64GB	1TB	Zen 3
gpu-1	3 dias	gpu1	RTX A6000 48GB + Threadripper 3970X 32c	128GB	1TB	Zen 2
gpu-2	3 dias	gpu2	RTX 4070 Ti 16GB + Ryzen 7 8c	64GB	0.5TB	Zen 2



Lembrar que o scratch será esvaziado a cada 15 dias.

Exemplo de script para `scicom`

```
#!/bin/bash
#SBATCH --job-name=nome_aluno-tarefa       #Nome job com o nome do aluno/prof
#SBATCH --nodes=1                          #Numero de Nós
#SBATCH --ntasks=1                         #Numero total de tarefas MPI/OPENMP
#SBATCH --partition z3-short               #Fila a ser utilizada; é possível colocar
mais filas: z3-short,z1-short
#SBATCH --output %x-slurm_job-%j.out
#SBATCH --error  %x-slurm_job-%j.err

#opcional
##SBATCH --exclusive                       #para usar um nó exclusivamente
##SBATCH --exclude=node[1-13]

echo $SLURM_SUBMIT_DIR                     # faça sbatch da pasta onde está o script
cd $SLURM_SUBMIT_DIR

# ...standard slurm stuff...

# transferir tudo pro scratch do nó e depois voltar para pasta local
scratch_dir='/local-scratch/'${SLURM_JOB_NAME}
```

```
mkdir -p $scratch_dir
cp -r * $scratch_dir
cd $scratch_dir

#mpirun -np $SLURM_NTASKS “exe”
touch result.txt
srun sleep 60

home_dir=${SLURM_SUBMIT_DIR}/${SLURM_JOB_NAME}
mkdir -p $home_dir
mv ${scratch_dir}/* $home_dir

echo 'done!'
```

Exemplo de script para o headnode `gpu`

```
#!/bin/bash
#SBATCH --job-name=nome_aluno-tarefa
#SBATCH --account=conta_gpu
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --partition=gpu-1
#SBATCH --output %x-slurm_job-%j.out
#SBATCH --error  %x-slurm_job-%j.err

echo "where am i?"
pwd
echo $HOSTNAME
echo ""
…
```

Introdução ao SLURM

SLURM é uma ferramenta de código aberto para gerenciar e agendar trabalhos em clusters de computadores. Ele permite que você use e compartilhe recursos computacionais de maneira eficiente. O SLURM é altamente flexível e pode ser configurado para gerenciar de algumas a muitas milhares de máquinas.

Comandos básicos:
- `sinfo`: fornece informações sobre os nós do cluster SLURM. Mostra quais nós estão disponíveis, quais estão sendo usados e qual é o seu estado.
- `squeue`: mostra todos os trabalhos atualmente na fila. Pode ser útil redefinir o comando squeue:
  `alias squeue='squeue --iterate=3 --format="%.8i %20P %20j %10u %8T %.10M %9D %20R"'`
- `sbatch`: usado para submeter um trabalho para o cluster. Você precisa escrever um script de shell que inclui comandos SLURM (veja os exemplos acima) e o comando que deseja executar. Para submeter o script precisa executar: `sbatch script.sh`
- `scancel`: permite que você cancele um trabalho que submeteu. Use o comando `scancel` seguido pelo ID do trabalho, que pode ser encontrado usando o comando `squeue -u conta_scicom`
- `srun`: permite a execução interativa de comandos ou scripts em um nó de computação atribuído pelo SLURM. O srun é frequentemente usado para iniciar processos dentro de uma alocação existente ou para solicitar recursos e iniciar um processo imediatamente.

- `scontrol show jobid #NUMERO-DO-JOB -dd`: exibe informações detalhadas sobre um job.

Manual para o comando sbatch: https://slurm.schedmd.com/sbatch.html

Vídeo-explicação pelo Rodrigo Amorim, que configurou o scicom.

Introdução a rsync options

`rsync -PRavzhHS`

- `-a` (or `-archive`): Preserves file attributes, permissions, timestamps, and symbolic links.
- `-v` (or `-verbose`): Provides verbose output.
- `-z` (or `-compress`): Enables compression during the transfer, reducing network bandwidth usage.

Those 3 above seem like the main flags to use.

- `-h` (ou `-human-readable`): Displays the output in a human-readable format, using units like kilobytes (KB), megabytes (MB), etc.
- `-P` combination of `-progress` and `-partial`
    - `-progress`: Displays the progress of the transfer, keeping you informed about the current status.
    - `-partial`: Allows partial file transfers, enabling resumption of interrupted transfers.
- `-R` ensures that the entire directory hierarchy is replicated on the destination
    - It's worth noting that if you only need to sync the contents of a directory without preserving the entire path structure, using `-r` is usually sufficient instead of `-R`
- `-S`: Using this flag is particularly beneficial when syncing files that contain **sparse** sections, such as virtual machine disk images or log files with large blank spaces. It ensures that only the actual data is transferred, improving the efficiency of the synchronization operation.
- `-H`: If the source has files with hard links, to preserve those hard links on the destination or/and to save disk space on the destination by using hard links instead of duplicating data. Don't use it if the source data do not contain hard links or the destination system do not support hard links.
- `–inplace`: When `rsync` transfers a file to the destination, it creates a temporary file with a different name and then renames it to the original filename once the transfer is complete. However, with `--inplace`, `rsync` updates the destination file in-place without creating a temporary file. This can save disk space, so if disk space is a concern use it. The same thing can also increase the speed, so if speed is a concern, use. Because with this flag `rsync` updates the destination directory without creating temporary file, if the transfer process gets interrupted, the destination file may be left in an inconsistent state. In such cases, you may end up with a partially updated file on the destination.
- `--exclude=".*"`: to not transfer the hidden files.