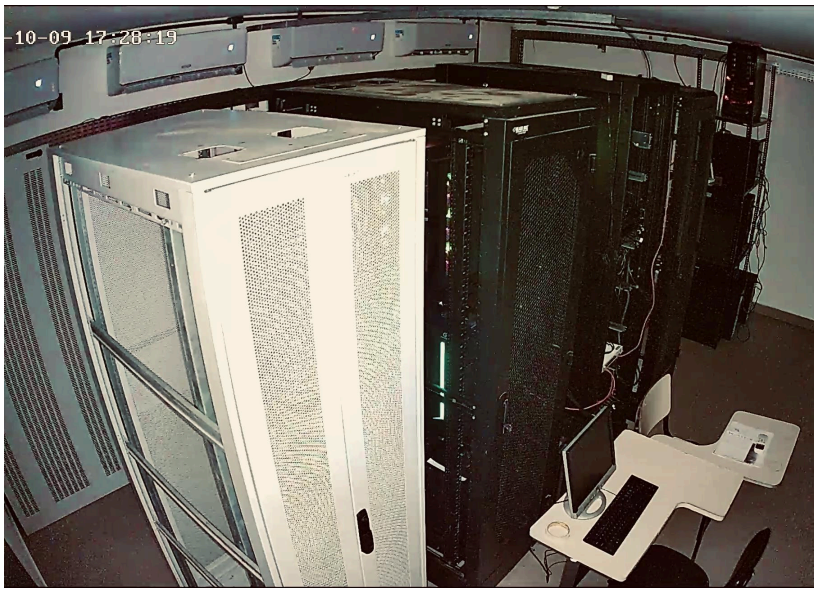
Manual `scicom`

A Máquina

O Laboratório Multiusuário de Computação Científica (Sci-Com) está localizado na sala 7 do anexo A do CCE. O espaço conta com quatro sistemas de ar-condicionado, monitoramento eletrônico e remoto de temperatura, instalação elétrica dedicada e controle de acesso físico restrito, com câmeras internas de segurança. Mais informações estão disponíveis em https://computacaocientifica.ufes.br/infraestrutura.



O Sci-Com é composto por CPUs AMD da família Zen, com um nó principal (headnode, `scicom`) e mais de 20 nós de computação. Cada nó possui ainda um scratch local SSD M.2 de 1 TB ou mais. A rede interna opera em 25 Gbps por fibra óptica e recebe uma conexão à internet por fibra óptica de 10 Gbps. O sistema é protegido por nobreaks com potência total de 16 kVA. O armazenamento principal consiste em um armazenamento HDD de 96 TB configurado em RAID 6. Há também um armazenamento SSD NVMe com capacidade total de 16 TB, configurado em RAID 0, para a conexão de 100 Gbps da rede e-Ciência. O sistema operacional é Rocky Linux e o gerenciamento de tarefas é realizado pelo SLURM.

Agradecimento

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

The authors acknowledge the use of computational resources from the \href{https://computacaocientifica.ufes.br/scicom}{Sci-Com Lab} of the Department of Physics at UFES, supported by FAPES, CAPES, and CNPq.

Contas

Inicialmente, apenas professores terão contas próprias, e eles compartilharão o acesso com seus alunos. Cada aluno deverá utilizar uma subpasta específica e ambiente conda/mamba 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

Para solicitações e dúvidas, favor escrever exclusivamente para scicom@ufes.br

Acesso ao head node `scicom`

1. Da UFES (mais rápido):
   a. `ssh conta_scicom@172.20.76.10`

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

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

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

Há bibliotecas já prontas para as arquiteturas Zen, Zen2 e Zen3. As bibliotecas Zen3 são compatíveis com Zen4.

O sci-com utiliza o Spack como sistema unificado de instalação e carregamento de pacotes. Para consultar os softwares disponíveis, o usuário deve empregar o ***`spack find`*** para visualizar os pacotes já instalados no sistema, incluindo versões, variantes e compiladores. O carregamento de um software específico no ambiente é feito com ***`spack load`*** <pacote> (podendo-se especificar versão, compilador ou variantes, quando necessário). De forma complementar, comandos como ***`spack info`*** <pacote> permitem inspecionar detalhes do pacote, ***spack spec*** <pacote> mostram a árvore completa de dependências e opções de compilação, e ***spack unload*** <pacote> remove o pacote do ambiente corrente.

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 no nó de execução 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.

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.

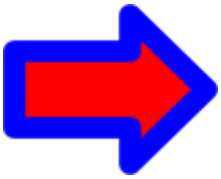
SLURM

O headnode é um Threadripper 5965wx 24c e 128 GB de memória.

O sistema de armazenamento é equipado com 2 processadores Intel Xeon Silver 4310 12c e 256 GB de memória RAM. A área `home` possui capacidade de 96 TB, configurada em RAID 6 (6 HDDs de 16 TB), enquanto o `home-scratch` conta com 16 TB em RAID 0 (4 SSDs NVMe de 4 TB), alcançando vazão nominal de leitura e escrita de até 224 Gbps. Esse storage é responsável pela interface direta com a conexão de 100 Gbps da rede e-Ciência.

Fila	Tempo	Nós	Hardware	Memória	Local scratch	Geração
ry-internet	3d	node15	Ryzen 9 7950X 16c	64GB	1TB	Zen 4
z3-long	14d	node[1,2, 14]	Threadripper 5975WX 32c	256GB	1TB/2TB para 1 e 2	Zen 3
z3-short	3d	node[1-7, 14]	Threadripper 5975WX 32c	128/256GB	1TB	Zen 3
z1-short	3d	node[8,12-13]	Threadripper 2990WX 32c	128GB	1TB	Zen 1
ry-short	3d	node[15-16]	Ryzen 9 7950X 16c	64GB	1TB	Zen 4
ry-short	3d	node[17-24]	Ryzen 9 7950X 16c	128GB	1TB	Zen 4
ep-short	14d	node25	1 x Epyc 9754 128c	768GB	4TB	Zen 4
ep-short	14d	node26	2 x Epyc 9754 128c	1536GB	8TB	Zen 4

gpu-nv	3d	nodenv2	RTX A6000 48GB + Threadripper 3970X 32c	128GB	1TB	Zen 2
gpu-nv	3d	nodenv1	RTX 4070 Ti 16GB + Ryzen 9 5950X 16c	64GB	1TB	Zen 3



Lembrar que o scratch será esvaziado a cada 15 dias.

Exemplo de script

```
#!/bin/bash
#SBATCH --job-name=nome_aluno-tarefa       #Nome job com o nome do aluno/prof
#SBATCH --nodes=1                          #Número 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!'
```

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.