

Entregável 3.1. Processamento do PyCCD em plataforma de supercomputação (MACC)

Índice

1. Introdução	3
2. Acesso ao MACC	5
3. Upload dos dados no MACC	5
4. Instalação dos recursos necessários do PyCCD no MACC	9
5. Executar scripts no MACC	11
6. Resultados e conclusões	14
7. Anexos	16
7.1. Comandos SLURM	16
7.2. Scripts PyCCD (utilizando MPI)	17

Índice das Figuras

Figura 1 – Partições disponíveis no MACC.	4
Figura 2 – Ligação ao MACC.	5
Figura 3 – Mensagem exibida ao conectar-se com sucesso ao MACC.	5
Figura 4 – Configuração do acesso via SFTP pelo WinSCP para aceder ao cluster do INCD.	6
Figura 5 – Configuração do acesso via SFTP pelo WinSCP para aceder ao MACC... 6	6
Figura 6 – Configuração avançada do acesso via SFTP pelo WinSCP.	7
Figura 7 – Configuração da chave privada para estabelecer ligação ao MACC através do WinSCP.	8
Figura 8 – Autenticação com senha para concluir a conexão com o MACC.	8
Figura 9 – O painel à esquerda apresenta as pastas e diretorias armazenados localmente na máquina do utilizador, enquanto o painel à direita exibe as pastas e diretorias localizados no MACC.	9
Figura 10 – Ligação ao MACC por SSH.	9
Figura 11 – Download da pasta do PyCCD para o MACC, via GitHub.	10
Figura 12 – Estrutura da pasta 'S2CHANGE' após o download ser concluído no MACC.	10
Figura 13 – Download do script de instalação do Miniconda no MACC.	10
Figura 14- Permissão de execução do script de instalação do Miniconda no MACC.	10
Figura 15 – Iniciar a instalação do Miniconda no MACC.	11

Figura 16 – Criação do ambiente virtual (ccdISA) para processamento do PyCCD no MACC.....	11
Figura 17 – Lista dos ambientes virtuais disponíveis no MACC.	11
Figura 18 – Ativação do ambiente virtual ccdISA no MACC.....	11
Figura 19 – Exemplo do ficheiro de submissão (submit_job.sh) para execução de um script no MACC.	12
Figura 20 – Comando utilizado para submeter o ficheiro de submissão submit_job.sh ao SLURM para agendamento e execução do job no MACC e respetiva mensagem retornada pelo SLURM indicando a submissão bem-sucedida do job, com o identificador único 313612.....	13
Figura 21 – Comando utilizado para monitorizar em tempo real o arquivo de log slurm-313612.out, exibindo as mensagens de execução, erros ou resultados intermediários.	13
Figura 22 – Exemplo de saída do comando sacct, exibindo informações detalhadas sobre jobs concluídos no SLURM, incluindo ID do job, nome, usuário, estado final, número de nós utilizados, tempo de execução e consumo de CPU.	14

Índice das Tabelas

Tabela 1 – Resultados dos testes realizados para 1 milhão de pixels na partição normal-x86. O tempo de computação refere-se exclusivamente ao cálculo do CCD, não incluindo o tempo de pré-processamento dos dados nem o tempo necessário para salvar os resultados.	15
Tabela 2 – Projeção do tempo de processamento e consumo de core-horas para 500 milhões de pixels, considerando a configuração de 25 nós e 5 CPUs por nó.	15

1. Introdução

Este relatório descreve as etapas realizadas no âmbito da execução do projeto definido no entregável E.3.1., com foco na avaliação e implementação de alternativas para o processamento computacional do PyCCD. Inicialmente, o processamento do código PyCCD foi realizado na Infraestrutura Nacional de Computação Distribuída (INCD), por meio de uma candidatura à Rede Nacional de Computação Avançada (RNCA), na categoria A0 de acesso experimental (com um total de 50.000 core-horas de CPU). Neste ambiente configurado com 5 nós e 96 CPUs, para uma amostra de 1 milhão de pixels o tempo de processamento foi de 21 minutos e 45 segundos consumindo cerca de 274.3 core-horas.

Com a obtenção de novos recursos computacionais por meio de uma nova candidatura ao RNCA, o objetivo agora é reduzir ainda mais o tempo de processamento e otimizar a utilização dos recursos disponíveis. Desta vez, o acesso é do tipo A1, ou seja, um acesso de desenvolvimento, e a infraestrutura utilizada é Minho Advanced Computing Center (MACC), que oferece maior capacidade de computação. Com esta nova infraestrutura, dispomos de 100.000 core-horas de CPU e 1.500 GPU-horas, tornando viável o processamento do PyCCD para todas as tiles que constituem Portugal Continental, garantindo maior eficiência e escalabilidade para lidar com a totalidade dos dados (500 milhões de pixels).

Ainda estão em andamento testes para otimizar o pré-processamento dos dados, que inclui a leitura dos GeoTiffs para cada tile e a construção de ficheiros (numpy ou hdf5) necessários para o processamento do CCD. Além disso, estão a ser realizados ajustes no formato de saída, uma vez que o script está a guardar mais informações do que o necessário.

Quanto às diferenças entre as infraestruturas, no INCD, na partição destinada à execução do nosso código (denominada fct), utilizávamos um total de 5 nós, cada um com 96 CPUs e 512 GB de RAM. Já no MACC, temos 3 partições dedicadas a CPUs (geralmente com "x86" no nome) e 3 outras destinadas a GPUs (geralmente com "a100" no nome) – Figura 1. Nas partições de CPU, contamos com uma partição com 2 nós, outra com 64 nós e uma terceira com 128 nós. Cada nó possui 128 CPUs e 256 GB de RAM.

Partition	Architecture	Max Nodes	Time Limit
dev-arm	aarch64	2	4 hours
normal-arm	aarch64	128	48 hours
large-arm	aarch64	512	72 hours
dev-x86	x86_64	2	4 hours
normal-x86	x86_64	64	48 hours
large-x86	x86_64	128	72 hours
dev-a100-40	x86_64	1	4 hours
normal-a100-40	x86_64	4	48 hours
dev-a100-80	x86_64	1	4 hours
normal-a100-80	x86_64	4	48 hours

Figura 1 – Partições disponíveis no MACC.

Os resultados e testes apresentados neste relatório, agora realizados no ambiente MACC, são consistentes com o código que foi testado no INCD. No entanto, surgiram algumas dificuldades ao tentar adaptar o código e os ambientes/módulos na nova máquina, devido às diferenças nas configurações em relação ao INCD. Após esses testes, foi possível executar o código PyCCD, testando diversas configurações de nós e CPUs, cujos resultados serão detalhados neste relatório.

2. Acesso ao MACC

Após a aceitação do projeto e a criação das contas correspondentes no portal MACC (<https://portal.deucalion.macc.fccn.pt/user/login>), foram fornecidos os acessos necessários. O processo de conexão é realizado por meio do programa MobaXterm, disponível em: <https://mobaxterm.mobatek.net/>.

Para estabelecer a ligação ao cluster, deve abrir o MobaXterm. No terminal deve executar o seguinte comando, substituindo 'scaetano' pelo nome de utilizador escolhido e id_rsa_macc pelo caminho onde guardou a chave privada: `ssh -i id_rsa_macc scaetano@login.deucalion.macc.fccn.pt`

Logo após deverá digitar a password que definiu ao criar a chave privada:

```
scaetano@DMmlc3:~$ ssh -i id_rsa_macc scaetano@login.deucalion.macc.fccn.pt
Enter passphrase for key 'id_rsa_macc':
```

Figura 2 – Ligação ao MACC.

Após inserir as credenciais corretamente, o acesso ao servidor será estabelecido, e deverá aparecer a seguinte janela:

```
88888888b. 888888888888 888 888 .d88888b. d8888 888 8888888 .d88888b. 888b 888
888 Y88b 888 888 888 d88P Y88b d88888 888 888 d88P Y88b 8888b 888
888 888 888 888 888 888 888 d88P888 888 888 888 888 88888b 888
888 888 8888888 888 888 888 d88P 888 888 888 888 888 888Y88b 888
888 888 888 888 888 888 888 d88P 888 888 888 888 888 888 Y88b888
888 888 888 888 888 888 888 d88P 888 888 888 888 888 888 Y88888
888 .d88P 888 Y88b. .d88P Y88b d88P d8888888888 888 888 Y88b. .d88P 888 Y8888
88888888P 88888888888 Y88888P Y8888P 888 8888888 8888888 888888 Y88888P 888 Y888
Last login: Sat Feb 22 13:55:23 2025 from 193.136.147.150

Greetings scaetano

Welcome to the Portuguese EuroHPC supercomputer, *please* read documentation before use:
https://docs.macc.fccn.pt/

=====
Filesystem      Space  Quota  Limit Files  Quota  Limit
/home/scaetano  11426M 20480M 25600M 187k    200k   250k
=====
Disk quotas for prj 60120 (pid 60120):
Filesystem      used  quota  limit  grace  files  quota  limit  grace
/projects/F202410004CPCAA1
                2.414T  15T    18T    -      7133    0     0     -
=====
```

Figura 3 – Mensagem exibida ao conectar-se com sucesso ao MACC.

3. Upload dos dados no MACC

Para carregar dados diretamente no MACC, utilizou-se o programa WinSCP, disponível para download em: <https://winscp.net/eng/download.php>. A configuração do acesso ao cluster utilizando o WinSCP, foi feita seguindo os passos abaixo:

- I. Abrir o WinSCP, de seguida clicar em New Tab. Uma nova janela de login será exibida:

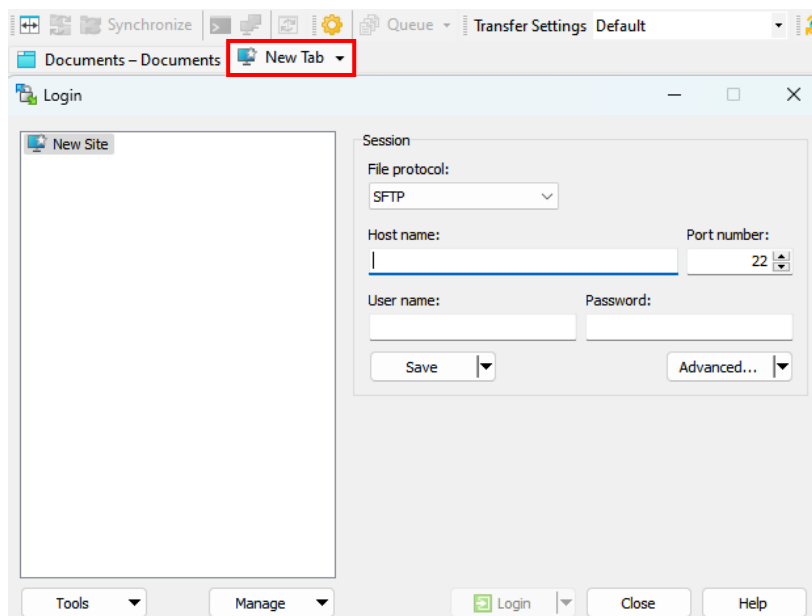


Figura 4 – Configuração do acesso via SFTP pelo WinSCP para aceder ao cluster do INCD.

- II. Configuração da sessão: preencher os campos conforme abaixo.
- Hostname: inserir o endereço do cluster (login.deucalion.macc.fccn.pt).
 - Username: inserir o nome de utilizador fornecido.

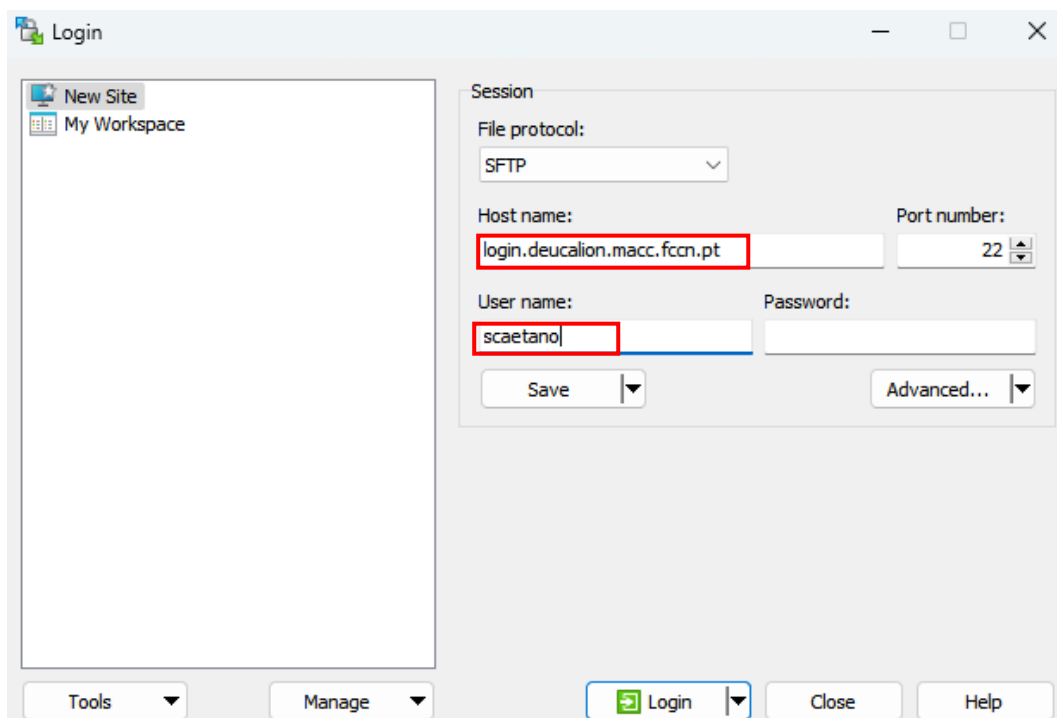


Figura 5 – Configuração do acesso via SFTP pelo WinSCP para aceder ao MACC.

- III. Configuração Avançada: clicar no botão “Advanced...”. Aceder à aba “SSH > Authentication”. No campo destinado à chave de autenticação, clicar no botão “...” e seleccionar o ficheiro da chave privada (no formato ppk) que foi gerado com a chave pública para a criação da conta no cluster:

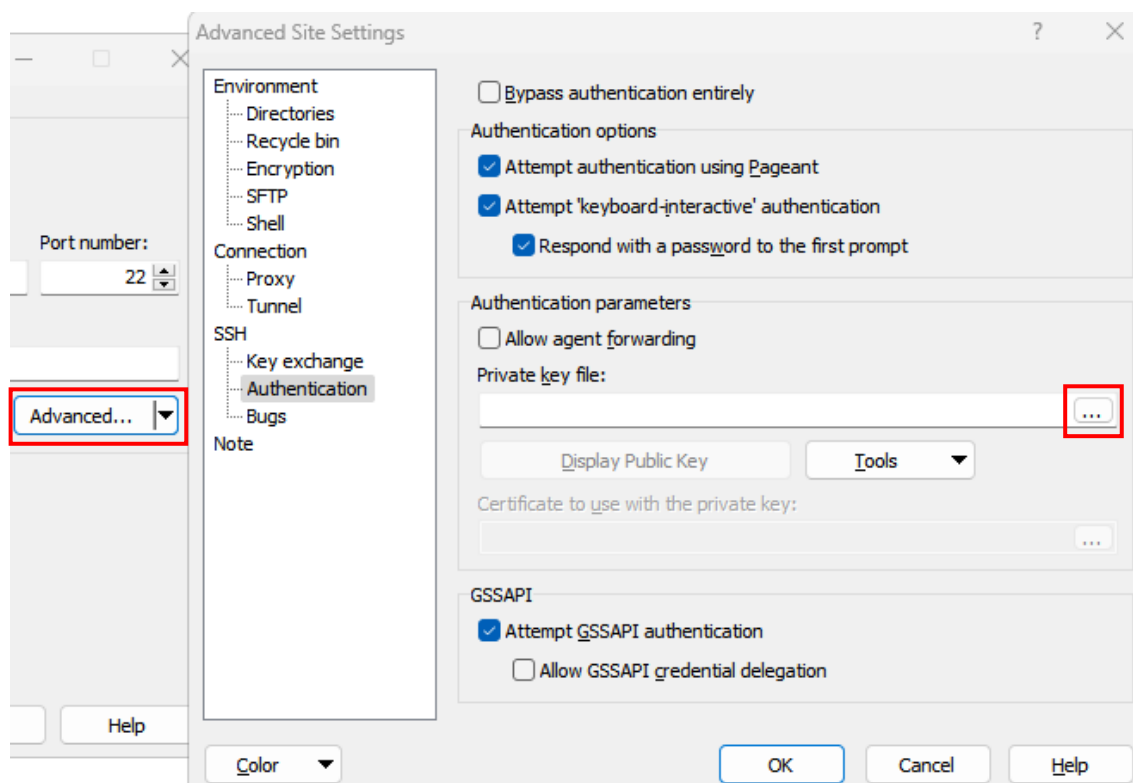


Figura 6 – Configuração avançada do acesso via SFTP pelo WinSCP.

- IV. Após seguir os passos, deverá ficar com o seguinte aspeto:

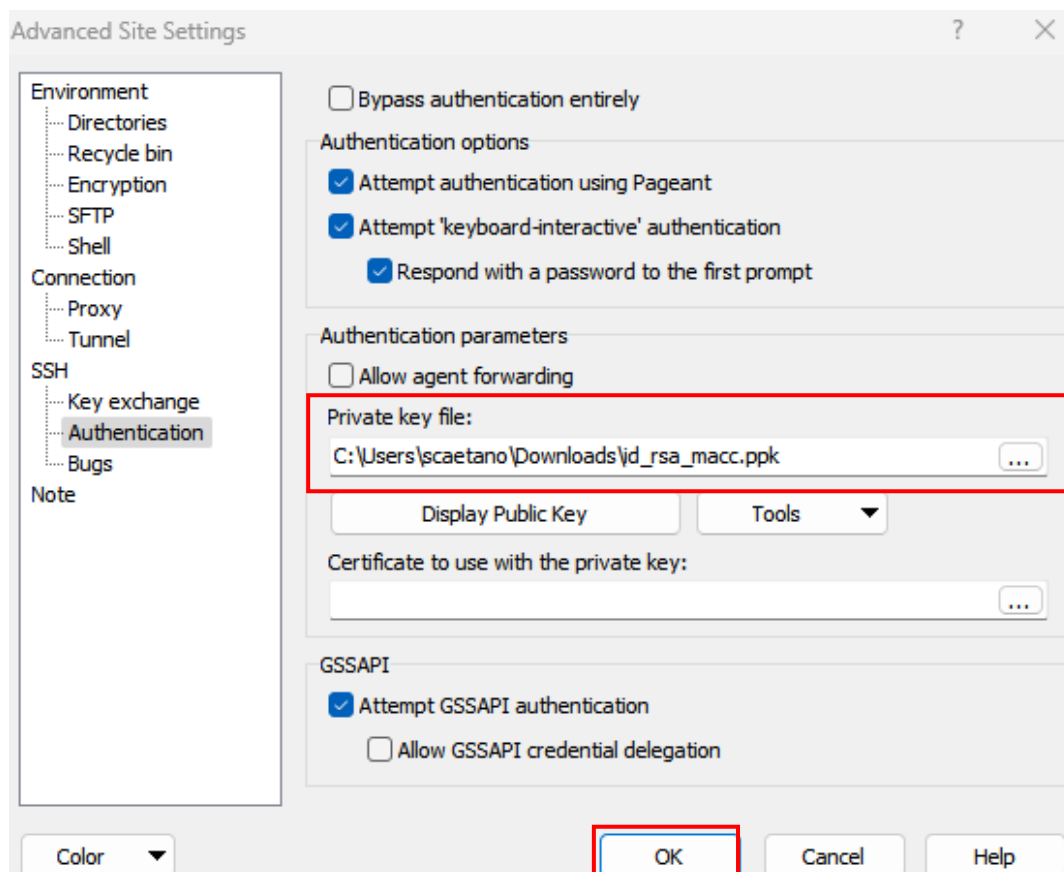


Figura 7 – Configuração da chave privada para estabelecer ligação ao MACC através do WinSCP.

- V. Precisar  inserir a senha definida ao criar a chave privada para concluir a conex o com o MACC com sucesso:

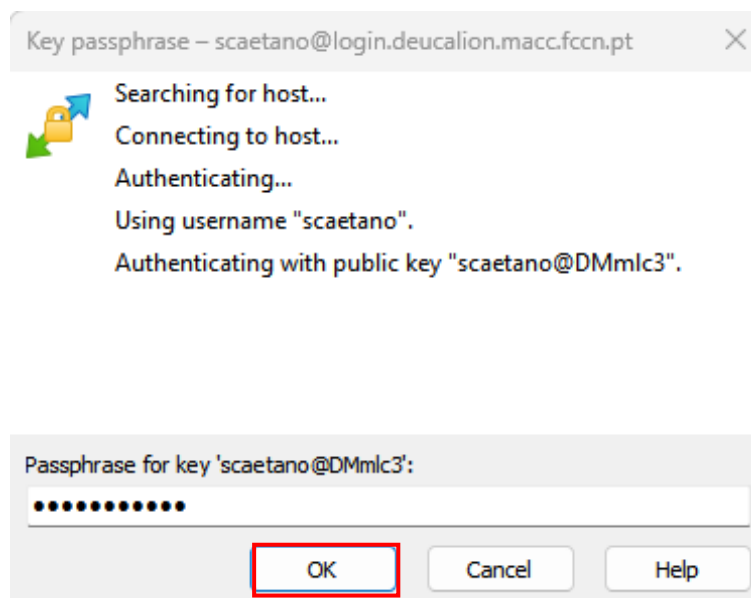


Figura 8 – Autentica o com senha para concluir a conex o com o MACC.

Com o cluster devidamente configurado no WinSCP o programa está apto a realizar a transferência de ficheiros entre a máquina local e o MACC.

- Painel à esquerda: exhibe os ficheiros e diretorias armazenados na máquina local.
- Painel à direita: apresenta os ficheiros e diretorias disponíveis no MACC.

A partir desta interface, é possível copiar, mover ou editar ficheiros de forma prática e intuitiva entre os dois ambientes.

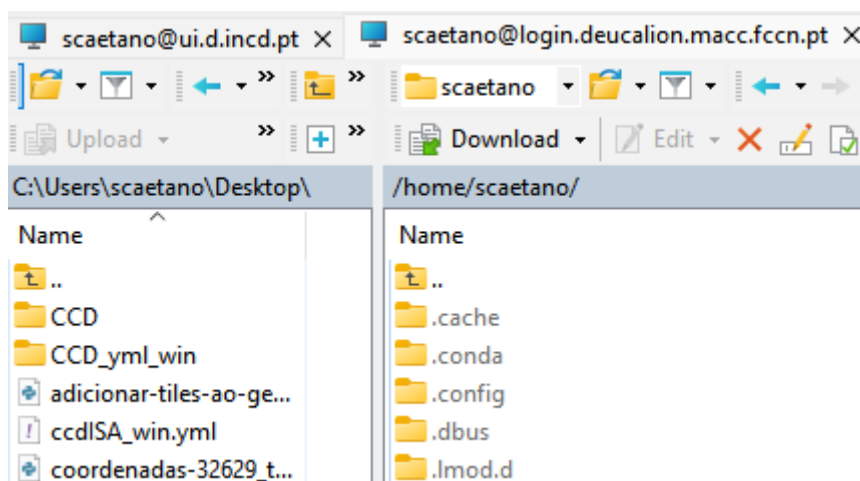


Figura 9 – O painel à esquerda apresenta as pastas e diretorias armazenados localmente na máquina do utilizador, enquanto o painel à direita exhibe as pastas e diretorias localizados no MACC.

4. Instalação dos recursos necessários do PyCCD no MACC

A configuração do PyCCD no MACC foi realizada via GitHub. A pasta transferida (nome: S2CHANGE) para o MACC já contém os scripts do PyCCD. O ficheiro **ccdISA_macc.yml** está presente na pasta e contém as especificações necessárias para a criação do ambiente virtual requerido para a execução do PyCCD.

Os passos seguintes só precisam de ser realizados uma única vez no MACC:

- I. Abrir o mobaXterm e estabelecer ligação com o cluster. No terminal, deve inserir o seguinte comando: **ssh -i id_rsa_macc scaetano@login.deucalion.macc.fccn.pt**

```
scaetano@DMmlc3:~$ ssh -i id_rsa_macc scaetano@login.deucalion.macc.fccn.pt
Enter passphrase for key 'id_rsa_macc':
```

Figura 10 – Ligação ao MACC por SSH.

- II. Após estabelecer a ligação ao MACC, deve fazer o download da pasta do PyCCD utilizando o seguinte comando: **git clone https://github.com/manuelcampagnolo/S2CHANGE.git**

```
(base) [scaetano@ln01 ~]$ git clone https://github.com/manuelcampagnolo/S2CHANGE.git
Cloning into 'S2CHANGE'...
remote: Enumerating objects: 1487, done.
remote: Counting objects: 100% (310/310), done.
remote: Compressing objects: 100% (99/99), done.
remote: Total 1487 (delta 272), reused 210 (delta 210), pack-reused 1177 (from 1)
Receiving objects: 100% (1487/1487), 109.77 MiB | 5.61 MiB/s, done.
Resolving deltas: 100% (802/802), done.
Updating files: 100% (95/95), done.
```

Figura 11 – Download da pasta do PyCCD para o MACC, via GitHub.

- III. Quando o download for concluído, a pasta "S2CHANGE" estará na sua diretoria principal e terá a seguinte estrutura:

```
/home/scaetano/S2CHANGE/
Name
..
.git
documents
scripts
.gitignore
ccdISA_linux.yml
ccdISA_macc.yml
ccdISA_win.yml
README.md
```

Figura 12 – Estrutura da pasta 'S2CHANGE' após o download ser concluído no MACC.

- IV. Após fazer o download da pasta do PyCCD, é necessário instalar o Miniconda que facilita a criação/instalação e a gestão de ambientes virtuais. O primeiro passo é baixar o script de instalação do Miniconda para o servidor, executando o seguinte comando: **wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh**

```
(base) [scaetano@ln01 ~]$ wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
--2025-02-23 10:35:48-- https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
Resolving repo.anaconda.com (repo.anaconda.com)... 104.16.32.241, 104.16.191.158, 2606:4700::6810:20f1, ...
Connecting to repo.anaconda.com (repo.anaconda.com)|104.16.32.241|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 154615621 (147M) [application/octet-stream]
Saving to: 'Miniconda3-latest-Linux-x86_64.sh'
```

Figura 13 – Download do script de instalação do Miniconda no MACC.

- V. Agora é necessário dar permissão de execução do script de instalação que acabou de fazer download. Executar o seguinte comando: **chmod +x Miniconda3-latest-Linux-x86_64.sh**

```
(base) [scaetano@ln01 ~]$ chmod +x Miniconda3-latest-Linux-x86_64.sh
```

Figura 14- Permissão de execução do script de instalação do Miniconda no MACC.

- VI. Com o script agora executável, iniciar a instalação do Miniconda com o comando: **./Miniconda3-latest-Linux-x86_64.sh**

```
(base) [scaetano@ln01 ~]$ ./Miniconda3-latest-Linux-x86_64.sh

Welcome to Miniconda3 py312_25.1.1-2

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>> █
```

Figura 15 – Iniciar a instalação do Miniconda no MACC.

- VII. Após fazer o download do Miniconda, é necessário criar o ambiente virtual chamado **ccdISA**, onde os scripts do PyCCD serão executados. Para isso, deve utilizar o seguinte comando: **conda env create -f S2CHANGE/ccdISA_linux.yml**

```
(base) [scaetano@ln01 ~]$ conda env create -f S2CHANGE/ccdISA_macc.yml
```

Figura 16 – Criação do ambiente virtual (ccdISA) para processamento do PyCCD no MACC.

- VIII. Certifique-se de que o ambiente foi instalado corretamente com o comando: **conda info --envs**. Será exibido uma lista de todos os ambientes disponíveis. Confirme se o ambiente **ccdISA** consta na lista:

```
(base) [scaetano@ln01 ~]$ conda info --envs

# conda environments:
#
base                * /home/scaetano/miniconda3
ccdISA              /home/scaetano/miniconda3/envs/ccdISA
```

Figura 17 – Lista dos ambientes virtuais disponíveis no MACC.

- IX. Uma vez confirmado que o ambiente está instalado, ative-o com o comando: **conda activate ccdISA**. O prompt do terminal será alterado para indicar que o ambiente está ativo, apresentando um formato semelhante a este:

```
(base) [scaetano@ln01 ~]$ conda activate ccdISA
(ccdISA) [scaetano@ln01 ~]$ █
```

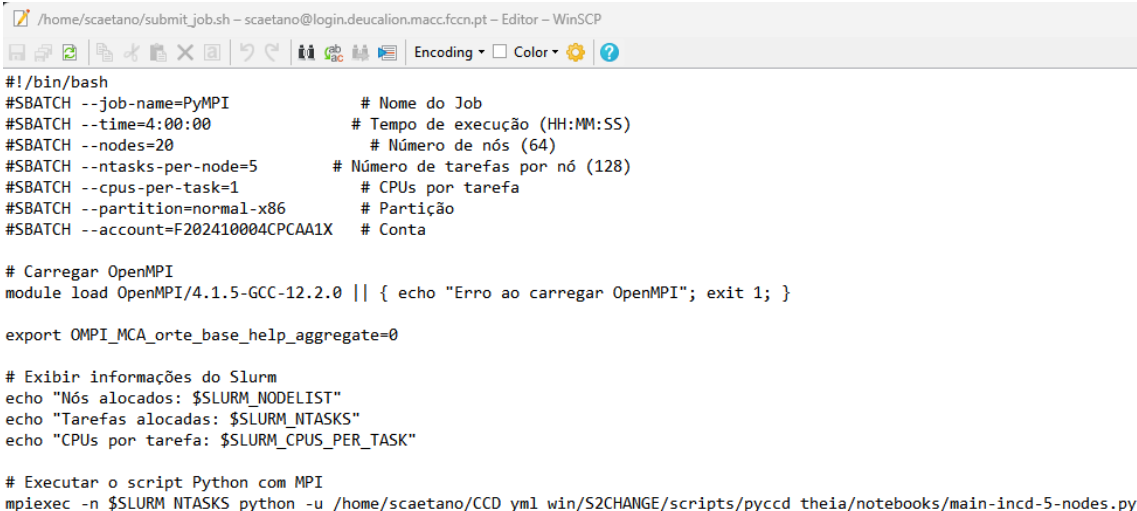
Figura 18 – Ativação do ambiente virtual ccdISA no MACC.

5. Executar scripts no MACC

Após a configuração e ativação do ambiente virtual **ccdISA**, a execução de scripts no MACC é realizada por meio de um ficheiro de submissão chamado **submit_job.sh**, que deve ser criado e armazenado na diretoria correspondente no MACC (/home/scaetano). Este ficheiro funciona como um guia estruturado, onde

especifica os parâmetros essenciais para o agendamento, alocação de recursos e execução dos scripts no SLURM.

A Figura 19 mostra um exemplo de um ficheiro **submit_job.sh** configurado para executar o script principal do PyCCD no MACC.



```
#!/bin/bash
#SBATCH --job-name=PyMPI           # Nome do Job
#SBATCH --time=4:00:00             # Tempo de execução (HH:MM:SS)
#SBATCH --nodes=20                 # Número de nós (64)
#SBATCH --ntasks-per-node=5       # Número de tarefas por nó (128)
#SBATCH --cpus-per-task=1         # CPUs por tarefa
#SBATCH --partition=normal-x86    # Partição
#SBATCH --account=F202410004CPCAA1X # Conta

# Carregar OpenMPI
module load OpenMPI/4.1.5-GCC-12.2.0 || { echo "Erro ao carregar OpenMPI"; exit 1; }

export OMPI_MCA_orte_base_help_aggregate=0

# Exibir informações do Slurm
echo "Nós alocados: $SLURM_NODELIST"
echo "Tarefas alocadas: $SLURM_NTASKS"
echo "CPUs por tarefa: $SLURM_CPUS_PER_TASK"

# Executar o script Python com MPI
mpirun -n $SLURM_NTASKS python -u /home/scaetano/CCD_ym1_win/S2CHANGE/scripts/pyccd_theia/notebooks/main-incd-5-nodes.py
```

Figura 19 – Exemplo do ficheiro de submissão (submit_job.sh) para execução de um script no MACC.

No ficheiro de submissão **submit_job.sh**, mostrado na Figura 19, encontram-se os principais parâmetros utilizados para configurar o ambiente de execução no SLURM, sendo estes:

- **#SBATCH --job-name:** Define um nome identificador para o projeto. Este nome aparecerá no sistema de monitorização do cluster.
- **#SBATCH --time:** Especifica o tempo máximo de execução. Após esse período, o trabalho será automaticamente encerrado, mesmo que não tenha sido terminado.
- **#SBATCH --nós:** Determina o número de nós a serem utilizados para o projeto.
- **#SBATCH --ntasks-per-node:** Define o número de tarefas simultâneas que cada nó irá processar.
- **#SBATCH --partition:** Especifica a partição (grupo de recursos) onde o script será executado.
- **#SBATCH --account:** Identificador do projeto que permite reconhecer os recursos disponíveis.

Para submeter o ficheiro de submissão ao cluster, utiliza-se o comando: **sbatch submit_job.sh**, que envia o job ao sistema de agendamento. Após a submissão, o sistema retorna um identificador único, como no exemplo:

```
(ccdISA) [scaetano@ln01 ~]$ sbatch submit_job.sh
Submitted batch job 313612
```

Figura 20 – Comando utilizado para submeter o ficheiro de submissão `submit_job.sh` ao SLURM para agendamento e execução do job no MACC e respetiva mensagem retornada pelo SLURM indicando a submissão bem-sucedida do job, com o identificador único 313612.

Este identificador serve para visualizar o status do trabalho e acompanhar a sua execução. Durante a execução do job, o comando: **stdbuf -oL python** do ficheiro **submit_job.sh** é responsável por iniciar o script **main.py** do PyCCD, garantindo que a saída seja exibida em tempo real no terminal. O progresso pode ser monitorizado com o comando: **tail -f slurm-313612.out**, que permite visualizar o conteúdo do arquivo de log gerado pelo SLURM, incluindo mensagens de execução ou possíveis erros:

```
(ccdISA) [scaetano@ln01 ~]$ tail -f slurm-313612.out
Núms alocados: cnx[299-318]
Tarefas alocadas: 100
CPUs por tarefa: 1
Numero total de pixels processados: 1000000
Executando com batch_size = 50 e N = 1000000
Numero de CPUs para o ProcessPoolExecutor: 100
O arquivo '/projects/F202410004CPCAA1/outputs_RI/numpy/T29TNE/'
e processando os dados existentes...
A recolher nome e data dos tifs...
[Rank 0] Total de batches criados: 20000
Processo 4: 100%|██████████| 199/200 [06:06<00:01, 1.84s/it]]
Processo 54: 100%|██████████| 200/200 [06:13<00:00, 1.87s/it]]
Processo 9: 100%|██████████| 200/200 [06:15<00:00, 1.88s/it]]
Processo 79: 100%|██████████| 200/200 [06:15<00:00, 1.88s/it]]
Processo 94: 100%|██████████| 200/200 [06:15<00:00, 1.88s/it]]
Processo 99: 100%|██████████| 199/200 [06:16<00:01, 1.89s/it]]
Processo 49: 100%|██████████| 200/200 [06:16<00:00, 1.88s/it]]
Processo 29: 100%|██████████| 200/200 [06:17<00:00, 1.89s/it]]
Processo 24: 100%|██████████| 200/200 [06:18<00:00, 1.89s/it]]
Processo 89: 100%|██████████| 200/200 [06:18<00:00, 1.89s/it]]
Processo 19: 100%|██████████| 200/200 [06:19<00:00, 1.90s/it]]
Processo 39: 100%|██████████| 200/200 [06:19<00:00, 1.90s/it]]
Processo 14: 100%|██████████| 199/200 [06:19<00:01, 1.91s/it]]
Processo 44: 100%|██████████| 200/200 [06:20<00:00, 1.90s/it]]
Processo 69: 100%|██████████| 200/200 [06:22<00:00, 1.91s/it]]
Processo 59: 99%|██████████| 198/200 [06:22<00:03, 1.93s/it]]
Processo 34: 100%|██████████| 199/200 [06:23<00:01, 1.93s/it]]
Processo 84: 100%|██████████| 200/200 [06:25<00:00, 1.93s/it]]
Processo 74: 99%|██████████| 198/200 [06:27<00:03, 1.96s/it]]
Processo 64: 100%|██████████| 200/200 [06:28<00:00, 1.94s/it]]
Processo 0: 100%|██████████| 200/200 [07:19<00:00, 2.20s/it]]
Processo 7: 100%|██████████| 200/200 [07:19<00:00, 2.20s/it]]
Processo 1: 100%|██████████| 200/200 [07:21<00:00, 2.21s/it]]
```

Figura 21 – Comando utilizado para monitorizar em tempo real o arquivo de log `slurm-313612.out`, exibindo as mensagens de execução, erros ou resultados intermediários.

Para uma análise mais detalhada do uso dos recursos do MACC, pode-se utilizar o seguinte comando: **sacct -A F202410004CPCAA1X --starttime=2025-01-01 --format=JobID,JobName,User,State,NNodes,Elapsed,CPUTime,CPUTimeRAW**

Abaixo encontra-se a explicação de cada parâmetro do anterior comando:

- **sacct**: Exibe informações sobre jobs já concluídos no sistema de gerenciamento de filas SLURM.
- **-A F202410004CPCAA1X**: Especifica a conta (Account) utilizada para a execução dos jobs (ID do projeto).
- **--starttime=2025-02-25**: Define a data de início da consulta, garantindo que apenas jobs finalizados a partir de 1 de janeiro de 2025 sejam listados.
- **JobID**: Identificação do job.
- **JobName**: Nome atribuído ao job pelo usuário.
- **User**: Usuário que submeteu o job.
- **State**: Estado final do job (exemplo: COMPLETED, FAILED, CANCELLED).
- **NNós**: Número de nós utilizados na execução.
- **Elapsed**: Tempo total de execução (HH:MM:SS).
- **CPUTime**: Tempo total de CPU consumido pelo job.
- **CPUTimeRAW**: Tempo total de CPU consumido pelo job, em core-segundos.

No SLURM, as extensões como "batch", "extern+" e "0" correspondem a subprocessos ou componentes do job principal. No entanto, o CPUTimeRAW do job sem extensão já engloba o tempo total de CPU utilizado, incluindo essas subdivisões:

313910	PyMPI	scaetano	COMPLETED	25	00:23:06	51-08:00:00	4435200
313910.batch	batch		COMPLETED	1	00:23:06	2-01:16:48	177408
313910.exte+	extern		COMPLETED	25	00:23:06	51-08:00:00	4435200
313910.0	orted		COMPLETED	24	00:23:06	51-08:00:00	4435200

Figura 22 – Exemplo de saída do comando **sacct**, exibindo informações detalhadas sobre jobs concluídos no SLURM, incluindo ID do job, nome, usuário, estado final, número de nós utilizados, tempo de execução e consumo de CPU.

6. Resultados e conclusões

O objetivo principal dos testes foi avaliar se o tempo de computação poderia ser reduzido em comparação com a máquina do INCD (~20 minutos), utilizando o mesmo número de 10^6 pixels. Os testes foram realizados na partição normal-x86 (64 nós e 128 CPUs) e estão detalhados na Tabela 1. Abaixo estão os resultados obtidos para diferentes configurações de nós e tarefas por nó (= número de CPUs):

PARTIÇÃO: normal-x86				
Nós	Ntasks-per-node/CPU	Total Ntasks	Tempo de processamento	Core-horas (inclui salvar outputs)
5	96	480	oom_killed	
10	10	100	14 mins e 8 segs	569.6
15	10	150	oom_killed	
20	5	100	7 mins e 53 segs	4014080
20	10	200	oom_killed	
25	5	125	6 mins e 29 segs	1232
30	5	150	5 mins e 43 segs	CANCELLED

Tabela 1 – Resultados dos testes realizados para 1 milhão de pixels na partição normal-x86. O tempo de computação refere-se exclusivamente ao cálculo do CCD, não incluindo o tempo de pré-processamento dos dados nem o tempo necessário para salvar os resultados.

A partir dos testes realizados, foi possível observar que o tempo de computação diminui à medida que o número de nós aumenta, embora alguns testes resultem em falhas devido à falta de memória (indicado por "oom_killed"). Nos casos em que o número de nós era menor, como em 5 nós com 96 tarefas por nó (totalizando 480 tarefas), o processamento não foi concluído devido ao erro de falta de memória. No entanto, com a configuração de 10 nós e 10 tarefas por nó, o tempo de computação foi de 14 minutos e 8 segundos, mostrando uma melhoria em relação ao tempo de 20 minutos observado na infraestrutura do INCD.

Ao aumentar o número de nós e diminuir o número de tarefas por nó, como na configuração de 20 nós e 5 tarefas por nó (totalizando 100 tarefas), o tempo de computação foi reduzido para 7 minutos e 53 segundos. A configuração de 25 nós e 5 tarefas por nó levou a um tempo de computação de 6 minutos e 29 segundos, e a configuração de 30 nós e 5 tarefas por nó reduziu ainda mais o tempo para 5 minutos e 43 segundos, o que representa uma melhoria significativa.

No geral, os testes mostram que a utilização de mais nós e menos tarefas por nó proporciona uma redução no tempo de processamento, embora seja importante observar que o uso excessivo de tarefas por nó pode levar a falhas de memória.

Com base nos testes realizados, é possível estimar o tempo de processamento e o consumo de core-horas para um cenário com 500 milhões de pixels. Esta projeção considera as configurações de 25 nós e 5 CPUs por nó, e não inclui o tempo de pré-processamento dos dados nem o tempo necessário para salvar os resultados:

NÓS	CPU	TEMPO DE PROCESSAMENTO	CORE-HORAS
25	5	~ 54 horas	616.000

Tabela 2 – Projeção do tempo de processamento e consumo de core-horas para 500 milhões de pixels, considerando a configuração de 25 nós e 5 CPUs por nó.

7. Anexos

7.1. Comandos SLURM

De seguida, encontram-se listados alguns comandos utilizados no SLURM, acompanhados das respetivas descrições:

- **squeue -u <username>** exibe a lista de jobs ativos do utilizador especificado:

```
(ccdISA) [scaetano@ln01 ~]$ squeue -u scaetano
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	ODELIST(REASON)
-------	-----------	------	------	----	------	-------	-----------------

- **sbatch <script>** submete um job para execução a partir de um script de submissão:

```
(ccdISA) [scaetano@ln01 ~]$ sbatch submit_job.sh
```

Submitted batch job 313612

- **scancel <JOBID>** cancela a execução de um job específico, utilizando o seu identificador (JOBID):

```
(ccdISA) [scaetano@ln01 ~]$ scancel -u scaetano
```

- **sinfo** exibe informações sobre as partições e nós disponíveis no MACC:

```
(ccdISA) [scaetano@ln01 ~]$ sinfo
```

PARTITION	AVAIL	TIMELIMIT	NODES	STATE	ODELIST
ooda	up	8:00:00	11	alloc	cna[0017-0027]
dev-arm	up	4:00:00	16	idle	cna[0001-0016]
normal-arm	up	2-00:00:00	5	drain*	cna[0115-0117,1287-1288]
normal-arm	up	2-00:00:00	147	down*	cna[0105-0114,0118-0120,6320,1465-1472,1529-1536,1593-1600,1609-1616]
normal-arm	up	2-00:00:00	215	alloc	cna[0017-0099,0897-1016,1528,1537-1592,1601-1608,1627,1630-1632]
normal-arm	up	2-00:00:00	1249	idle	cna[0100-0104,0121-0176,6320,1465-1472,1529-1536,1593-1600,1609-1616]
large-arm	up	3-00:00:00	5	drain*	cna[0115-0117,1287-1288]
large-arm	up	3-00:00:00	147	down*	cna[0105-0114,0118-0120,6320,1465-1472,1529-1536,1593-1600,1609-1616]
large-arm	up	3-00:00:00	215	alloc	cna[0017-0099,0897-1016,1528,1537-1592,1601-1608,1627,1630-1632]
large-arm	up	3-00:00:00	1249	idle	cna[0100-0104,0121-0176,6320,1465-1472,1529-1536,1593-1600,1609-1616]
dev-x86	up	4:00:00	8	idle	cnx[001-008]
normal-x86	up	2-00:00:00	5	down*	cnx[191,294,319,324,488]
normal-x86	up	2-00:00:00	4	drain	cnx[230,292,298,389]
normal-x86	up	2-00:00:00	25	alloc	cnx[031,153,293,295-297,309-311,313,315,317,319,321,323,325,327,329,331,333,335,337,339,341,343,345,347,349,351,353,355,357,359,361,363,365,367,369,371,373,375,377,379,381,383,385,387,389,391,393,395,397,399,401,403,405,407,409,411,413,415,417,419,421,423,425,427,429,431,433,435,437,439,441,443,445,447,449,451,453,455,457,459,461,463,465,467,469,471,473,475,477,479,481,483,485,487,489,491,493,495,497,499,501,503,505,507,509,511,513,515,517,519,521,523,525,527,529,531,533,535,537,539,541,543,545,547,549,551,553,555,557,559,561,563,565,567,569,571,573,575,577,579,581,583,585,587,589,591,593,595,597,599,601,603,605,607,609,611,613,615,617,619,621,623,625,627,629,631,633,635,637,639,641,643,645,647,649,651,653,655,657,659,661,663,665,667,669,671,673,675,677,679,681,683,685,687,689,691,693,695,697,699,701,703,705,707,709,711,713,715,717,719,721,723,725,727,729,731,733,735,737,739,741,743,745,747,749,751,753,755,757,759,761,763,765,767,769,771,773,775,777,779,781,783,785,787,789,791,793,795,797,799,801,803,805,807,809,811,813,815,817,819,821,823,825,827,829,831,833,835,837,839,841,843,845,847,849,851,853,855,857,859,861,863,865,867,869,871,873,875,877,879,881,883,885,887,889,891,893,895,897,899,901,903,905,907,909,911,913,915,917,919,921,923,925,927,929,931,933,935,937,939,941,943,945,947,949,951,953,955,957,959,961,963,965,967,969,971,973,975,977,979,981,983,985,987,989,991,993,995,997,999]
normal-x86	up	2-00:00:00	458	idle	cnx[009-030,032-152,154-156,158-160,162-184,186-188,190-192,194-196,198-200,202-204,206-208,210-212,214-216,218-220,222-224,226-228,230-232,234-236,238-240,242-244,246-248,250-252,254-256,258-260,262-264,266-268,270-272,274-276,278-280,282-284,286-288,290-292,294-296,298-300,302-304,306-308,310-312,314-316,318-320,322-324,326-328,330-332,334-336,338-340,342-344,346-348,350-352,354-356,358-360,362-364,366-368,370-372,374-376,378-380,382-384,386-388,390-392,394-396,398-400,402-404,406-408,410-412,414-416,418-420,422-424,426-428,430-432,434-436,438-440,442-444,446-448,450-452,454-456,458-460,462-464,466-468,470-472,474-476,478-480,482-484,486-488,490-492,494-496,498-500,502-504,506-508,510-512,514-516,518-520,522-524,526-528,530-532,534-536,538-540,542-544,546-548,550-552,554-556,558-560,562-564,566-568,570-572,574-576,578-580,582-584,586-588,590-592,594-596,598-600,602-604,606-608,610-612,614-616,618-620,622-624,626-628,630-632,634-636,638-640,642-644,646-648,650-652,654-656,658-660,662-664,666-668,670-672,674-676,678-680,682-684,686-688,690-692,694-696,698-700,702-704,706-708,710-712,714-716,718-720,722-724,726-728,730-732,734-736,738-740,742-744,746-748,750-752,754-756,758-760,762-764,766-768,770-772,774-776,778-780,782-784,786-788,790-792,794-796,798-800,802-804,806-808,810-812,814-816,818-820,822-824,826-828,830-832,834-836,838-840,842-844,846-848,850-852,854-856,858-860,862-864,866-868,870-872,874-876,878-880,882-884,886-888,890-892,894-896,898-900,902-904,906-908,910-912,914-916,918-920,922-924,926-928,930-932,934-936,938-940,942-944,946-948,950-952,954-956,958-960,962-964,966-968,970-972,974-976,978-980,982-984,986-988,990-992,994-996,998-1000]
large-x86	up	3-00:00:00	5	down*	cnx[191,294,319,324,488]
large-x86	up	3-00:00:00	4	drain	cnx[230,292,298,389]
large-x86	up	3-00:00:00	25	alloc	cnx[031,153,293,295-297,309-311,313,315,317,319,321,323,325,327,329,331,333,335,337,339,341,343,345,347,349,351,353,355,357,359,361,363,365,367,369,371,373,375,377,379,381,383,385,387,389,391,393,395,397,399,401,403,405,407,409,411,413,415,417,419,421,423,425,427,429,431,433,435,437,439,441,443,445,447,449,451,453,455,457,459,461,463,465,467,469,471,473,475,477,479,481,483,485,487,489,491,493,495,497,499,501,503,505,507,509,511,513,515,517,519,521,523,525,527,529,531,533,535,537,539,541,543,545,547,549,551,553,555,557,559,561,563,565,567,569,571,573,575,577,579,581,583,585,587,589,591,593,595,597,599,601,603,605,607,609,611,613,615,617,619,621,623,625,627,629,631,633,635,637,639,641,643,645,647,649,651,653,655,657,659,661,663,665,667,669,671,673,675,677,679,681,683,685,687,689,691,693,695,697,699,701,703,705,707,709,711,713,715,717,719,721,723,725,727,729,731,733,735,737,739,741,743,745,747,749,751,753,755,757,759,761,763,765,767,769,771,773,775,777,779,781,783,785,787,789,791,793,795,797,799,801,803,805,807,809,811,813,815,817,819,821,823,825,827,829,831,833,835,837,839,841,843,845,847,849,851,853,855,857,859,861,863,865,867,869,871,873,875,877,879,881,883,885,887,889,891,893,895,897,899,901,903,905,907,909,911,913,915,917,919,921,923,925,927,929,931,933,935,937,939,941,943,945,947,949,951,953,955,957,959,961,963,965,967,969,971,973,975,977,979,981,983,985,987,989,991,993,995,997,999]
large-x86	up	3-00:00:00	458	idle	cnx[009-030,032-152,154-156,158-160,162-184,186-188,190-192,194-196,198-200,202-204,206-208,210-212,214-216,218-220,222-224,226-228,230-232,234-236,238-240,242-244,246-248,250-252,254-256,258-260,262-264,266-268,270-272,274-276,278-280,282-284,286-288,290-292,294-296,298-300,302-304,306-308,310-312,314-316,318-320,322-324,326-328,330-332,334-336,338-340,342-344,346-348,350-352,354-356,358-360,362-364,366-368,370-372,374-376,378-380,382-384,386-388,390-392,394-396,398-400,402-404,406-408,410-412,414-416,418-420,422-424,426-428,430-432,434-436,438-440,442-444,446-448,450-452,454-456,458-460,462-464,466-468,470-472,474-476,478-480,482-484,486-488,490-492,494-496,498-500,502-504,506-508,510-512,514-516,518-520,522-524,526-528,530-532,534-536,538-540,542-544,546-548,550-552,554-556,558-560,562-564,566-568,570-572,574-576,578-580,582-584,586-588,590-592,594-596,598-600,602-604,606-608,610-612,614-616,618-620,622-624,626-628,630-632,634-636,638-640,642-644,646-648,650-652,654-656,658-660,662-664,666-668,670-672,674-676,678-680,682-684,686-688,690-692,694-696,698-700,702-704,706-708,710-712,714-716,718-719,720,721,722,723,724,725,726,727,728,729,730,731,732,733,734,735,736,737,738,739,740,741,742,743,744,745,746,747,748,749,750,751,752,753,754,755,756,757,758,759,760,761,762,763,764,765,766,767,768,769,770,771,772,773,774,775,776,777,778,779,780,781,782,783,784,785,786,787,788,789,790,791,792,793,794,795,796,797,798,799,800,801,802,803,804,805,806,807,808,809,810,811,812,813,814,815,816,817,818,819,820,821,822,823,824,825,826,827,828,829,830,831,832,833,834,835,836,837,838,839,840,841,842,843,844,845,846,847,848,849,850,851,852,853,854,855,856,857,858,859,860,861,862,863,864,865,866,867,868,869,870,871,872,873,874,875,876,877,878,879,880,881,882,883,884,885,886,887,888,889,890,891,892,893,894,895,896,897,898,899,900,901,902,903,904,905,906,907,908,909,910,911,912,913,914,915,916,917,918,919,920,921,922,923,924,925,926,927,928,929,930,931,932,933,934,935,936,937,938,939,940,941,942,943,944,945,946,947,948,949,950,951,952,953,954,955,956,957,958,959,960,961,962,963,964,965,966,967,968,969,970,971,972,973,974,975,976,977,978,979,980,981,982,983,984,985,986,987,988,989,990,991,992,993,994,995,996,997,998,999]
dev-a100-40	up	4:00:00	1	drain	gnx511
dev-a100-40	up	4:00:00	3	alloc	gnx[502-504]
dev-a100-40	up	4:00:00	13	idle	gnx[506,510,512-513,516,518-519,521-522,524-525,527-528,530-531,533-534,536-537,539-540,542-543,545-546,548-549,551-552,554-555,557-558,560-561,563-564,566-567,569-570,572-573,575-576,578-579,581-582,584-585,587-588,590-591,593-594,596-597,599-600,602-603,605-606,608-609,611-612,614-615,617-618,620-621,623-624,626-627,629-630,632-633,635-636,638-639,641-642,644-645,647-648,650-651,653-654,656-657,659-660,662-663,665-666,668-669,671-672,674-675,677-678,680-681,683-684,686-687,689-690,692-693,695-696,698-699,701-702,704-705,707-708,710-711,713-714,716-717,719-720,722-723,725-726,728-729,731-732,734-735,737-738,740-741,743-744,746-747,749-750,752-753,755-756,758-759,761-762,764-765,767-768,770-771,773-774,776-777,779-780,782-783,785-786,788-789,791-792,794-795,797-798,799,800,801,802,803,804,805,806,807,808,809,810,811,812,813,814,815,816,817,818,819,820,821,822,823,824,825,826,827,828,829,830,831,832,833,834,835,836,837,838,839,840,841,842,843,844,845,846,847,848,849,850,851,852,853,854,855,856,857,858,859,860,861,862,863,864,865,866,867,868,869,870,871,872,873,874,875,876,877,878,879,880,881,882,883,884,885,886,887,888,889,890,891,892,893,894,895,896,897,898,899,900,901,902,903,904,905,906,907,908,909,910,911,912,913,914,915,916,917,918,919,920,921,922,923,924,925,926,927,928,929,930,931,932,933,934,935,936,937,938,939,940,941,942,943,944,945,946,947,948,949,950,951,952,953,954,955,956,957,958,959,960,961,962,963,964,965,966,967,968,969,970,971,972,973,974,975,976,977,978,979,980,981,982,983,984,985,986,987,988,989,990,991,992,993,994,995,996,997,998,999]
normal-a100-40	up	2-00:00:00	1	drain	gnx511
normal-a100-40	up	2-00:00:00	3	alloc	gnx[502-504]
normal-a100-40	up	2-00:00:00	13	idle	gnx[506,510,512-513,516,518-519,521-522,524-525,527-528,530-531,533-534,536-537,539-540,542-543,545-546,548-549,551-552,554-555,557-558,560-561,563-564,566-567,569-570,572-573,575-576,578-579,581-582,584-585,587-588,590-591,593-594,596-597,599-600,602-603,605-606,608-609,611-612,614-615,617-618,620-621,623-624,626-627,629-630,632-633,635-636,638-639,641-642,644-645,647-648,650-651,653-654,656-657,659-660,662-663,665-666,668-669,671-672,674-675,677-678,680-681,683-684,686-687,689-690,692-693,695-696,698-699,701-702,704-705,707-708,710-711,713-714,716-717,719-720,722-723,725-726,728-729,731-732,734-735,737-738,740-741,743-744,746-747,749-750,752-753,755-756,758-759,761-762,764-765,767-768,770-771,773-774,776-777,779-780,782-783,785-786,788-789,791-792,794-795,797-798,799,800,801,802,803,804,805,806,807,808,809,810,811,812,813,814,815,816,817,818,819,820,821,822,823,824,825,826,827,828,829,830,831,832,833,834,835,836,837,838,839,840,841,842,843,844,845,846,847,848,849,850,851,852,853,854,855,856,857,858,859,860,861,862,863,864,865,866,867,868,869,870,871,872,873,874,875,876,877,878,879,880,881,882,883,884,885,886,887,888,889,890,891,892,893,894,895,896,897,898,899,900,901,902,903,904,905,906,907,908,909,910,911,912,913,914,915,916,917,918,919,920,921,922,923,924,925,926,927,928,929,930,931,932,933,934,935,936,937,938,939,940,941,942,943,944,945,946,947,948,949,950,951,952,953,954,955,956,957,958,959,960,961,962,963,964,965,966,967,968,969,970,971,972,973,974,9

7.2. Scripts PyCCD (utilizando MPI)

Todos os scripts do PyCCD podem ser encontrados no repositório GitHub <https://github.com/manuelcampagnolo/S2CHANGE/>. O script abaixo é o principal do modelo, desenvolvido para dividir os pixels e executar o processamento do CCD de maneira distribuída, utilizando múltiplos nós e CPUs. Para fazer correr este script, utiliza-se o ficheiro `submit_job.sh`

Logo abaixo, encontra-se o ficheiro `submit_job.sh`, responsável por executar o script principal do PyCCD num ambiente distribuído, dividindo o processamento entre diversos nós e CPUs.

7.2.1. Script Main.py

```
# -*- coding: utf-8 -*-
import os
import platform

# Verifica o sistema operacional
# Windows
if platform.system() == "Windows":
    user_profile = os.environ['USERPROFILE']
    directory_path = os.path.join(user_profile, 'Desktop', 'S2CHANGE')
else: # Linux
    user_home = os.path.expanduser("~")
    directory_path = os.path.join(user_home, 'S2CHANGE')
os.chdir(directory_path)
import pandas as pd
import rasterio
import sys
from pathlib import Path
# Assumir onde esta a pasta dos scripts do PyCCD
PASTA_DE_SCRIPTS = Path(__name__).parent.absolute() / 'scripts' / 'pyccd'

if PASTA_DE_SCRIPTS not in sys.path:
    sys.path.append(str(PASTA_DE_SCRIPTS))
import ccd
from datetime import datetime
from shared.processing import check_or_initialize_file, runDetectionForPoint,
create_geodataframe_from_parquet
from shared.utils import fromParamsReturnName, getNumberOfPixelsFromNpy
from tqdm import tqdm
from concurrent.futures import ProcessPoolExecutor
import warnings
warnings.filterwarnings('ignore')
import numpy as np
```



```

#      |---- math_utils.py
#      |---- parameters.py
#      |---- procedures.py
#      |---- qa.py
#      |---- version.py
#      |---- SUBFOLDER notebooks
#      |---- addNewImageToFile.py
#      |---- avaliacao_exatidao_pyccd.py
#      |---- main.py (** ficheiro principal **)
#      |---- plot.py
#      |---- processing.py
#      |---- read_files.py
#      |---- utils.py
# %%
# -----
#      INPUTS
# -----
var = 'THEIA' # choose variable: THEIA or GEE
BDR = 'DGT' # choose variable: DGT or NAV
S2_tile = 'T29TNE' # escolher o tile S2
# Caminho onde estão os dados todos
public_documents = Path('/projects/F202410004CPCAA1/')
# Caminhos para a base de dados de validação
# -> BDR DGT:
BDR_DGT = public_documents / 'BDR_300_artigo' /
'BDR_CCDC_TNE_Adjusted.shp'
# -> BDR NAVIGATOR:
BDR_NAVIGATOR = public_documents / 'BDR_Navigator' / 'nvg_2018_ccd.gpkg'

# -> IMAGENS SENTINEL:
FOLDER_THEIA = public_documents / 'imagens_Theia' # Caminho dados THEIA
FOLDER_GEE = public_documents / 's2_images' # Caminho dados GEE

if var == 'THEIA':
    tiles = FOLDER_THEIA / S2_tile
else:
    tiles = FOLDER_GEE / S2_tile
# %%
# -----
#      PARAMETROS PRE PROCESSAMENTO
# -----
min_year = 2017 # ano inicial da corrida do CCD
max_date = datetime(2023, 12, 31) # data até onde se corre o ccd

input_bands=['B3', 'B4', 'B8', 'B12']
bands_dict={1: 'NDVI', 2: 'B3', 3: 'B4', 4: 'B8', 5: 'B12'}
bandas_desejadas = bands_dict.keys() # to check

```

```

NODATA_VALUE = 65535
MAX_VALUE_NDVI = 10000

EXECUTAR_PLOT = False # (false para não fazer; true para fazer)
ROW_INDEX = 8 # plot para uma linha do CSV (escolher a linha no row_index)

BATCH_SIZE = 1000 # Ajustar o tamanho do lote para processamento em paralelo

img_collection = tiles.parts[-2]

CRS_THEIA = 32629
CRS_WGS84 = 4326
# -----
#   OUTPUTS
# -----
if BDR == 'DGT':
    BDR_FILE = BDR_DGT
    FOLDER_OUTPUTS = public_documents / 'output_BDR300'
else:
    BDR_FILE = BDR_NAVIGATOR
    FOLDER_OUTPUTS = public_documents / 'output_BDR-NAV'

FOLDER_NPY = FOLDER_OUTPUTS / 'numpy' / S2_tile
FOLDER_PLOTS = FOLDER_OUTPUTS / 'plots' / S2_tile
FOLDER_PARQUET = FOLDER_OUTPUTS / 'tabular' / S2_tile
FOLDER_SHP = FOLDER_OUTPUTS / 'shapefiles' / S2_tile

# Função para criar diretórios se não existirem
def create_directory_if_not_exists(path):
    if not path.exists():
        path.mkdir(parents=True, exist_ok=True)

# Criar os diretórios
create_directory_if_not_exists(FOLDER_NPY)
create_directory_if_not_exists(FOLDER_PLOTS)
create_directory_if_not_exists(FOLDER_PARQUET)
create_directory_if_not_exists(FOLDER_SHP)

# -----
#   PARAMETROS PROCESSAMENTO
# -----
raster_files = sorted(tiles.glob('*.*'), key=lambda f: f.stat().st_size, reverse=True)

raster_path = None
if raster_files:
    largest_file = raster_files[0]

```

```

try:
    with rasterio.open(largest_file) as src:
        if src.read(1).size > 0: # Verificar se a imagem tem dados válidos
            raster_path = largest_file
except:
    raster_path = None # Se houver erro ao abrir, nada é selecionado

# Imprimir o tiff selecionado
if raster_path:
    print("Imagem selecionada:", raster_path)
else:
    print("Nenhuma imagem válida foi encontrada.")
# -----
#   PARAMETROS CCD
# -----
alpha = ccd.parameters.defaults['ALPHA'] # Looks for alpha in the parameters.py
file
ccd_params = ccd.parameters.defaults
##### NOME BASE DOS FICHEIROS A SEREM GERADOS #####
filename = fromParamsReturnName(img_collection, ccd_params, (S2_tile, tiles),
BDR, min_year, max_date)
##### OUTPUTS #####
output_file = FOLDER_NPY / "{}.h5".format(filename) # ficheiro numpy (matriz) dos
dados (nr de imagens x nr de bandas x nr total de pontos)

# -----
#   PARAMETROS DA VALIDAÇÃO
# -----
# datas do filtro das datas da análise (DGT 300)
##### Não alterar #####
dt_ini = '2018-09-12' # data inicial
dt_end = '2021-09-30' # data final
# Margem de tolerância entre a quebra do Modelo e do Analista
theta = 60 # +/- theta dias de diferença
# bandar a filtrar com base na magnitude
bandFilter = None # não é implementado ainda - não mexer
### Configurações MPI
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()
###
# Função para processar um lote
def process_batch(args):
    sel_values_block, xs_slice, ys_slice, tif_dates_ord = args
    arg_list = [
        (i, sel_values_block, tif_dates_ord, xs_slice, ys_slice, NODATA_VALUE,
MAX_VALUE_NDVI, FOLDER_OUTPUTS, CRS_THEIA, CRS_WGS84, img_collection)

```

```

        for i in range(sel_values_block.shape[2])
    ]
    return [runDetectionForPoint(arg) for arg in arg_list]

# Função para processar um lote
def process_single_batch(batch, sel_values_path, xs_path, ys_path, tif_dates_ord,
progress):
    start, end = batch

    # Carregar apenas o bloco específico para o lote
    h5_file = h5py.File(sel_values_path, 'r')
    sel_values_block = h5_file['values'][ :, :, start:end]
    xs_slice = h5_file['xs'][start:end]
    ys_slice = h5_file['ys'][start:end]

    # Processar o bloco específico
    result = process_batch((sel_values_block, xs_slice, ys_slice, tif_dates_ord))

    # Atualizar a barra de progresso compartilhada
    progress.value += 1
    return result

def main(batch_size=None):
    if rank == 0:
        # Processo mestre
        tif_dates_ord, N = check_or_initialize_file(
            output_file, tiles, var, S2_tile, min_year, max_date, BDR_FILE,
            bandas_desejadas, FOLDER_OUTPUTS, img_collection, NODATA_VALUE,
            raster_path
        )

        # Dividir os Índices de dados
        indices = list(range(0, N, batch_size))
        batches = [
            (start, min(start + batch_size, N)) for start in indices
        ]
        print(f"[Rank {rank}] Total de batches criados: {len(batches)}")

        # Dividir lotes entre ranks
        batches_per_rank = [batches[i::size] for i in range(size)]
    else:
        tif_dates_ord = None
        batches_per_rank = None

    # Compartilhar dados entre processos
    tif_dates_ord = comm.bcast(tif_dates_ord, root=0)
    my_batches = comm.scatter(batches_per_rank, root=0)

```

```

# Criar uma barra de progresso compartilhada
with Manager() as manager:
    progress = manager.Value('i', 0) # Contador compartilhado
    total_batches = len(my_batches) # Número total de lotes

    with tqdm(total=total_batches, desc=f"Processo {rank}") as pbar:
        def update_progress(_):
            pbar.n = progress.value # Atualiza a barra de progresso com o valor
            compartilhado
            pbar.refresh()

        # Processar os lotes atribuídos usando ProcessPoolExecutor
        with ProcessPoolExecutor(max_workers=cpus_slurm) as executor:
            futures = [
                executor.submit(
                    process_single_batch,
                    batch,
                    output_file, # Caminho do sel_values
                    str(output_file.with_suffix("")) + '_xs.npy', # Caminho do xs
                    str(output_file.with_suffix("")) + '_ys.npy', # Caminho do ys
                    tif_dates_ord,
                    progress
                )
                for batch in my_batches
            ]
            for future in futures:
                future.add_done_callback(update_progress)

        local_results = [future.result() for future in futures]

# Salvar os resultados localmente por processo
dfs = [df for results in local_results for df in results]
if dfs:
    # Cria um Parquet para cada processo
    rank_parquet_filename = FOLDER_PARQUET /
f'{filename}_rank_{rank}.parquet'
    result_df = pd.concat(dfs, ignore_index=True)
    for col in result_df.columns:
        if result_df[col].apply(lambda x: isinstance(x, list)).any():
            result_df = result_df.explode(col)
    result_df.to_parquet(rank_parquet_filename, index=False)

comm.Barrier() # Sincronizar todos os ranks antes de continuar

if rank == 0:
    all_parquet_files = list(FOLDER_PARQUET.glob(f'{filename}_rank_*.parquet'))

```

```

if not all_parquet_files:
    raise FileNotFoundError(f"Nenhum arquivo encontrado correspondente ao
padrao {filename}_rank_*.parquet em {FOLDER_PARQUET}")

for parquet_filename in all_parquet_files:
    try:
        parquet_file = parquet_filename.stem
        # Função para criar o shapefile para cada Parquet de cada processo
        create_geodataframe_from_parquet(
            parquet_file, CRS_WGS84, CRS_THEIA, S2_tile, FOLDER_PARQUET,
FOLDER_SHP
        )

    except Exception as e:
        print(f"Erro ao processar o arquivo {parquet_file}: {e}")

print(f"Todos os shapefiles individuais foram criados em {FOLDER_SHP}.")

if __name__ == '__main__':
    n = getNumberOfPixelsFromNpy(output_file)
    if rank == 0:
        print(f"Numero total de pixels processados: {n}")
        print(f"Executando com batch_size = {BATCH_SIZE} e n = {n}")
        print(f'Numero de CPUs para o ProcessPoolExecutor: {os.cpu_count()}')
        main(BATCH_SIZE)

```

7.2.2. Ficheiros submit_job.sh

```

#!/bin/bash
#SBATCH --job-name=PyMPI          # Nome do Job
#SBATCH --time=4:00:00           # Tempo de execução (HH:MM:SS)
#SBATCH --nodes=25                # Número de nós (64)
#SBATCH --ntasks-per-node=5       # Número de tarefas por nó (128)
#SBATCH --cpus-per-task=1         # CPUs por tarefa
#SBATCH --partition=normal-x86    # Partição
#SBATCH --account=F202410004CPCAA1X # Conta

# Carregar OpenMPI
module load OpenMPI/4.1.5-GCC-12.2.0 || { echo "Erro ao carregar OpenMPI"; exit
1; }

export OMPI_MCA_orte_base_help_aggregate=0

# Exibir informações do Slurm
echo "Nós alocados: $SLURM_NODELIST"
echo "Tarefas alocadas: $SLURM_NTASKS"

```



```
echo "CPUs por tarefa: $SLURM_CPUS_PER_TASK"  
# Executar o script com MPI  
mpiexec -n $SLURM_NTASKS python -u  
/home/scaetano/CCD_yml_win/S2CHANGE/scripts/pyccd_theia/notebooks/main  
-incd-5-nodes.py
```