RAGNAR HPC CLUSTER

Practical use from JupyterHub

Institute of Astrophysics — Universidad Andrés Bello

1.5 h | Hands-on course | Aug 2026

Notebooks & documentation: `https://astrounab.cl/cluster/`

Support: `ia_cluster@unab.cl`

AGENDA

1 · Access & orientation (15 min)

The cluster (hardware/architecture) · SSH · JupyterHub · software & modules

2 · SLURM & best practices (15 min)

JupyterHub vs batch · sbatch/squeue · parallelism

3 · JupyterHub in practice (20 min)

kernel, FITS, terminal, file browser

4 · Scientific examples (40 min)

4MOST · VVVX · CHANCES (13 min each)

5 · Wrap-up & questions (5 min)

THE RAGNAR CLUSTER

Context: hardware & architecture

THE RAGNAR CLUSTER

Context: hardware & architecture

WHAT IS RAGNAR?

- **High Performance Computing (HPC)** for astrophysics.
- Built to **model and analyse** complex astrophysical phenomena and accelerate scientific discovery.
- **Users:** undergraduate/PhD students, postdocs and faculty at the Institute.
- Supports thesis projects, PhD dissertations and collaborative research.

- Users: undergraduate/PhD students, postdocs and faculty at the Institute.
- Supports thesis projects, PhD dissertations and collaborative research.

HARDWARE RESOURCES

Nodes	CPU cores	Memory	Partitions
ragnar-01–02	28	128 GB	general / standard
ragnar-03–05	20	256 GB	general / bigmem
ragnar-06	40	256 GB	general / bigmem
ragnar-07	24	127 GB	general / standard
Total	**180**	**~1.4 TB**	

- 7 compute nodes (ragnar-01 … ragnar-07) + a login node.
- ragnar-01/02/07: more cores per node · ragnar-03–06: more memory per node.

ARCHITECTURE

> User → **Login node** → **SLURM** scheduler → compute nodes

- One **login node** to get in; the **7 compute nodes** do the work.
- **Centralized storage:** `/mnt/nfs1` and `/scratch`.
- OS: **Rocky Linux 9.7** · scheduler: **SLURM 25.11.5**.

- Centralized storage: `/mnt/nfs1` and `/scratch`.
- OS: Rocky Linux 9.7 · scheduler: SLURM 25.11.5.

PARTITIONS & NODE STATES

- **general*** is the default partition — jobs without an explicit partition go there.
- Also available: **standard** (cores per node) and **bigmem** (maximum memory).
- Node states: **down/mix** (maintenance/partially busy), **alloc**, **idle**.

```
$ sinfo
PARTITION  AVAIL  TIMELIMIT  NODES  STATE  NODELIST
standard     up   30-00:00   2      idle   ragnar-[01-02]
standard     up   30-00:00   1      mix    ragnar-07
bigmem       up   30-00:00   4      idle   ragnar-[03-06]
general*     up   30-00:00   6      idle   ragnar-[01-06]
general*     up   30-00:00   1      mix    ragnar-07
```

STORAGE & FILESYSTEMS

/MNT/NFS1 (HOME)

- Scripts, code, small results.
- **Daily backup**, shared across nodes (~39 TB).

/SCRATCH

- Local to each node; high I/O.
- **Not backed up**; purged regularly.

> Also: `/mnt/nfs2` (datasets: 4MOST, VVVX, CHANCES) and `/mnt/nfs3` — large shared storage.

1 · ACCESS & ORIENTATION

15 min

Connect and find your way around the cluster

15 min

Connect and find your way around the cluster

SSH CONNECTION

```
ssh $USER@172.20.1.138
# or by name:  ssh $USER@ragnar-master
# from outside the university network: use VPN first
```

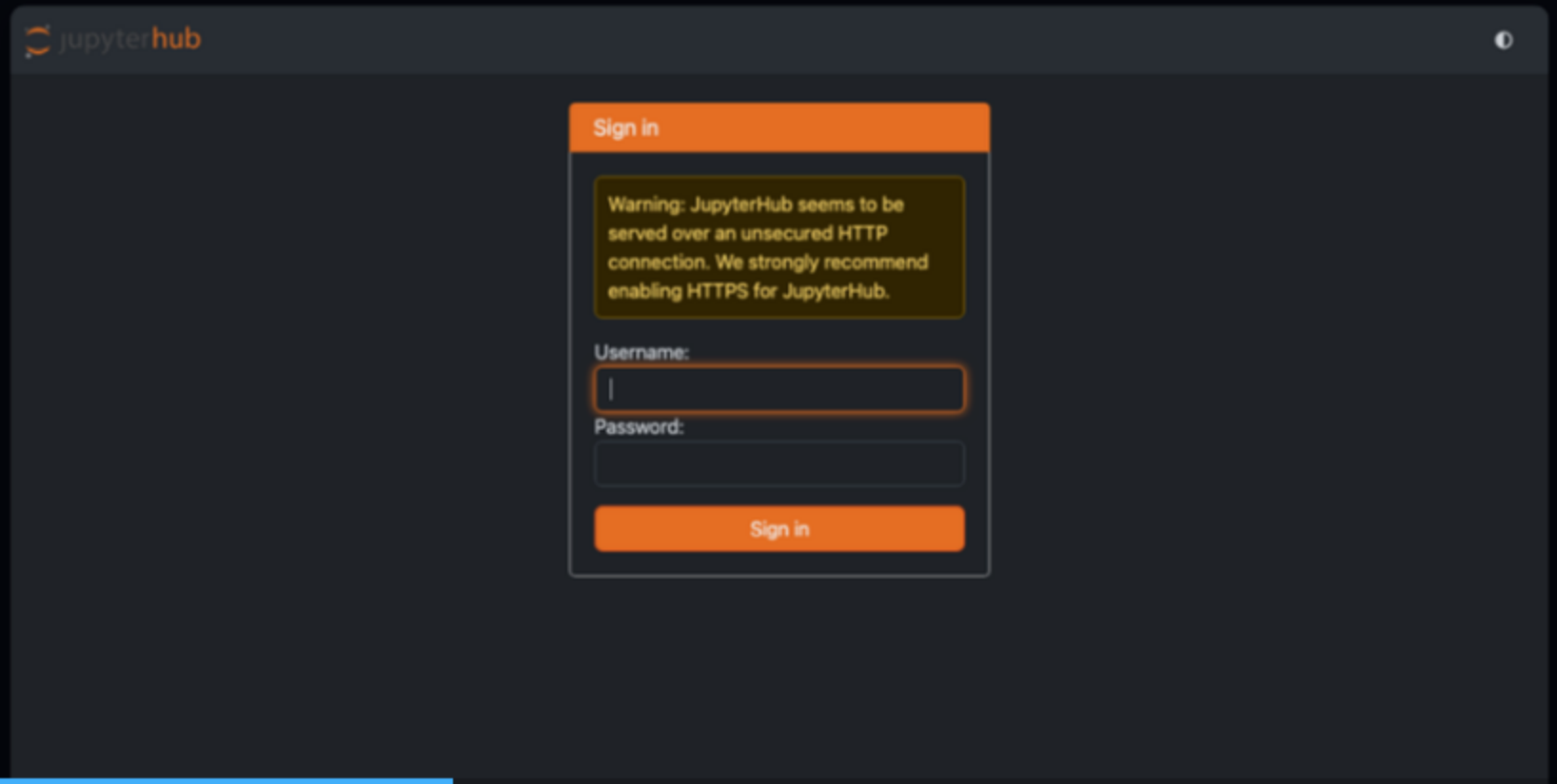
- First login: **change your password**.
- **Etiquette:** never run heavy computations on the login node — only prepare scripts and monitor jobs there.

- Etiquette: never run heavy computations on the login node — only prepare scripts and monitor jobs there.

JUPYTERHUB IN THE BROWSER

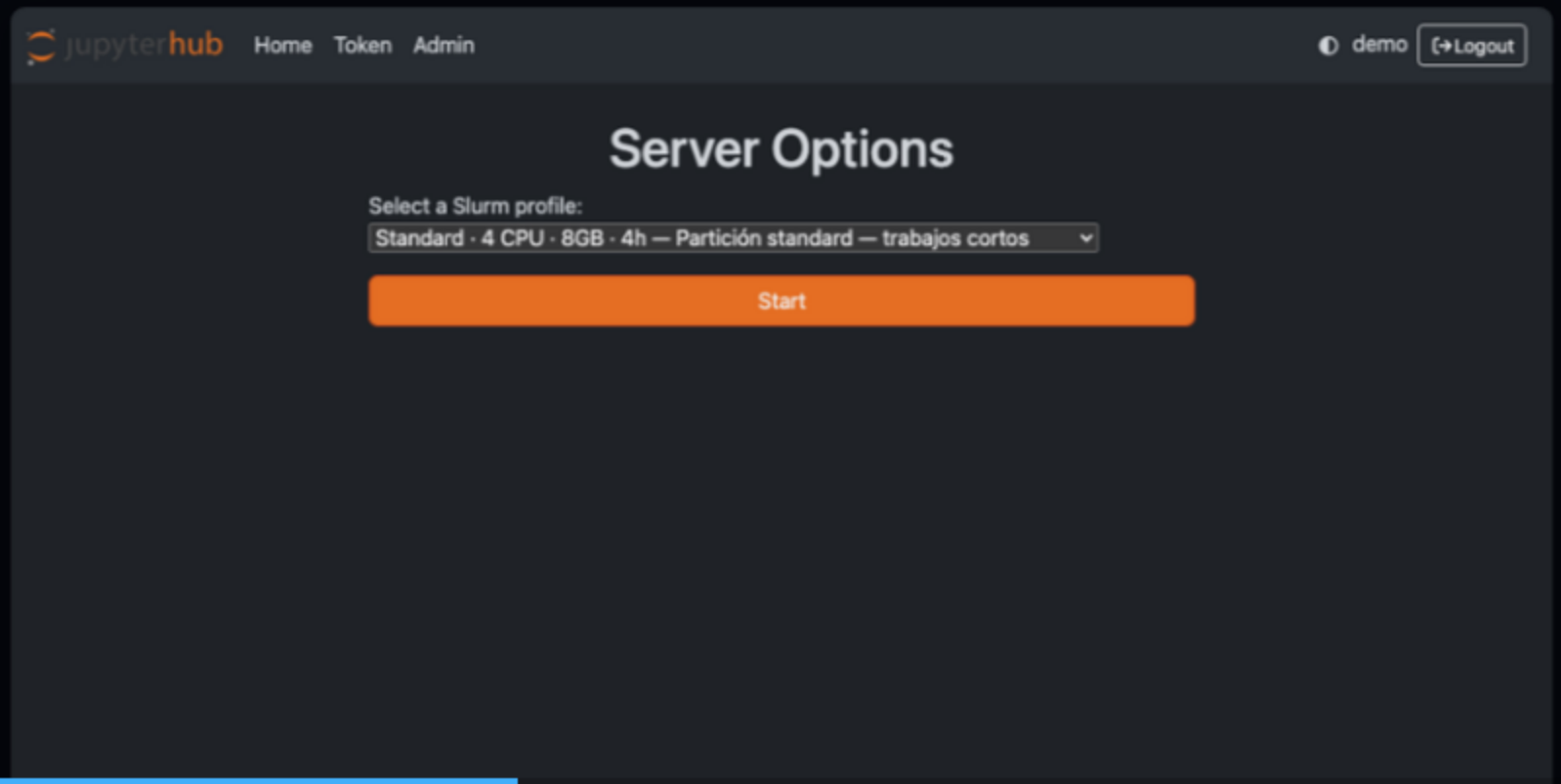
```
http://172.20.1.138:8000   # or: http://ragnar-master
```

- Same account and password as the cluster.
- From any browser on the university network (or VPN).



PROFILE SELECTION AT LOGIN

- On start you pick a profile: **cores**, **memory**, **duration** and **partition**.
- **Standard** 4/8/16 CPU (8–32 GB, 4–24 h) · **BigMem** 8/16 CPU (64–128 GB) · **General** 4 CPU.
- **Rule of thumb:** start small; ask for more only if your problem needs it.



ACCOUNT REQUEST & POLICIES

- Requests and documentation: `https://astrounab.cl/cluster/`
- **Evaluation criteria:** proposal quality, CV, HPC experience and prior contributions.
- **Policies:** resources for scientific purposes only, cite the cluster in publications, non-commercial use.

- Evaluation criteria: proposal quality, CV, HPC experience and prior contributions.
- Policies: resources for scientific purposes only, cite the cluster in publications, non-commercial use.

DIRECTORY LAYOUT

- `/mnt/nfs1/$USER` — your workspace (home, daily backup)
- `/mnt/nfs2/` — project datasets: **4MOST, VVVX, CHANCES**
- `/mnt/nfs3/` — more shared storage
- `/scratch` — fast temporary storage (per node)

> Everything shares storage via **NFS**: what you save at home is visible to all nodes.

SOFTWARE & MODULES

within Block 1

Lmod · EasyBuild · installing your own libraries

SOFTWARE & MODULES

within Block 1

Lmod · EasyBuild · installing your own libraries

AVAILABLE SOFTWARE

Category	Representative modules
Astronomy	Astropy, IRAF, DS9, SExtractor
Languages	Python, Julia, MATLAB, R
Visualization	Gnuplot, Matplotlib, ParaView
Utilities	Git, CMake, MPI, CUDA

Modules are grouped by **toolchain** and architecture, managed with **Lmod**.

Modules are grouped by toolchain and architecture, managed with Lmod.

MODULE MANAGEMENT (LMOD)

```
ml                      # see loaded modules
ml av                   # list everything available
ml spider python        # search a package & its versions
ml Python/3.11.3        # load a version + dependencies
ml purge                # clean the environment
```

- `ml av` groups by toolchain; (D) marks the default versions.

EASYBUILD — REPRODUCIBLE SOFTWARE

- Framework that **automates compilation and installation** of scientific software.
- Reproducibility and dependency management.
- Each version lives in a **separate prefix** → interchangeable modules.
- Missing a package? Ask the administrator with version and details; an **EasyBuild recipe** will be evaluated.

- Missing a package? Ask the administrator with version and details; an EasyBuild recipe will be evaluated.

YOUR OWN LIBRARIES — NO ROOT NEEDED

```
pip install --user <package>
```

- `--user` installs into **your** space (no root).
- The course libraries are already installed: numpy, scipy, pandas, matplotlib, astropy, astroquery, specutils, emcee, corner, joblib, h5py.
- Ideal for pure-Python packages; ask for compiled software if missing.

2 · SLURM & BEST PRACTICES

15 min

When batch, when interactive, and how to do it well

15 min

When batch, when interactive, and how to do it well

WHAT IS SLURM?

- Open-source workload manager: **allocates nodes, launches and schedules jobs**, tracks usage.
- Coordinates **partitions, nodes and tasks** so users share the cluster fairly.
- **De facto standard** on most academic and national supercomputers.

JUPYTERHUB VS. BATCH

Interactive (JupyterHub)

explore, plot, debug

Batch (sbatch)

long or massively parallel compute, no open session needed

explore, plot, debug

long or massively parallel compute, no open session needed

KEY DIRECTIVES IN A JOB SCRIPT

- **--partition** — queue (e.g. `standard`, `general*`, `bigmem`)
- **-n / --ntasks** — MPI processes to launch
- **-c / --cpus-per-task** — threads per task (OpenMP / in-process)
- **--mem** — total RAM of the job
- **--time** — wall clock limit; the job is killed when exceeded

- --mem — total RAM of the job
- --time — wall clock limit; the job is killed when exceeded

JOB SCRIPT — 1 TASK × 20 CORES

```
#!/bin/bash
#SBATCH --job-name=python20
#SBATCH --partition=standard
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=20
#SBATCH --mem=100G
#SBATCH --time=08:00:00
#SBATCH --output=python20_%j.out

module load python/3.11.3
srun python my_script.py
```

Submit with `sbatch script.sh` — monitor with `squeue`.

JOB SCRIPT — 1 TASK × 28 CORES

```
#!/bin/bash
#SBATCH --job-name=python28
#SBATCH --partition=standard
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=28
#SBATCH --mem=128G
#SBATCH --time=12:00:00
#SBATCH --output=python28_%j.out

module load python/3.11.3
srun python heavy_task.py
```

More cores and memory for heavily parallel apps (28-core nodes).

More cores and memory for heavily parallel apps (28-core nodes).

ESSENTIAL COMMANDS & OUTPUTS

```
# Node status (sinfo) — see earlier slide
$ squeue -u $USER
JOBID   USER  PARTITION  TIME      STATE
12345   user  standard   00:12:34  RUNNING
12346   user  standard   00:00:00  PENDING

$ srun --pty bash      # interactive shell on a node
$ scancel 12345        # cancel a job
$ sacct                # history of finished jobs
```

SLURM NOTEBOOK — MANAGE JOBS FROM A NOTEBOOK

Module 2 — SLURM basics and best practices

Ragnar HPC Course | Cluster UNAB | Aug 2026

Goal: learn when to use JupyterHub vs SLURM batch, launch jobs from JupyterHub, monitor their status and follow the cluster's good practices.

1. JupyterHub vs SLURM batch

Feature	JupyterHub	SLURM batch
Interactivity	Yes	No
Max time	Hours	Days
Resources	Limited by session	Up to the node
Best for	Exploration, plotting	Large/long computations

Rule of thumb: if you need to interact, use the notebook; if it takes a long time or is massively parallel, use `sbatch`.

2. Basic SLURM scripts

A SLURM script is a bash script with `#SBATCH` directives. We'll create example scripts from the notebook.

In [ ]:
```
# Create a sample SLURM script
script_slurm = """#!/bin/bash
#SBATCH -J demo_job
#SBATCH -p general
#SBATCH -c 4
#SBATCH --mem=8G
```

PARALLELISM: MPI · OPENMP · HYBRID

MPI (TASKS)

- Independent processes, private memory, message passing across nodes.

```
sbatch --ntasks=4 --cpus-per-task=1 \
       --mem=8G --time=1:00:00 script.sh
```

OPENMP (DATA)

- Threads sharing memory, splitting data on one node.

```
sbatch --ntasks=1 --cpus-per-task=20 \
       --mem=32G --time=2:00:00 script.sh
```

Hybrid: combine both (e.g. `--ntasks=4 --cpus-per-task=7`).

BEST PRACTICES

- Never run heavy computations on the **login node** — submit jobs to SLURM.
- Request **only** the cores/memory/time you need.
- Big temporary data in `/scratch`; code and results in `/mnt/nfs1`.
- Cancel jobs you no longer need (`scancel`).
- Transfer results with `rsync` / `scp`; datasets on `/mnt/nfs2` and `/mnt/nfs3`.
- **Close your JupyterHub session** when done → frees resources.

3 · JUPYTERHUB IN PRACTICE

20 min

Hands-on: your first notebook with real data

20 min

CREATE A NOTEBOOK + CHOOSE THE KERNEL

- **New → Python 3.12** from the Launcher.
- Check the kernel (top right corner).

```
import sys, os, platform
print(sys.version)
print(platform.node())
```

Module 3 — JupyterHub in practice

Ragnar HPC Course | Cluster UNAB | Aug 2026

In this notebook you will learn to:

- Create and run code and markdown cells
- Install libraries without root permissions
- Read FITS files from the shared storage
- Plot astronomical data
- Use the integrated terminal and the file browser

1. Hello JupyterHub

Run the cell below (Shift+Enter or click the Play button). This confirms your Python 3.12 kernel is working correctly.

INSTALL YOUR OWN LIBRARIES

```
pip install --user emcee corner specutils joblib
```

- No root permissions: `--user` installs into your space.
- The cluster is shared → nobody uses root.

```
pip install --user astropy corner ipympl matplotlib
```

- No root permissions: `--user` installs into your space.
- The cluster is shared → nobody uses root.

READ AND PLOT A FITS FILE

```python
from astropy.io import fits
from astropy.wcs import WCS
import matplotlib.pyplot as plt

hdul = fits.open("/mnt/nfs2/4most/espectro.fits")
```

- **Astropy**: the standard astronomy format.
- **Matplotlib**: visualization.

- Astropy: the standard astronomy format.
- Matplotlib: visualization.

INTEGRATED TERMINAL + FILE BROWSER

- **Terminal:** system commands without leaving the hub — `ml av`, `squeue`, `df -h`.
- **File browser:** navigate, upload/download; access to `/mnt/nfs*`.

4 · SCIENTIFIC EXAMPLES

40 min

Three 13-min demos: 4MOST · VVVX · CHANCES

40 min

4.1 — 4MOST: SPECTRAL ANALYSIS

specutils joblib cross-correlation

Module 4.1 — 4MOST: Spectral analysis

Ragnar HPC Course | Cluster UNAB | Aug 2026

Goal: read FITS spectra, compute radial velocities and process many spectra in parallel using `specutils` and `joblib`.

1. Setup

Import the required libraries. `specutils` and `astropy` come with the cluster's Python module.

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from astropy.io import fits
import astropy.units as u
from specutils import Spectrum1D, SpectralRegion
from specutils.analysis import correlation
from joblib import Parallel, delayed
import glob
import pandas as pd
import os

print("Libraries loaded successfully.")
```

2. Read a FITS spectrum

4.1 · READ SPECTRA WITH SPECUTILS

```
from specutils import Spectrum1D
from astropy.io import fits

hdul = fits.open("/mnt/nfs2/4most/espectro.fits")
spec = Spectrum1D.read(hdul)
```

- 4MOST: massive spectroscopic survey.
- `Spectrum1D`: flux + dispersion in one object.

- 4MOST : massive spectroscopic survey.
- `Spectrum1D` : flux + dispersion in one object.

4.1 · RADIAL VELOCITY + PARALLELISM

```python
from joblib import Parallel, delayed

def rv_one(fname):
    # ... cross-correlation with a template ...
    return rv

rvs = Parallel(n_jobs=-1)(delayed(rv_one)(f) for f in files)
```

- Radial velocity by **cross-correlation**.
- **joblib**: many spectra in parallel.

- Radial velocity by cross-correlation.
- joblib: many spectra in parallel.

4.2 — VVVX: INFRARED PHOTOMETRY

astropy | SExtractor | astroquery | Gaia

Module 4.2 — VVVX: Infrared photometry

Ragnar HPC Course | Cluster UNAB | Aug 2026

Goal: visualize IR images, run SExtractor from the notebook, explore the resulting catalog and cross-match with Gaia.

1. Setup

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from astropy.io import fits
from astropy.wcs import WCS
from astropy.visualization import ZScaleInterval, ImageNormalize
from astropy.table import Table
from astropy.coordinates import SkyCoord, match_coordinates_sky
import astropy.units as u
import subprocess
import os
import pandas as pd

print("Libraries loaded.")
```

2. Visualize a VVVX image

VVVX survey images live in `/mnt/nfs2/vvvx/tiles/`. These are Ks-band (near-infrared) images.

4.2 · VISUALIZE AN IR IMAGE (ZSCALE)

```
from astropy.visualization import ZScaleInterval, ImageNormalize

interval = ZScaleInterval()
norm = ImageNormalize(data, interval=interval)
# plt.imshow(norm(data))
```

- **ZScale** highlights the faint structures typical of the IR.

- ZScale highlights the faint structures typical of the IR.

4.2 · SEXTRACTOR FROM THE NOTEBOOK

```
sex imagen.fits -CATALOG_NAME cat.fits
```

```
import subprocess
subprocess.run(["sex", "imagen.fits", "-CATALOG_NAME", "cat.fits"])
```

- Run **SExtractor** (detection/photometry) via `subprocess`.
- Read the catalog and plot a **color–magnitude diagram**.

- Run SExtractor (detection/photometry) via `subprocess`.
- Read the catalog and plot a color–magnitude diagram.

4.2 · CROSS-MATCH WITH GAIA

```
from astroquery.gaia import Gaia
Gaia.ROW_LIMIT = 500
tab = Gaia.query_object_async(cat, radius=5*u.arcsec)
```

- Cross-match positions with the **Gaia** catalog.

4.3 — CHANCES: BAYESIAN MCMC

- Cross-match positions with the Gaia catalog.

emcee | multiprocessing | corner

Module 4.3 — CHANCES: Bayesian MCMC

Ragnar HPC Course | Cluster UNAB | Aug 2026

Goal: load galaxy cluster data, define a model, run MCMC with `emcee` in parallel and visualize the posteriors with `corner`.

1. Setup

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import emcee
import corner
from astropy.io import fits
from astropy.cosmology import FlatLambdaCDM
import astropy.units as u
from multiprocessing import Pool
import os
import time

print("Libraries loaded.")
```

2. Load the cluster data

CHANCES data are stored in `/mnt/nfs2/chances/clusters/`. Each file contains member galaxies of a cluster with positions, velocities and errors.

4.3 · NFW MODEL, LIKELIHOOD & PRIOR

```python
def nfw(r, rho0, rs):
    return rho0 / ((r/rs) * (1 + r/rs)**2)

def log_prior(theta): ...
def log_likelihood(theta): ...
```

- Dark matter density profile **NFW**.

- Dark matter density profile NFW.

4.3 · EMCEE WITH MULTIPROCESSING

```
import emcee, multiprocessing

with multiprocessing.Pool() as pool:
    sampler = emcee.EnsembleSampler(
        nwalkers, ndim, lnprob, pool=pool)
    sampler.run_mcmc(theta0, nsteps)
```

- Spread **walkers** across cores with `multiprocessing.Pool`.

- Spread walkers across cores with `multiprocessing.Pool`.

4.3 · POSTERIORS WITH CORNER

```
import corner
fig = corner.corner(samples, labels=[r"$\rho_0$", r"$r_s$"])
```

- Converging chains and **posterior distributions**.
- Correlations between parameters.

FULL WORKFLOW

1 · Access → **SSH / JupyterHub**

2 · Resources → **profile, modules** (`ml av`)

3 · Compute → **interactive notebook** or **SLURM batch**

4 · Data → **/mnt/nfs1 · /scratch**

5 · Results → **rsync / scp**, then close the session

CONCLUSION & SUPPORT

- Ragnar: **180 cores and ~1.4 TB RAM** for simulations that exceed the laptop.
- Plan with **SLURM**, choose tasks/threads wisely, use **/scratch** for temporary data.
- Use cases: galaxy formation, gravitational waves, catalogs and transient detection.

> Documentation & forms: `https://astrounab.cl/cluster/`
> Support: `ia_cluster@unab.cl`

QUESTIONS

Course notebooks: `https://astrounab.cl/cluster/`

JupyterHub: `http://172.20.1.138:8000`

Thank you! — Institute of Astrophysics · Cluster UNAB