# Using EL-PASO on HPC Clusters

This tutorial explains how to use **EL-PASO** on an **HPC** (High-Performance Computing) cluster that uses **Slurm** for job management. The main idea is to split a large data processing task into many smaller, parallel jobs to make the process faster and more efficient.

---

## 1. The Core Idea: Divide and Conquer

Instead of running one long job to process a huge dataset, we can break down the task into smaller, time-based chunks. For example, a 10-year dataset can be divided into 120 monthly jobs, all of which can be run simultaneously on the cluster. This is called **task parallelism**.

---

## 2. The Tools You'll Use

EL-PASO provides two key scripts to manage this process:

- `submit_slurm_jobs.py` : A Python script that acts as a command-line interface. You tell it a start time, end time, and a chunk size ( `daily` , `monthly` , or `yearly` ), and it automatically creates and submits a Slurm job for each time chunk.
- `job_script_template.sh` : A template for the Slurm job script. This file tells the cluster how to run your processing code for each chunk. You will need to customize this file to point to your specific data processing script.

---

## 3. Step-by-Step Instructions

Follow these three steps to run your EL-PASO tasks on the cluster:

**Step 1: Create Your Processing Script**

Write a Python script that processes data for a specific time range. This script should accept two string arguments: a start time and an end time.

For example, your script, let's call it `process_my_data.py` , should look something like this:

```
import sys
from datetime import datetime
import el_paso as ep
```

```
if __name__ == "__main__":
    start_time_str = sys.argv[1]
    end_time_str = sys.argv[2]
    # Convert string arguments to datetime objects
    start_time = datetime.fromisoformat(start_time_str)
    end_time = datetime.fromisoformat(end_time_str)

    # Your data processing code goes here
    # ...
```

---

**Step 2: Customize the Job Template**

Edit the `job_script_template.sh` file to match your needs.

1. **Update the Slurm options**: Adjust the `#SBATCH` directives to specify the resources (e.g., `CPUs`, `memory`, `time`) your job needs.
2. **Point to your script**: Change the `python your_program.py` line to `python process_my_data.py` (or whatever you named your script).

Here's an example of the modified template:

```
#!/bin/bash
#SBATCH --job-name=elpaso-job
#SBATCH --cpus-per-task=4
#SBATCH --mem=8GB

# Get the start and end times passed from the submission script
START_TIME=$1
END_TIME=$2

# Run your processing script with the time arguments
python process_my_data.py "$START_TIME" "$END_TIME"
```

---

**Step 3: Submit Your Jobs**

Finally, use the `submit_slurm_jobs.py` script from your terminal to submit all the jobs. Specify the total time range you want to process and the desired chunk size.

```
python submit_slurm_jobs.py 2012-01-01 2019-01-01 Monthly
```