Submitting a Job Array

So far, we have submitted individual jobs to the Discovery cluster. But what if we have several independent calculations that we would like the cluster to work on at the same time?

For this exercise, we have a fictional year of daily weather observations in a file named `weather.csv`.

We want to answer five different questions about the data:

```
Array Task   Question
------------ ---------------------------------------------
1            What was the average temperature?
2            What was the highest temperature?
3            What was the lowest temperature?
4            How much total precipitation was recorded?
5            How many days fell below freezing?
```

Rather than submitting five separate batch scripts, we are going to use a **Slurm job array**.

> **Note**
>
> The weather data used in this exercise is fabricated for the workshop. It is not historical weather data.

Inside of your Class_Examples folder you will find another folder called `Array_job`.

Please CD into that directory:

```
cd Array_Job
```

Understanding the Job Array

The batch script `weather_array.sh` contains:

```
#!/bin/bash -l

#SBATCH --job-name=weather
#SBATCH --array=1-5
#SBATCH --cpus-per-task=1
#SBATCH --mem=1GB
```

```
#SBATCH --time=00:05:00
#SBATCH --output=weather_%A_%a.out

echo "Array Job ID: $SLURM_ARRAY_JOB_ID"
echo "Array Task ID: $SLURM_ARRAY_TASK_ID"
echo "Running on: $(hostname)"
echo

python3 analyze_weather.py $SLURM_ARRAY_TASK_ID
```

The important new directive is:

```
#SBATCH --array=1-5
```

This tells Slurm to create an array containing five tasks, numbered 1 through 5 because those are the taskIDs we specified.

Job array IDs do not have to start at 1 or be consecutive. For example, --array=2,4,6 would create three tasks with IDs 2, 4, and 6.

Each task requests **1 CPU** and **1GB of memory**.

SLURM_ARRAY_TASK_ID

Slurm automatically gives each task a number using the environment variable:

```
SLURM_ARRAY_TASK_ID
```

For our array, those values will be `1`, `2`, `3`, `4`, and `5`.

We pass that number to our Python program:

```
python3 analyze_weather.py $SLURM_ARRAY_TASK_ID
```

The Python program uses the task number to decide which analysis to perform:

```
Task 1 ----> Average Temperature ----> result_1.txt
Task 2 ----> Highest Temperature ----> result_2.txt
Task 3 ----> Lowest Temperature -----> result_3.txt
Task 4 ----> Total Precipitation ----> result_4.txt
Task 5 ----> Freezing Days ----------> result_5.txt
```

All five tasks use the same Python program and the same weather dataset, but each task performs a different calculation.

Slurm Output Files

Each array task will also create its own Slurm output file.

This line:

```
#SBATCH --output=weather_%A_%a.out
```

tells Slurm how to name those files.

`%A` represents the main array job ID and `%a` represents the individual array task ID.

For example, if Slurm assigns our array job ID `9509500`, we would get:

```
weather_9509500_1.out
weather_9509500_2.out
weather_9509500_3.out
weather_9509500_4.out
weather_9509500_5.out
```

Combining the Results

Our five array tasks will produce five result files:

```
result_1.txt
result_2.txt
result_3.txt
result_4.txt
result_5.txt
```

We ultimately want to combine those into a single report.

The script `combine_weather.sh` does this:

```
#!/bin/bash -l

#SBATCH --job-name=combine_weather
#SBATCH --cpus-per-task=1
#SBATCH --mem=1GB
#SBATCH --time=00:01:00
#SBATCH --output=combine_weather_%j.out

echo "=== Hanover Weather Analysis ===" > final_weather_report.txt
echo >> final_weather_report.txt

for result in result_1.txt result_2.txt result_3.txt result_4.txt result_5.txt
do
   cat "$result" >> final_weather_report.txt
done
```

```
echo
echo "All five analyses are complete."
echo "Results combined into final_weather_report.txt"
```

There is one problem: `combine_weather.sh` cannot run until **all five array tasks have finished**.

We could sit and wait for them to finish before submitting the combine job, but Slurm provides a better way to handle this.

Job Dependencies

A **job dependency** tells Slurm that one job must wait for another job to finish.

Our `submit_weather.sh` script first submits the job array:

```
ARRAY_JOB_ID=$(sbatch --parsable weather_array.sh)
```

The `--parsable` option allows us to capture the Slurm job ID in the variable `ARRAY_JOB_ID`.

It then submits our combine job:

```
sbatch --dependency=afterok:$ARRAY_JOB_ID combine_weather.sh
```

The important part is:

```
--dependency=afterok
```

This tells Slurm:

> Do not run the combine job until the weather array has completed successfully.

Because the dependency is attached to the entire array, Slurm waits for **all five array tasks** before running `combine_weather.sh`.

Let's Run It!

Now that we understand what the scripts are going to do, let's submit everything:

```
./submit_weather.sh
```

You should see something similar to:

```
Submitted weather job array: 9509500
Submitted combine job: 9509501
The combine job will wait until all five array tasks finish successfully.
```

Quickly check your jobs:

```
squeue -u $USER
```

You may see something similar to:

```
JOBID       PARTITION   NAME           ST   TIME
9509500_1    standard    weather        R    0:12
9509500_2    standard    weather        R    0:12
9509500_3    standard    weather        R    0:12
9509500_4    standard    weather        R    0:12
9509500_5    standard    weather        R    0:12
9509501     standard    combine_weather   PD   0:00
```

There are a couple of interesting things happening here.

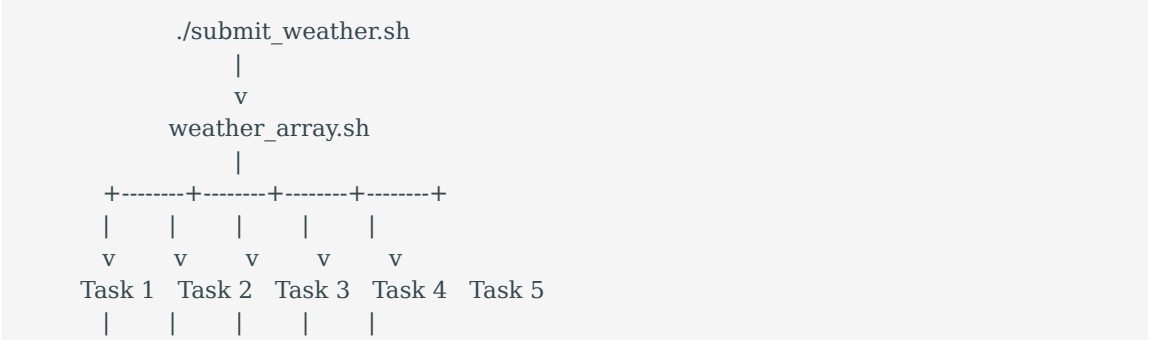
The five `weather` jobs are our **job array**. The number after the underscore identifies each individual array task.

The `combine_weather` job is `PD`, or **Pending**. It is not waiting because the cluster is necessarily busy. It is waiting because we specifically told Slurm that it depends on the weather array finishing successfully.

Once all five array tasks finish, Slurm automatically releases the combine job.

> **Note**
>
> Slurm does not guarantee that the five array tasks will start at the same time, run on different nodes, or run in numerical order. The scheduler places each task wherever the requested resources are available.

What's Going to Happen?



```
      v       v       v       v       v
result_1 result_2 result_3 result_4 result_5
   |       |       |       |       |
   +--------+--------+--------+--------+
                    |
            afterok dependency
                    |
                    v
             combine_weather.sh
                    |
                    v
          final_weather_report.txt
```

Look at the Results

Once the jobs have completed, check the files that were created:

```
ls -1 result_*.txt
```

You should have:

```
result_1.txt
result_2.txt
result_3.txt
result_4.txt
result_5.txt
```

Each file contains the answer calculated by one array task.

Take a look:

```
cat result_*.txt
```

Finally, look at the report created by our dependent job:

```
cat final_weather_report.txt
```

You should see:

```
=== Hanover Weather Analysis ===

Average Temperature: 45.2 F
Highest Temperature: 99.1 F
Lowest Temperature: -6.5 F
Total Precipitation: 41.85 inches
Days Below Freezing: 160
```

With one command, we asked Slurm to perform **five independent calculations in parallel** and then run another job after all five calculations successfully completed.

This same pattern can be used for processing research samples, analyzing many files, running parameter sweeps, simulations, or performing the same analysis across many datasets.