One Thousand Gaussian Calculations, Four Ways…
Choosing an HPC Submission Strategy

Honey S Chauhan
Sr HPC Engineer

Problem: short jobs, long tails

1000

Large submission size

17.83h

Long waits

~5m

mean job duration

Observations –

- Independent files – one node one task small CPU
- Few campaigns causing large pending backlog

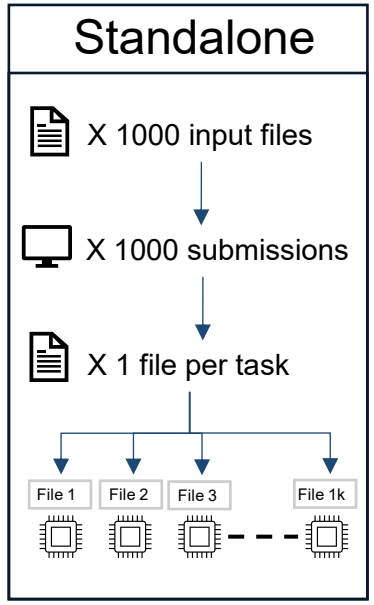
Researchers goal –
I have hundreds of independent calculations, lets finish them quickly

Our Question –
Can we reduce this queue flooding and maintain good run times?

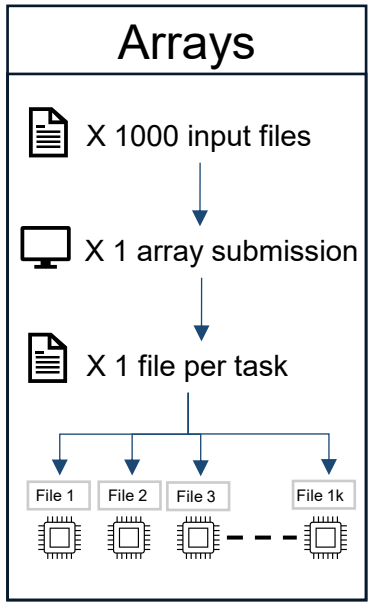
What we tested –

- 100, 500, 1000 identical files
- 4 CPU, 8 GB per job
- Reservation – 96 CPU+300GB
- Serially run: Standalone, Arrays, Bundles, GNU Parallel

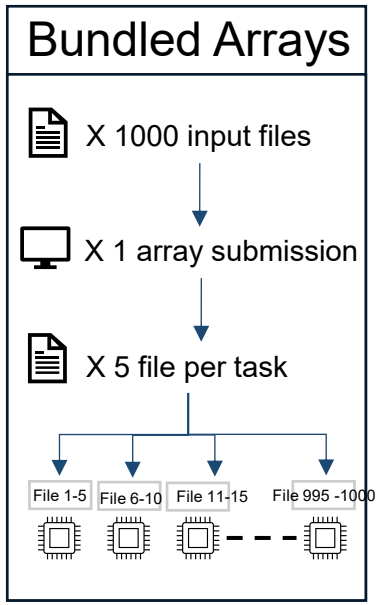
Overview: Submission patterns



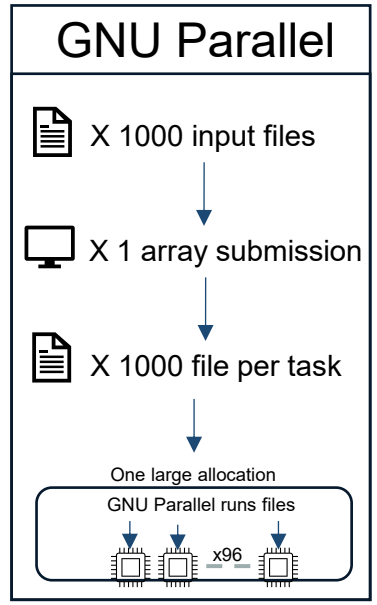
Slurm sees 1000 records → 1000 tasks



Slurm sees 1 record → 1000 tasks



Slurm sees 1 record → 200 tasks



Slurm sees 1 record → 1 big task

A workflow change is only acceptable if the researcher turnaround improves or stays the same.

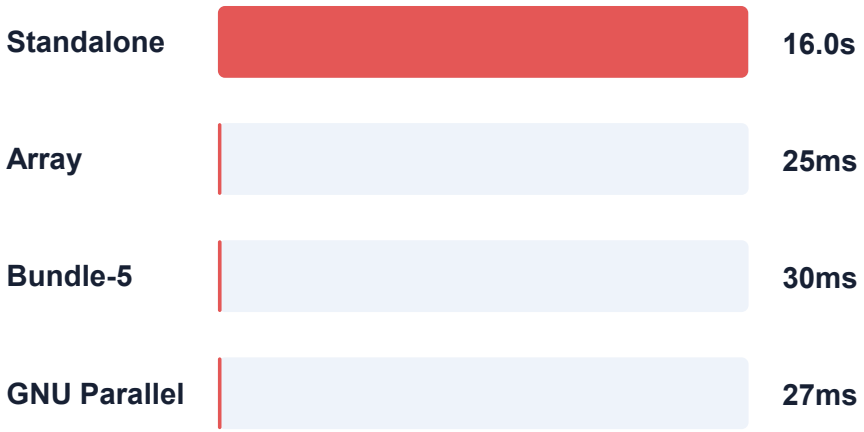
Finding 1: after resources are available, throughput is the same

All main methods are within ~3% overall; standalone, array and GNU Parallel are within ~1.1%.

Finding 2: the hidden scheduler cost

Submission time for 1000 files

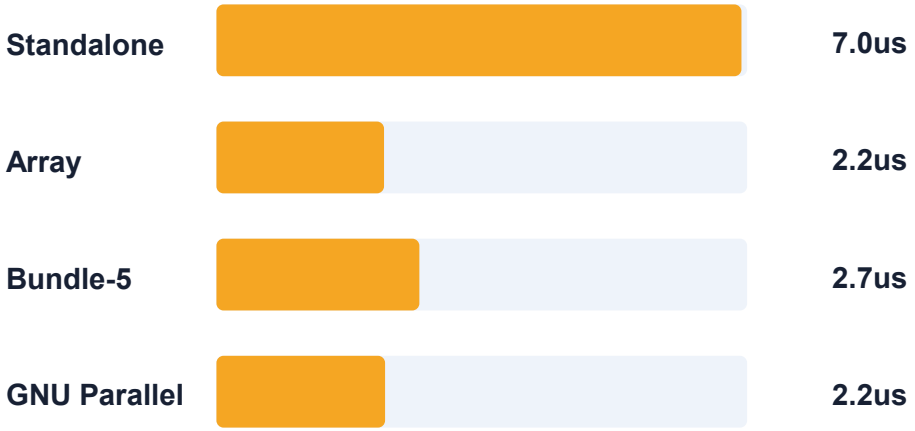
seconds spent issuing the campaign



~600× slower to submit

Mean RPC processing time

how expensive was each controller activity



3.2× more processing time

A practical solution:

Method	What Slurm sees	Best for	Trade-off
Array	1 campaign; each file is a task	default Gaussian campaigns	none measured for throughput
Multiple arrays	several arrays of ≤1000 tasks	>1000 normal files	more parent jobs, same logic
Small bundles	fewer tasks; each runs 2–20 files	10k tiny/similar files	tail imbalance + harder recovery
GNU Parallel	1 larger job allocation	post-processing; existing allocation	waits for a larger block; less Slurm visibility

☞ **The Goal is to reduce queue flooding without asking researchers to accept slower science.**

Questions