



SURF

Workflow tools compared

QCG-PilotJob, HyperQueue, Snakemake and FireWorks

Adapted and updated by Rohit Goswami

Why compare tools

- Many small tasks inside one Slurm allocation need a tool that starts a task as soon as a core is free.
- QCG-PilotJob is one such tool. HyperQueue, Snakemake and FireWorks are three others.
- Each one expresses “this task waits for those tasks” differently.
- The comparison: how the dependency is written, and how much of the allocation sits idle.

Hands-on

[hands-on/alternatives/](#)

The shared exercises

- **Example 4:** one average per CSV file, then an aggregation that must wait for every average.
- **Example 5:** sleeping tasks of different lengths, to measure idle time from the start and end time of each task.
- Same inputs and helper scripts. HyperQueue and Snakemake use 16 tasks; the file-backed FireWorks example uses one worker and one task.
- Job scripts: **average.sh** and **profile.sh** in a directory per tool.

HyperQueue

- A server and a worker run inside the allocation.
- One task per line of **files.txt**.

The dependency

- `--wait` returns when every task has finished; the aggregation is the next line.

one task per line of files.txt

```
hq submit --each-line files.txt \  
    --cpus=1 --wait \  
    --stdout "$averages/{TASK_ID}.csv" \  
    -- bash -c \  
    'python3 ../../example_04/average.py "$HQ_ENTRY"'
```

then, after every task

```
python3 ../aggregate.py "$averages"/*.csv > result.csv
```

Snakemake

- `snakemake --cores $SLURM_NTASKS` runs the rules on the cores of the allocation.

The dependency

- The input of **aggregate** lists every average file.

Snakefile

```
rule aggregate:
    input:
        expand("averages/{name}.csv",
              name=NAMES),
    output:
        "result.csv",
    shell:
        "cat {input} | sort > {output}"
```

FireWorks

- The LaunchPad is a file under \$TMPDIR; no MongoDB server.
- `r launch rapidfire`: one worker. A file LaunchPad allows no more.

The dependency

- `parents=averages` on the `aggregate` Firework.

workflow.py

```
command = shlex.join([sys.executable,
    os.path.join(HERE, "..", "aggregate.py"),
    "--manifest", manifest])
aggregate = Firework(
    ScriptTask.from_str(
        f"{command} > "
        + shlex.quote(os.path.join(
            HERE, "result.csv"))),
    name="aggregate", parents=averages)
lp.add_wf(Workflow(averages + [aggregate],
    name="average"))
```

Results: idle time

One run on two-core course-cluster nodes with 2 slots (FireWorks: its one worker), 32 tasks of example 5 at their original lengths, 25 October 2026. Not a general result.

Tool	Span	Idle
bash pipeline	267 s	37%
QCG-PilotJob	181 s	7%
HyperQueue	172 s	2%
Snakemake	250 s	32%
FireWorks, one worker	344 s	2% of 1 slot

Results: averages of 100 files

The same two-core nodes, 2 slots (FireWorks 1).

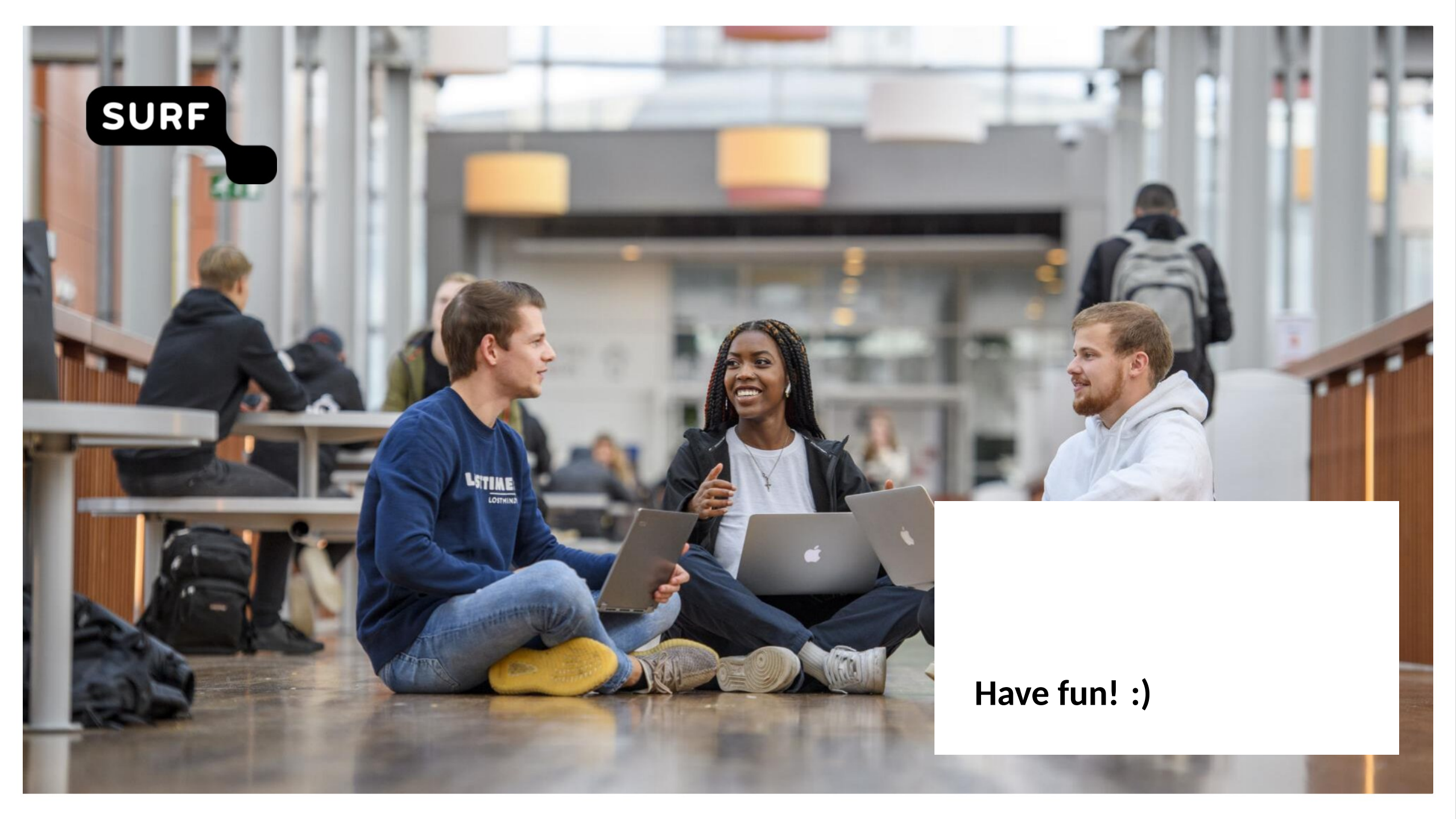
Tool	Time
HyperQueue	11 s
QCG-PilotJob (v4)	18 s
FireWorks, one worker	25 s
Snakemake	67 s

What the idle time comes from

- The bash pipeline runs tasks in blocks of the slot count, so a block waits for its longest task.
- QCG-PilotJob: 7%, a second at the start, three at the end, short gaps between tasks. With a module load in every task an earlier run showed 23%; the examples leave **modules** out.
- HyperQueue: 2%, the next task starts when a slot frees up. Snakemake: the same between tasks, but 77 s of its own start-up before the first one, so 32%.
- FireWorks: one worker on a file LaunchPad, so the tasks run one after another: 344 s.
- Plot lanes show the CPU number a task reported at start. Tasks are not pinned, so overlapping bars can leave white space that is not idle time.
- The idle figure comes from the task start and end times.

How to choose

- **QCG-PilotJob** and **HyperQueue**: many tasks inside one allocation. QCG-PilotJob is driven from Python; HyperQueue from the command line, with a Python API as well.
- **Snakemake**: rules on files; a rerun builds only what changed.
- **FireWorks**: a database of workflows with states and reruns. Without a MongoDB server the file LaunchPad allows one worker: 212 s for 1000 averages on one core on Snellius, against Snakemake's 15 s on 16 cores; alike per core, but FireWorks cannot take more cores.



SURF

Have fun! :)