GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN IN PUBLICA COMMODA SEIT 1737

GWDG

HPS

Davide Mattioli

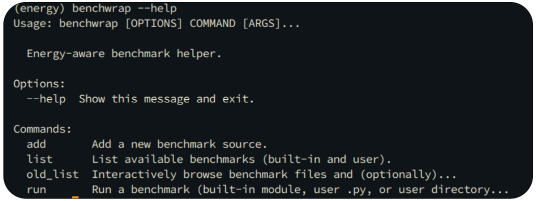
Benchmarking Suite update

August 2025

Table of contents

Benchwrap CLI

- Easy to install and use
- Lunch benchmarks with "benchwrap run"



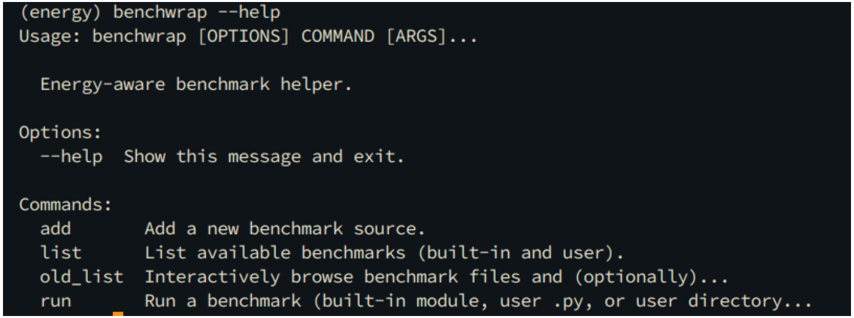
```
(energy) benchwrap --help
Usage: benchwrap [OPTIONS] COMMAND [ARGS]...

  Energy-aware benchmark helper.

Options:
  --help  Show this message and exit.

Commands:
  add       Add a new benchmark source.
  list      List available benchmarks (built-in and user).
  old_list  Interactively browse benchmark files and (optionally)...
  run       Run a benchmark (built-in module, user .py, or user directory...
```

Current commands



```
(energy) benchwrap --help
Usage: benchwrap [OPTIONS] COMMAND [ARGS]...

  Energy-aware benchmark helper.

Options:
  --help  Show this message and exit.

Commands:
  add       Add a new benchmark source.
  list      List available benchmarks (built-in and user).
  old_list  Interactively browse benchmark files and (optionally)...
  run       Run a benchmark (built-in module, user .py, or user directory...
```

Made with click

Click is a Python package for creating beautiful command line interfaces in a composable way with as little code as necessary. It's the "Command Line Interface Creation Kit". It's highly configurable but comes with sensible defaults out of the box.

Image source: `https://click.palletsprojects.com/en/stable/`

Code example

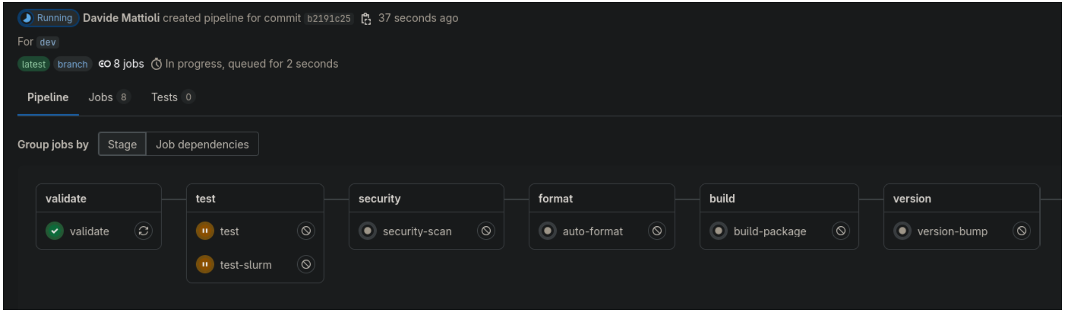
`Simple hello` command implemented in Click

Python

```python
@click.command()
@click.option('--count', default=1, help='Number of greetings.')
@click.option('--name', prompt='Your name',
              help='The person to greet.')
def hello(count, name):
    """Simple program that greets NAME for a total of COUNT times."""
    for x in range(count):
        click.echo(f"Hello {name}!")

```

Dev Ops pipeline

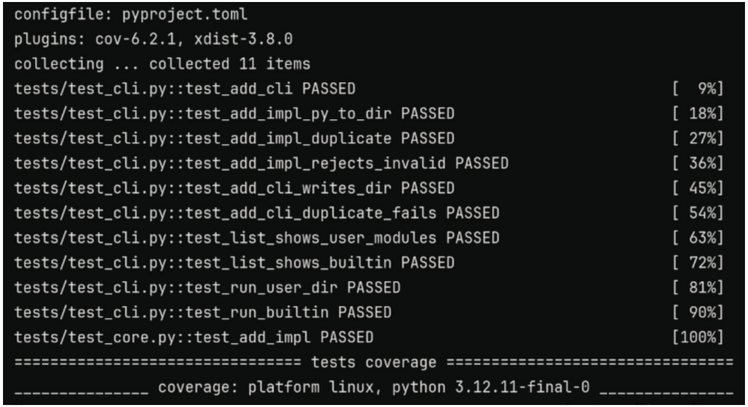


Running Davide Mattioli created pipeline for commit b2191c25 37 seconds ago

For dev

latest branch 8 jobs In progress, queued for 2 seconds

Test



```
configfile: pyproject.toml
plugins: cov-6.2.1, xdist-3.8.0
collecting ... collected 11 items
tests/test_cli.py::test_add_cli PASSED                                         [  9%]
tests/test_cli.py::test_add_impl_py_to_dir PASSED                              [ 18%]
tests/test_cli.py::test_add_impl_duplicate PASSED                              [ 27%]
tests/test_cli.py::test_add_impl_rejects_invalid PASSED                        [ 36%]
tests/test_cli.py::test_add_cli_writes_dir PASSED                              [ 45%]
tests/test_cli.py::test_add_cli_duplicate_fails PASSED                         [ 54%]
tests/test_cli.py::test_list_shows_user_modules PASSED                         [ 63%]
tests/test_cli.py::test_list_shows_builtin PASSED                              [ 72%]
tests/test_cli.py::test_run_user_dir PASSED                                    [ 81%]
tests/test_cli.py::test_run_builtin PASSED                                     [ 90%]
tests/test_core.py::test_add_impl PASSED                                       [100%]
================================ tests coverage ================================
_______________ coverage: platform linux, python 3.12.11-final-0 _______________
```

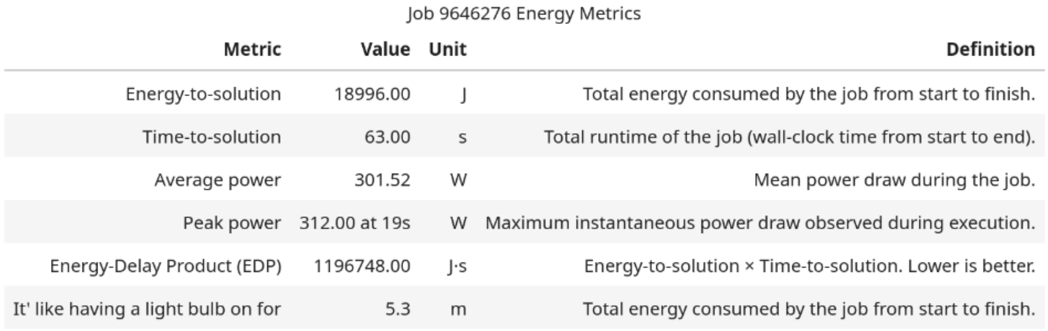
Data

- SLURM: Energy data
- LIKWID: FLOPS information

Bash

```bash
#SBATCH --profile=all
#SBATCH --acctg-freq=1
#SBATCH --acctg-freq=energy=1
module load likwid
DEST="$HOME/.local/share/benchwrap/job_${SLURM_JOB_ID}"
mkdir -p "$DEST"
srun --cpu-bind=cores \
  likwid-perfctr -g FLOPS_DP -t 1s \
  python3 -u -m benchwrap.benchmarks.flops_matrix_mul.workload 1>&2
```

Energy Metrics

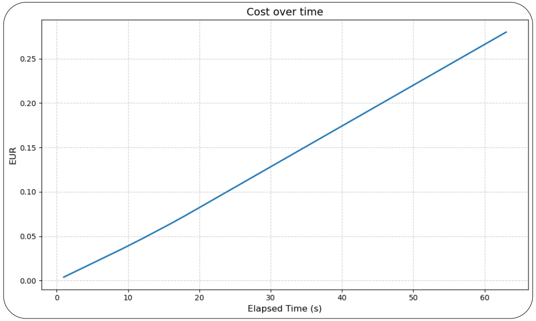


Job 9646276 Energy Metrics

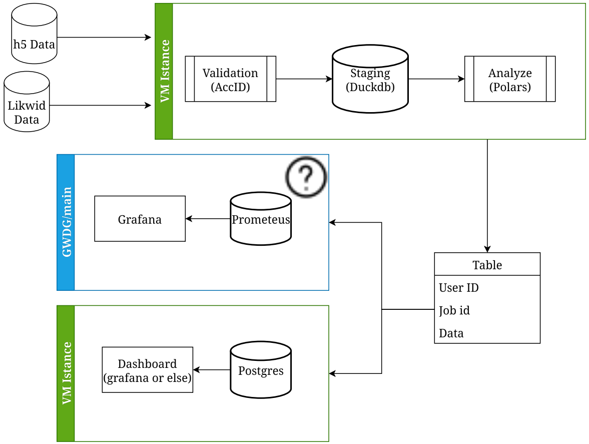
Metric	Value	Unit	Definition
Energy-to-solution	18996.00	J	Total energy consumed by the job from start to finish.
Time-to-solution	63.00	s	Total runtime of the job (wall-clock time from start to end).
Average power	301.52	W	Mean power draw during the job.
Peak power	312.00 at 19s	W	Maximum instantaneous power draw observed during execution.
Energy-Delay Product (EDP)	1196748.00	J·s	Energy-to-solution × Time-to-solution. Lower is better.
It' like having a light bulb on for	5.3	m	Total energy consumed by the job from start to finish.

Cost Metrics

- API calls to the real time energy prices in germany
- possibility of CO2 emission indicator



E2E ETL Pipeline



DuckDB – Transformation Layer

DuckDB

DuckDB – Why (Pros)

- Easy to set up.
- In-proces; zero server; single binary.
- Columnar + vectorized execution.
- Direct Parquet/CSV/Arrow reads.
- Larger-than-RAM via spills and buffer manager.
- Extensions: httpfs/S3, JSON, Spatial; EXPLAIN/ANALYZE for tuning.

DuckDB – Limits (Cons)

- Not suited for high-throughput OLTP or many concurrent writers.
- Single-process scope; cross-process sharing needs extra plumbing.
- Write concurrency/durability simpler than server RDBMS.
- Remote scans bandwidth-bound; partition/row-group layout matters.
- Spill-to-disk can be I/O bound.

PostgreSQL – Serving Layer

PostgreSQL

PostgreSQL —Why

- Role: curated store feeding Grafana; many concurrent reads/writes.
- Aggregation: materialized views for dashboards; scheduled refresh.
- Extensions: TimescaleDB(hypertables+retention)
- Pooling: PgBouncer for Grafana/ETL; keep max connections low.
- Ops: migrations and schema versioning and telemetry

Auth from academic cloud

- How do I use the API?
- Is there a documentation?
- Is it easy to implement?

Dashboard

- Should i use Grafana?
- Does it make sense to include it inside the HPC one?
- Should it be publicly accessible?