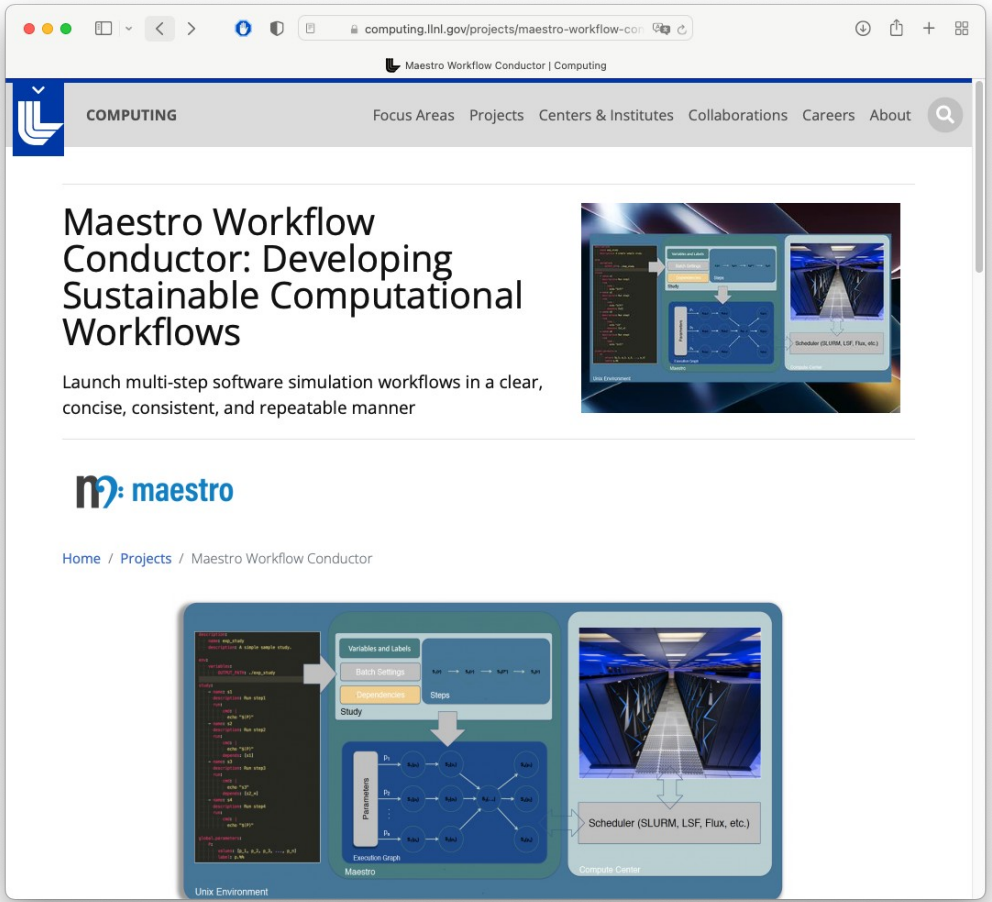
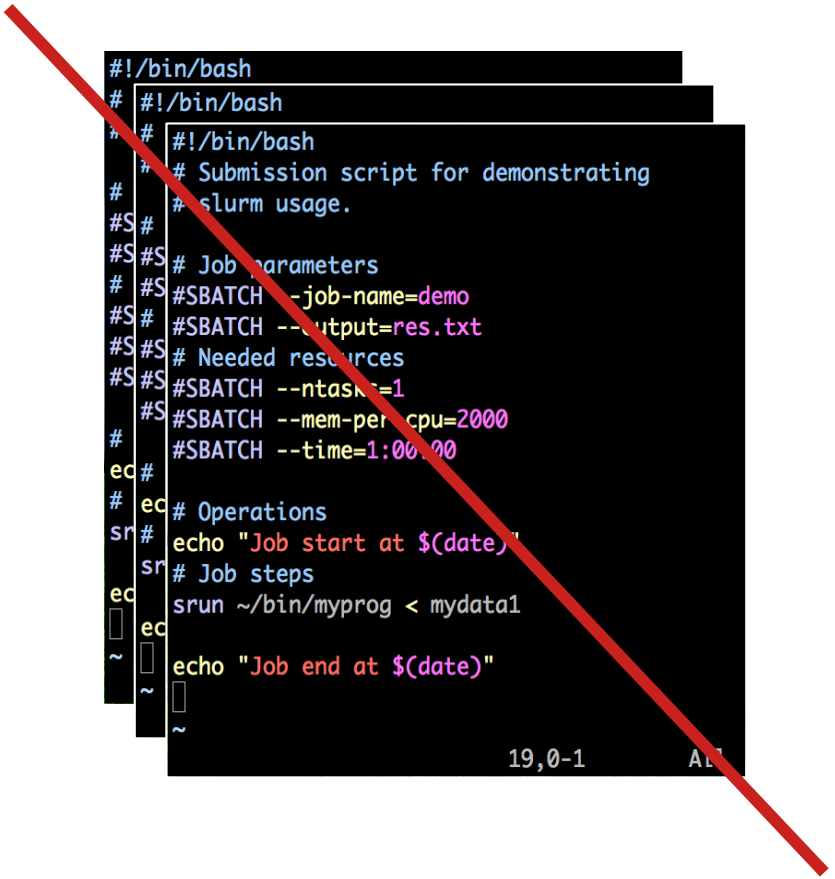
Maestro

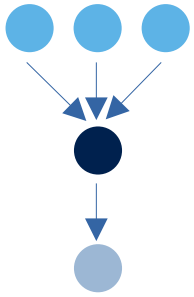


Install with `pip3 install maestrowf` (current version requires Python 3.13)

https://github.com/LLNL/maestrowf

Maestro


#!/bin/bash
#!/bin/bash
#!/bin/bash
Submission script for demonstrating
slurm usage.
Job parameters
#SBATCH --job-name=demo
#SBATCH --output=res.txt
Needed resources
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=2000
#SBATCH --time=1:00:00
Operations
echo "Job start at $(date)"
Job steps
srun ~/bin/myprog < mydata1
echo "Job end at $(date)"
19,0-1



Maestro

```yaml
description:
    name: Build archive
    description: A simple archive building study

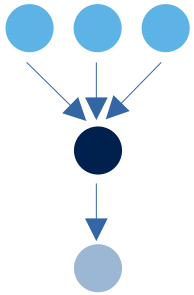
study:
    - name: generate
      description: creates some file
      run:
         cmd: |
           mkdir -p directory
           touch directory/file1.txt
```

Study overview

Study steps
- name
- description
- command(s)

Study: a YAML file with

- information about the study (documentation)
- list of steps and dependencies
- parameters to sweep through
- information about job requirements (scheduler-agnostic)



Maestro

```yaml
description:
    name: Build archive
    description: A simple archive building study

study:
    - name: generate
      description: creates some file
      run:
         cmd: |
           mkdir -p directory
           touch directory/file1.txt
```

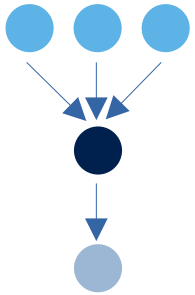
Study overview

Study steps

- name
- description
- command(s)

Maestro

- forces you to document your jobs
- organizes output directories (workspaces)
- monitors and manages (resubmit, etc.) jobs



Maestro

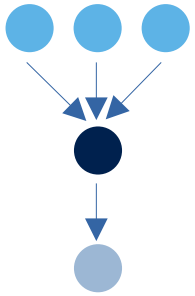
```
$ maestro --help

usage: maestro [-h] [-l LOGPATH] [-d DEBUG_LVL] [-c] [-v] {cancel,run,status} ...

The Maestro Workflow Conductor for specifiying, launching, and managing general
workflows.

positional arguments:  {cancel,run,status}
    cancel              Cancel all running jobs.
    run                 Launch a study based on a specification
    status              Check the status of a running study.

optional arguments:
  -h, --help            show this help message and exit
  -l LOGPATH, --logpath LOGPATH
                        Alternate path to store program logging.
  -d DEBUG_LVL, --debug_lvl DEBUG_LVL
                        Level of logging messages to be output:
                        5 - Critical
                        4 - Error
                        3 - Warning
                        2 - Info (Default)
                        1 - Debug
  -c, --logstdout       Log to stdout in addition to a file. [Default: True]
  -v, --version         show program's version number and exit
```

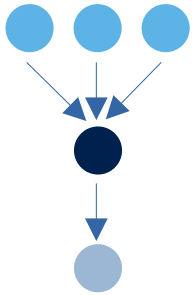


Maestro

```yaml
description:
    name: Build archive
    description: A simple archive building study

study:
    - name: generate
      description: creates some file
      run:
         cmd: |
           mkdir -p directory
           touch directory/file1.txt
```

```
$ maestro run buildarchive.yml
----------------------------------------
Submission attempts =       1
Submission restart limit =  1
Submission throttle limit = 0
Use temporary directory =   False
Hash workspaces =           False
Dry run enabled =           False
Output path =               /[…]/Build_archive_20220126-111006
----------------------------------------
Would you like to launch the study? [yn] y
Study launched successfully.
$
```



Maestro

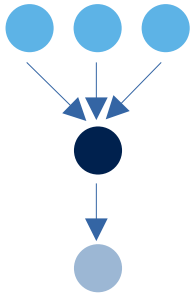
```
description:
    name: Build archive
    description: A simple archive building study

study:
    - name: generate
      description: creates some file
      run:
         cmd: |
           mkdir -p directory
           touch directory/file1.txt
```

View progress with `maestro status <directory>`:

```
$ ls
Build_archive_20220126-111006  buildarchive.yml

$ maestro status Build_archive_20220126-111006
Step Name    Workspace    State     Run Time         Elapsed Time
-----------  -----------  --------  ---------------  ------------
 […]
generate     generate     FINISHED  0d:00h:00m:00s   0d:00h:00m:00s
```



Maestro

Study directory:

```
$ tree Build_archive_20220126-111006
Build_archive_20220126-111006
├── meta
│   ├── environment.yaml
│   ├── metadata.yaml
│   ├── parameters.yaml
│   └── study
│       └── env.pkl
├── logs
│   └── Build archive.log
├── generate
│   ├── directory
│   │   └── file1.txt
│   ├── generate.59117.err
│   ├── generate.59117.out
│   └── generate.sh
├── Build archive.pkl
├── Build_archive.study.pkl
├── Build_archive.txt
├── batch.info
├── status.csv
└── buildarchive.yml
```

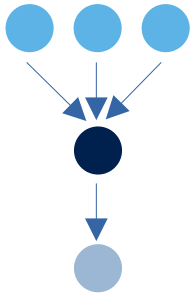
Parameters, environment, etc.

Maestro logs

Workspace

Maestro internal use

Study file



Maestro

```yaml
description:
    name: Build archive
    description: A simple archive building study

study:
    - name: generate
      description: creates some file
      run:
         cmd: |
           mkdir -p directory
           touch directory/file1.txt

    - name: build
      description: creates the tar file
      run:
         cmd: tar cvf archive.tar ../generate/directory
         depends: [generate]

    - name: compress
      description: compress the archive with gzip
      run:
         cmd: |
            ml gzip
            gzip -k $(compressoption) ../../build/archive.tar
         depends: [build]

global.parameters:
    compressoption:
        values: ['--fast', '--best']
        label: COPT.%%
```

Study overview

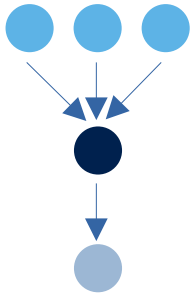
Step 1

Step 2

Step 3

- name
- description
- command(s)
- dependencies

Study parameters



Maestro

```yaml
description:
    name: Build archive
    description: A simple archive building study

study:
    - name: generate
      description: creates some file
      run:
         cmd: |
           mkdir -p directory
           touch directory/file1.txt

    - name: build
      description: creates the tar file
      run:
         cmd: tar cvf archive.tar ../generate/directory
         depends: [generate]

    - name: compress
      description: compress the archive with gzip
      run:
         cmd: |
            ml gzip
            gzip -k $(compressoption) ../../build/archive.tar
         depends: [build]

global.parameters:
    compressoption:
        values: ['--fast', '--best']
        label: COPT.%%
```

Study overview

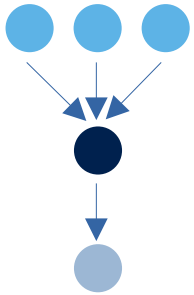
Step 1

Step 2

Step 3

- name
- description
- command(s)
- dependencies

Study parameters



Maestro

```yaml
description:
    name: Build archive
    description: A simple archive building study

study:
    - name: generate
      description: creates some file
      run:
         cmd: |
           mkdir -p directory
           touch directory/file1.txt

    - name: build
      description: creates the tar file
      run:
         cmd: tar cvf archive.tar ../generate/directory
         depends: [generate]

    - name: compress
      description: compress the archive with gzip
      run:
         cmd: |
            ml gzip
            gzip -k $(compressoption) ../../build/archive.tar
         depends: [build]

global.parameters:
    compressoption:
        values: ['--fast', '--best']
        label: COPT.%%
```

Study overview

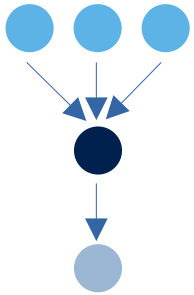
Step 1

Step 2

Step 3

- name
- description
- command(s)
- dependencies

Study parameters



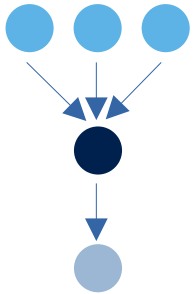
Maestro

Study status:

```
$ maestro status Build_archive_20220126-114700/
Step Name             Workspace    State     Run Time         Elapsed Time
--------------------  -----------  --------  ---------------  -------------
build                 build        FINISHED  0d:00h:00m:00s   0d:00h:00m:00s
compress_COPT.--best  COPT.--best  FAILED    0d:00h:00m:00s   0d:00h:00m:00s
compress_COPT.--fast  COPT.--fast  FINISHED  0d:00h:00m:00s   0d:00h:00m:00s
generate              generate     FINISHED  0d:00h:00m:00s   0d:00h:00m:00s
```

Study directory:

```
$ cd Build_archive_20220126-114700/compress
$ ls
COPT.--best  COPT.--fast
$ cd COPT.--best/
$ ls
compress_COPT.--best.86244.err
compress_COPT.--best.86244.out
compress_COPT.--best.sh
$ cat compress_COPT.--best.86244.err
gzip: ../../build/archive.tar.gz already exists;  not overwritten
```



Maestro

```
description:
    name: Build archive
    description: A simple archive building study

batch:
    type:  slurm
    queue: debug
    host: localhost
    bank: ceci

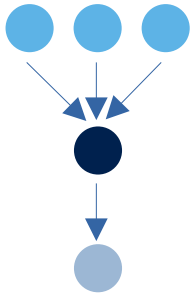
study:
    […]

    - name: compress
      description: compress the archive with gzip
      run:
         cmd: |
            ml gzip
            gzip -k $(compressoption) ../../build/archive.tar
         depends: [build]
         nodes: 1
         procs: 2
         walltime: “10:00”

global.parameters:
    compressoption:
        values: ['--fast', '--best']
        label: COPT.%%
```

Scheduler options

Job options



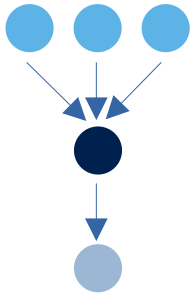
Maestro

Jobs are submitted

```
$ maestro status Build_archive_20220126-120732
Step Name              Workspace    State     Run Time         Elapsed Time
--------------------  -----------  --------  --------------  --------------
compress_COPT.--best  COPT.--best  FINISHED  --:--:--         0d:00h:01m:00s
build                 build        FINISHED  0d:00h:00m:00s  0d:00h:00m:00s
generate              generate     FINISHED  0d:00h:00m:00s  0d:00h:00m:00s
compress_COPT.--fast  COPT.--fast  FAILED    --:--:--         0d:00h:01m:00s

$ sacct

       JobID    JobName  Partition    Account  AllocCPUS      State
------------ ---------- ---------- ---------- ---------- ----------
70382817     compress_+      debug       ceci          1     FAILED
70382817.ba+      batch                  ceci          1     FAILED
70382817.ex+     extern                  ceci          1  COMPLETED
70382818     compress_+      debug       ceci          1  COMPLETED
70382818.ba+      batch                  ceci          1  COMPLETED
70382818.ex+     extern                  ceci          1  COMPLETED
```



Maestro

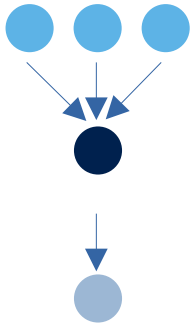
Submission scripts automatically generated:

```
$ cd Build_archive_20220126-120732/compress/COPT.--best/

$ cat compress_COPT.--best.slurm.sh
#!/bin/bash
#SBATCH –nodes=1
#SBATCH –partition=debug
#SBATCH –account=ceci
#SBATCH –time=10:00
#SBATCH –job-name="compress_COPT.--best"
#SBATCH –output="compress_COPT.--best.out"
#SBATCH –error="compress_COPT.--best.err"
#SBATCH --comment "compress the archive with gzip"

ml gzip

gzip -k --best ../../build/archive.tar
```



Maestro

```yaml
description:
    name: Build archive
    description: A simple archive building study

env:
    variables: # Static variables
        OUTPUT_PATH: helloworld
        PARAMS: ...

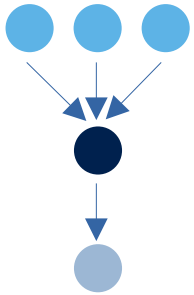


    dependencies:
      git:
        - name: Hello World
          path: ./hello
          url: https://github.com/dfr/helloworld
      paths:
        - name: input data
          path: ./data.txt

study:
    […]
```

Variables

Dependencies (software, cloned automatically) (data, study not launched if non existing)



Maestro

Maestro handles core functions of running a user's workflow

1. Run submission and monitoring

Maestro submits, monitors, and restart jobs. Maestro can also manage the amount of jobs submitted to the scheduler at a given time.

2. Workspace management

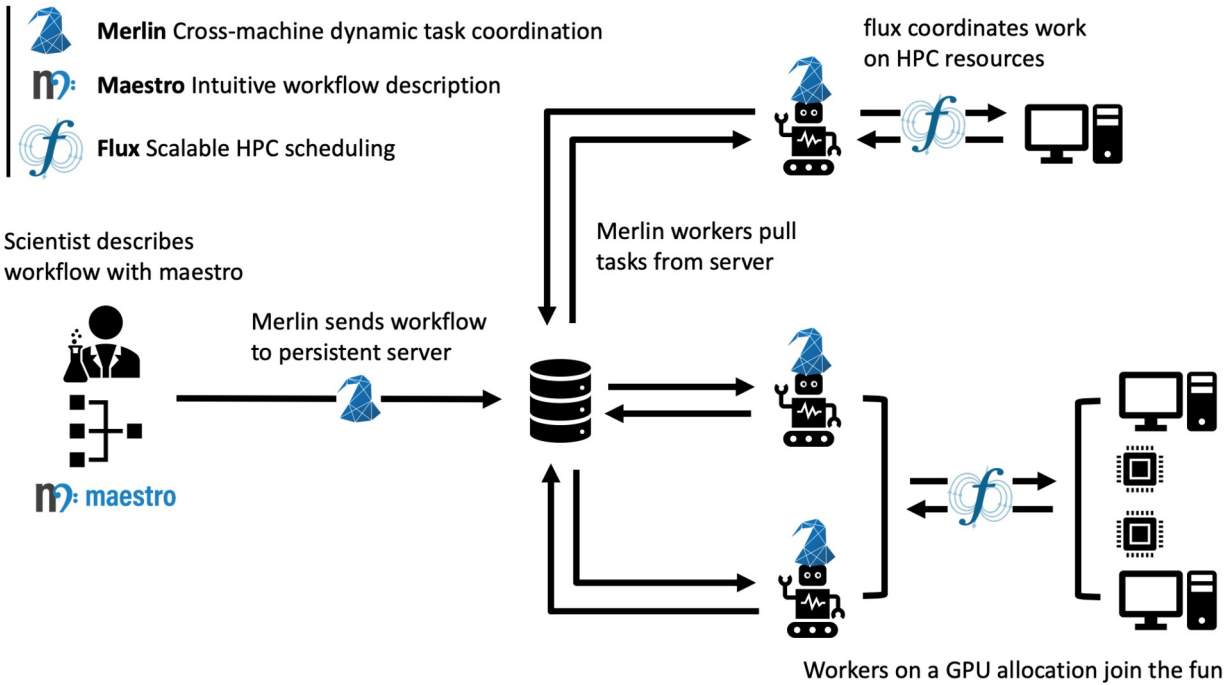
Maestro manages the study workspace creating files and ensuring data doesn't overwrite steps/studies.

3. Workflow Provenance

Maestro captures workflow provenance of what is run including the sampled parameters, study spec, and inputs.

https://computing.llnl.gov/sites/default/files/MaestroWF-slides-final.pdf

Merlin



https://github.com/LLNL/merlin