NHR-NORD@GÖTTINGEN

Workflow Tools

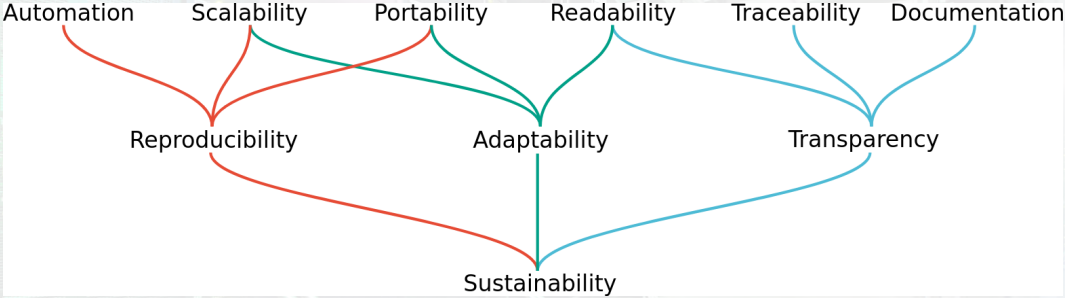
Covalent

Martin Paleico
martin-leandro.paleico@gwdg.de

July 10, 2024

hpc@gwdg.de
GWDG – Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen

Why Workflows?



(From: "Sustainable data analysis with Snakemake", 2021, official Snakemake Paper; and Snakemake Teaching Alliance repo)

More Workflows?

- Snakemake paper has a good overview and categorization of different workflow approaches: `f1000research.com/articles/10-33/v2`
- `github.com/pditommaso/awesome-pipeline`
- See previous presentation for more on Snakemake, Galaxy, Aiida

Covalent

Concept

- Similar in concept to Aiida
- Python based - decorators
- Works from your local PC or on-server, with its own quasi-scheduler daemon and a browser based GUI
- From there, jobs can run locally or on a remote HPC cluster (or another PC)
- Python object based as the unit of communication between individual tasks
- Some rough edges

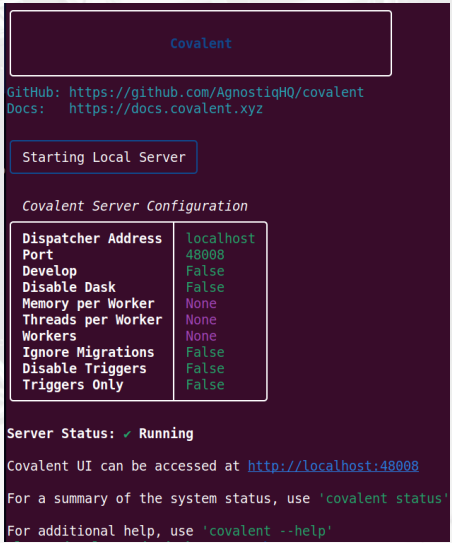
Caveats

- Commercial side
- Python library is open source
- Server/cloud application for enterprises/computing centres which is not
- Never know if they'll drop support, migrate features, etc...

Set up

NHR-NORD@GÖTTINGEN

```
conda create -n covalent_env
    python=3.8
pip3.8 install covalent
covalent start
```



Python Code

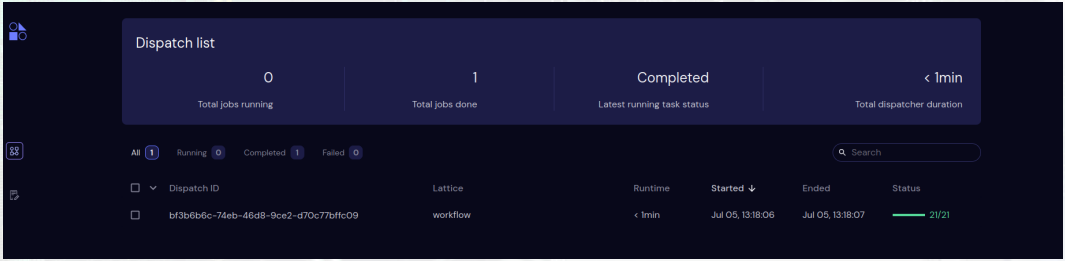
NHR-NORD@GÖTTINGEN

```python
import covalent as ct
# Construct manageable tasks by adding the @ct.electron decorator

@ct.electron
def add(x, y):
    return x + y

@ct.electron
def multiply(x, y):
    return x*y

@ct.electron
def divide(x, y):
    return x/y


# Construct the workflow by stitching
# the electrons with the @ct.lattice decorator
@ct.lattice
def workflow(x, y):
    r1 = add(x, y)
    r2 = [multiply(r1, y) for _ in range(4)]
    r3 = [divide(x, value) for value in r2]
    return r3

# Dispatch the workflow
dispatch_id = ct.dispatch(workflow)(x=1, y=2)
result = ct.get_result(dispatch_id)
print(result)
```

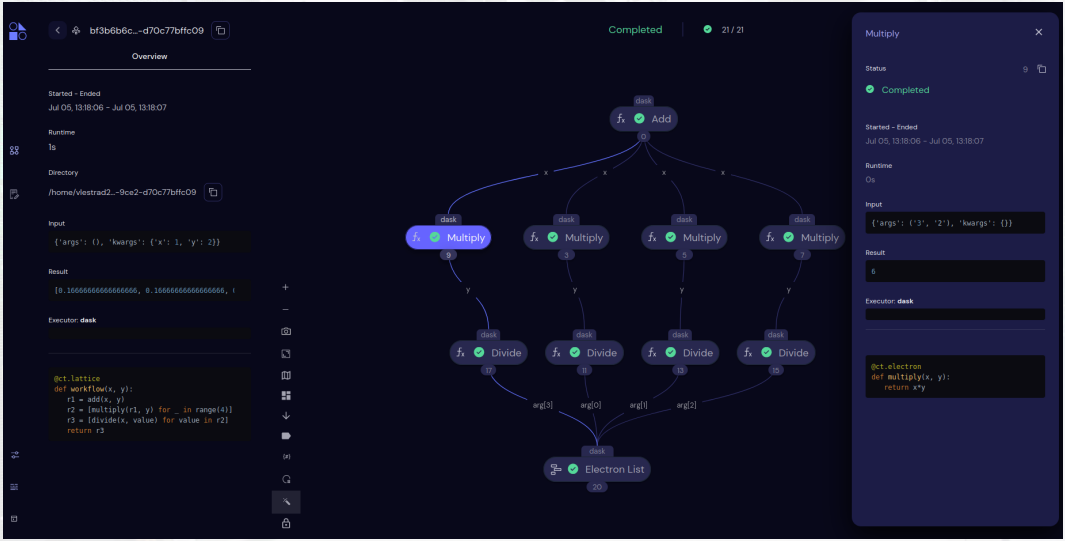
Python Code


```
Lattice Result
==============
status: RUNNING
result: None
input args:
input kwargs: 'x': 1, 'y': 2
error: None

start_time: 2024-07-05 11:18:06.310826
end_time: None

results_dir: /home/username/.cache/
    covalent/results/bf3b6b6c-74eb-46
    d8-9ce2-d70c77bffc09
dispatch_id: bf3b6b6c-74eb-46d8-9ce2-
    d70c77bffc09


Node Outputs
------------
add(0): None
:parameter:1(1): 1
:parameter:2(2): 2
multiply(3): None
:parameter:2(4): 2
multiply(5): None
:parameter:2(6): 2
multiply(7): None
:parameter:2(8): 2
(...)
:postprocess:reconstruct(19): None
:electron_list:(20): None
```

GUI


Dispatch list
0
Total jobs running
1
Total jobs done
Completed
Latest running task status
< 1min
Total dispatcher duration
All 1
Running 0
Completed 1
Failed 0
Search
Dispatch ID
Lattice
Runtime
Started
Ended
Status
bf3b6b6c-74eb-46d8-9ce2-d70c77bffc09
workflow
< 1min
Jul 05, 13:18:06
Jul 05, 13:18:07
21/21

GUI


bf3b6b6c...-d70c77bffc09
Overview
Started - Ended
Jul 05, 13:18:06 – Jul 05, 13:18:07
Runtime
1s
Directory
/home/vlestrad2...-9ce2-d70c77bffc09
Input
{'args': (), 'kwargs': {'x': 1, 'y': 2}}
Result
[0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666]
Executor: dask
@ct.lattice
def workflow(x, y):
r1 = add(x, y)
r2 = [multiply(r1, y) for _ in range(4)]
r3 = [divide(x, value) for value in r2]
return r3
Completed
21 / 21
dask
Add
0
x
Multiply
9
3
6
7
y
Divide
17
11
13
15
arg[3]
arg[0]
arg[1]
arg[2]
Electron List
20
Multiply
Status
9
Completed
Started - Ended
Jul 05, 13:18:06 – Jul 05, 13:18:07
Runtime
0s
Input
{'args': ('3', '2'), 'kwargs': {}}
Result
6
Executor: dask
@ct.electron
def multiply(x, y):
return x*y

Dispatch extras


```python
# Redispatch and modify
@ct.electron
def weighted_sum_xy(x, y):
    return 0.5 * (x + y)

redispatch_id = ct.redispatch(
    dispatch_id,
    replace_electrons={'add':
    weighted_sum_xy},
    reuse_previous_results=True)()
```

```python
# Cancel specific subtasks (and
    dependent subtasks)
ct.cancel(dispatch_id,task_ids=[0, 2])
# Get and reuse results
result = ct.get_result(dispatch_id=
    dispatch_id, wait=True)

# Combine decorators for Sublattices
@ct.electron
@ct.lattice
def cart_dist(x=0, y=0):
    x2 = square(x)
    y2 = square(y)
    sum_xy = add(x2, y2)
    return sqroot(sum_xy)
```

Other features...

- enforce software/environment dependencies
- dynamic responsive workflows
- workflow triggers

Leptons: Running non-Python code

NHR-NORD@GÖTTINGEN

```python
import covalent as ct
from pathlib import Path

#Get abs. paths
prot_name  = "1aki"
init_pdb   = Path('./input_files/{}.pdb'.format(prot_name)).resolve()
prot_name_upper = prot_name.upper()
clean_pdb  = Path('./input_files/{}_clean.pdb'.format(prot_name_upper)).resolve()

do_clean_pdb = ct.Lepton(
    language="bash",
    command="grep -v HOH {f_1} > {f_2}",
    argtypes=[
        (str, ct.Lepton.INPUT_OUTPUT),
        (str, ct.Lepton.INPUT_OUTPUT),
    ],
    named_outputs=["f_1", "f_2"],
)

@ct.lattice
def run_gromacs(init_pdb, clean_pdb):
    do_clean_pdb(
        f_1=init_pdb,
        f_2=clean_pdb,
    )
    return 1

dispatch_id = ct.dispatch(run_gromacs)(init_pdb, clean_pdb)
result = ct.get_result(dispatch_id, wait=True)
print(result)
```

File Handling

- Need to be careful with the location of files (absolute vs. relative paths)
- For remote execution, file transfer handlers are available
- All in all, not as suited for file handling as I/O to and from tasks
- Compare this with file-based Snakemake
- **Tasks should interact through Python objects**
- Which of course can replace files if you are smart about it

SLURM plug in

Running remote: Executors - SLURM

NHR-NORD@GÖTTINGEN

```
Local:
pip install covalent-slurm-plugin
# current asyncssh v. 2.15.0 breaks the plug in
pip install --force-reinstall -v "asyncssh==2.14.0"

Remote:
# need a conda env. with the same package versions:
conda create -n covalent_env python=3.8
conda activate covalent_env
# make sure this is the pip from the above env.
pip install covalent cloudpickle
```

SLURM Executor: Config

NHR-NORD@GÖTTINGEN

```
# Local: vi ~/.config/covalent/covalent.conf
[executors.slurm]
username = "remoteusername"
address = "glogin.hlrn.de"
ssh_key_file = ""
remote_workdir = "/home/remoteuname/covalent-workdir"
cache_dir = "/some/local/path"
conda_env = "covalent_env" #remote
create_unique_workdir = true
prerun_commands = []
postrun_commands = []
use_srun = false
srun_append = ""
bashrc_path = "$HOME/.bashrc" #remote, might error if this can't be loaded
    correctly
slurm_path = "/usr/bin" #remote
poll_freq = 30 #every this many seconds
cleanup = false # set to false for debugging and testing
ignore_versions = true # checks version compat. of remote and local packages
```

SLURM Executor: Config


```
[executors.slurm.options]
nodes = 1
ntasks = 1
cpus-per-task = 1
partition = "standard96:test"
parsable = ""

[executors.slurm.variables]

[executors.slurm.srun_options]
```

(Thanks to Student Assistant Dhiraj Kumar)

SLURM Executor: Config

- Can also be done at the Python level by defining an executor object
- Config file gets reprocessed by the covalent daemon
- Sometimes might need to stop and start the covalent daemon when modifying
- Config file can also be changed from the covalent browser GUI, and at the code level (ct.get_config, ct.set_config)
- Careful what option are local and which remote, particularly with paths

SLURM Executor: Connection

- Currently, can only use non-encrypted/passwordless SSH keys...
- **DO NOT** use SSH keys without passwords in our systems
- Two options: See appendix slides for more info:
  - Start an SSH tunnel with your usual encrypted SSH key, and connect to that one in the executor options (with an empty key)
  - Comment out some lines in the executor code, unlock your SSH key so it's available in ssh-agent, then use as normal

SLURM Executor: Code

NHR-NORD@GÖTTINGEN

```python
import covalent as ct

executor="slurm"

@ct.electron(executor=executor)
def task1(x):
    return x + 1

@ct.electron(executor=executor)
def task2(x):
    return x * 2

@ct.lattice
def workflow(x):
    y = task1(x)
    z = task2(y)
    return z


dispatch_id = ct.dispatch(workflow)(2)
output = ct.get_result(dispatch_id=
    dispatch_id, wait=True)
print(output)
```

SLURM Executor: Remote side

NHR-NORD@GÖTTINGEN

```
1 /home/remoteusername/covalent-workdir/bd7a129d-027c-48d7-a527-ecd49a47c067/
      node_0
2 slurm-bd7a129d-027c-48d7-a527-ecd49a47c067-0.sh
3 func-bd7a129d-027c-48d7-a527-ecd49a47c067-0.pkl
4 result-bd7a129d-027c-48d7-a527-ecd49a47c067-0.pkl
5 script-bd7a129d-027c-48d7-a527-ecd49a47c067-0.py
6 stdout-bd7a129d-027c-48d7-a527-ecd49a47c067-0.log
7 stderr-bd7a129d-027c-48d7-a527-ecd49a47c067-0.log
```

- slurm.sh: Actually submitted SLURM script (check that your SBATCH directives are correct, if the conda env. is being correctly loaded, etc.)
- script.py: Called from slurm.sh
- func.pkl, result.pkl: serialized Python objects and functions, your electron and its results are contained here
- stdout.log, stderr.log: Usual SLURM log files

Discussion

Takeaways

- Snakemake
  - Script based
  - Local only
  - Robust, developed
- Galaxy
  - GUI/browser based
  - Robust, developed
  - Best as a hosted server
- Covalent
  - Python-based
  - Remote execution possible
  - Best used for pure Python approaches

Covalent Tutorials

- Basics
  - `https://covalent.readthedocs.io/en/latest/concepts/basics.html`
  - `https://docs.covalent.xyz/docs/get-started/quick-start`
- Interesting applied tutorials
  - Electronic Structure Calculation

    `https://docs.covalent.xyz/docs/cloud/tutorials-cloud/simple_energy/`
  - MNIST

    `https://docs.covalent.xyz/docs/user-documentation/tutorials/mnist`
  - Many AI/ML, and Quantum Computing examples also available

Appendix

SLURM Executor: Connection 1: SSH Tunnel

- Create 2 new hosts in your local .ssh/config as in the following slide
- SSH to the first host, the tunnel, don't close the connection; should ask you for your usual authentication method
- (There is also a way of getting the tunnel to go into the background instead of maintaining an open connection)
- Configure covalent SLURM plug in to connect to the second host, point to a fake unused key file (or make modifications to the plugin code similar to method 2)
- To test, the setup, open the tunnel, then attempt to connect to the second host with SSH, should open a session without needing to input any password:
  ssh arbitraryuname@localhost -p 1234

SLURM Executor: Connection 1: SSH Tunnel

NHR-NORD@GÖTTINGEN

```
1   Host covalenttunnel
2   Hostname glogin.hlrn.de
3   User remoteuname
4   IdentityFile ~/.ssh/your-usual-ssh-key
5   IdentitiesOnly yes
6   LocalForward 1234 localhost:22 #-L option on the command line
7   ServerAliveInterval 120
8
9   Host covalentssh
10  Hostname localhost
11  User arbitraryuname
12  Port 1234
```

SLURM Executor: Connection 2: Modified code

- Modify the installed plug in code (/some/path/python3.8/site-packages/covalent_slurm_plugin/slurm.py) at around line 215 as in next slide
- Write the plug in configuration file as you would correctly (correct log in node address, username, and your usual, encrypted SSH key)
- Unlock your encrypted SSH key (many ways, but a normal SSH into the remote server will unlock the key for the duration of your current session and pass it to ssh-agent)
- Test the covalent SLURM plug in normally
- This works because asyncssh by default can retrieve keys from ssh-agent, no need to force an SSH key like the plug in does, so just let asyncssh do its thing
- Tested on Ubuntu LTS 22.04, might not work in other OS's

SLURM Executor: Connection 2: Modified code

NHR-NORD@GÖTTINGEN

```
        #if not self.ssh_key_file:
        #    raise ValueError("ssh_key_file is a required parameter in the Slurm plugin.")

        if self.cert_file and not Path(self.cert_file).exists():
            raise FileNotFoundError(f"Certificate file not found: {self.cert_file}")

        #if not Path(self.ssh_key_file).exists():
        #    raise FileNotFoundError(f"SSH key file not found: {self.ssh_key_file}")

        if self.cert_file:
            client_keys = [
                (
                    asyncssh.read_private_key(self.ssh_key_file),
                    asyncssh.read_certificate(self.cert_file),
                )
            ]
        else:
            #client_keys = [asyncssh.read_private_key(self.ssh_key_file)]
            client_keys=[] #blank key so it goes to ssh-agent
```