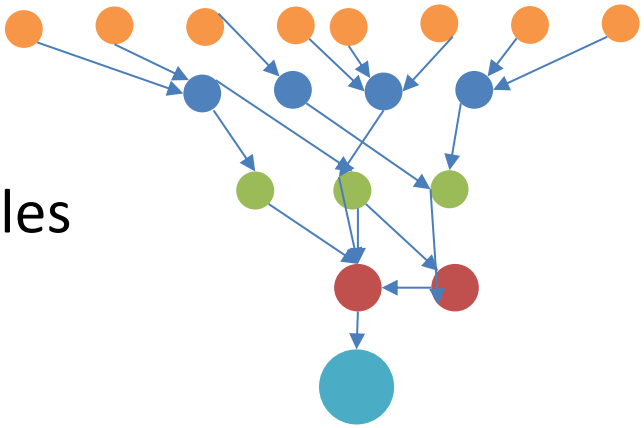
Snakemake

Workflows

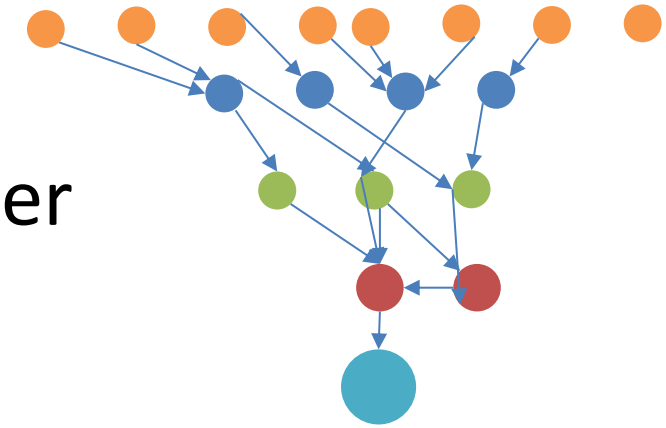
- Imagine the following analysis workflow:
  1. Extract needed infos from data files
  2. Run intermediate analysis on these files
  3. Merge Results
  4. Run some final analysis



- Now imagine:
  1. Final analysis settle (step 4), but you have additional data (step 1)
  2. In the end you realize there‘s a bug in some intermediate analysis (step 2)
  3. You are not sure what the status of the merged files is (step 3)

Workflows

- Can be complicated
- Usually one forgets the details (correlated with time after paper submission)
- Organize the Workflow!
  - Should be able to detect missing/outdated files
  - Should be able to execute on a Cluster
  - My analysis is in Python



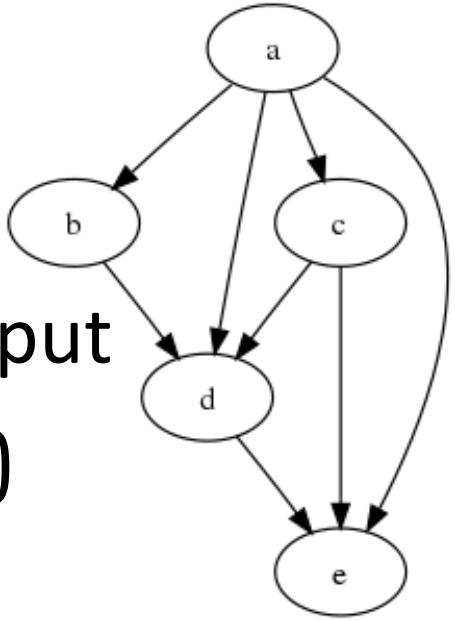
CMake

Luigi

nextflow

Snakemake

- Python package
  https://snakemake.readthedocs.io/en/stable/
- Structure similar to make
- Set of rules:
  - Input
  - Output
- Keeps track of changes in Input/Output
- Builds DAG (Directed acyclic graph)



Snakemake

- Start out simple
  - Set of rules between input and ouptut files

```
rule extract:
        input:
                "input/{run}.dat"
        output:
                "output/{run}.dat"
        shell:
                "python3 extract.py {input} {output}"
```

  - {run} is a wildcard. Rule can be run like

```
djukanov@Dalibors-MBP snakemake % snakemake -c1 output/1.dat
Building DAG of jobs...
MissingInputException in rule extract in file /Users/djukanov/Documents/work/snakemake/Snakefile, line 7:
Missing input files for rule extract:
    output: output/1.dat
    wildcards: run=1
    affected files:
        input/1.dat
```

  - {input} {output} passed as params

Snakemake

- If input files are there:

```
djukanov@Dalibors-MBP snakemake % snakemake -c1 output/1.dat
Building DAG of jobs...
Using shell: /usr/local/bin/bash
Provided cores: 1 (use --cores to define parallelism)
Rules claiming more threads will be scaled down.
Job stats:
job        count    min threads    max threads
-------  -------  -------------  -------------
extract        1              1              1
total          1              1              1

Select jobs to execute...

[Mon Mar 27 11:44:40 2023]
rule extract:
    input: input/1.dat
    output: output/1.dat
    jobid: 0
    reason: Updated input files: input/1.dat
    wildcards: run=1
    resources: tmpdir=/var/folders/8h/wlx1wyj57r5g40vj6mmgs83h0000gn/T

[Mon Mar 27 11:44:40 2023]
Finished job 0.
1 of 1 steps (100%) done
Complete log: .snakemake/log/2023-03-27T114439.136589.snakemake.log
```

Snakemake

- Output depending on multiple files

```
rule average:
        input:
                expand("output/{runs}.dat",runs=range(5))
        output:
                "output/average.dat"
        shell:
                "python3 average.py {input}"
~
```

- expand(…,args) like in python
- Above rule takes five input files as input
- Produces one output file

Snakemake

- rule all, like in Makefile

```
rule all:
        input:
                #expand("output/{runs}.dat",runs=range(5))
                "output/average.dat"
```

- Will invoke all rules to get „output/average.dat“

```
djukanov@Dalibors-MBP snakemake % snakemake -S
Building DAG of jobs...
output_file     date    rule    version log-file(s)     status  plan
output/average.dat      Mon Mar 27 11:38:42 2023        average -               ok      update pending
output/0.dat    Mon Mar 27 11:38:41 2023        extract -               updated input files     update pending
output/1.dat    Mon Mar 27 11:44:40 2023        extract -               updated input files     update pending
output/2.dat    Mon Mar 27 11:38:42 2023        extract -               updated input files     update pending
output/3.dat    Mon Mar 27 11:38:42 2023        extract -               updated input files     update pending
output/4.dat    Mon Mar 27 11:38:42 2023        extract -               updated input files     update pending
```

Snakemake

- Easy running tasks in parallel option -c

```
djukanov@Dalibors-MBP snakemake % snakemake -c 1
Building DAG of jobs...
Using shell: /usr/local/bin/bash
Provided cores: 1 (use --cores to define parallelism)
Rules claiming more threads will be scaled down.
Job stats:
job         count    min threads    max threads
-------  -------  -------------  -------------
all             1              1              1
average         1              1              1
extract         5              1              1
total           7              1              1

Select jobs to execute...
```

Snakemake

- Can also use slurm
- Make a profile
- Run with –p slurm

```
(venv_p3) [djukanov@login21 sigmaterm]$ cat ~/.config/snakemake/slurm/config.yaml
cluster:
  sbatch
    --nodes={resources.nodes}
    --ntasks-per-node={resources.tasks_per_node}
    --mem={resources.mem_mb}
    --time={resources.runtime}
    --job-name={rule}_{wildcards}
    --output={resources.output}
    --partition={resources.partition}
    --account={resources.account}
default-resources:
  - mem_mb=2000
  - runtime=480
  - partition='himster2_exp'
  - account='m2_him_exp'
  - output='slurm-%j.out'
  - tasks_per_node=32
  - nodes=1
jobs: 100
latency-wait: 60
local-cores: 8
max-jobs-per-second: 20
keep-going: True
printshellcmds: True
max-status-checks-per-second: 1
scheduler: greedy
cluster-status: ~/.config/snakemake/slurm/slurm_status.py
```

Snakemake

- Need a clusterstatus script

```
(venv_p3) [djukanov@login21 sigmaterm]$ cat ~/.config/snakemake/slurm/slurm_status.py
#!/usr/bin/env python3
import subprocess
import sys
jobid = ((sys.argv[-1].split(' '))[-1]).rstrip()
output = str(subprocess.check_output("sacct -j %s --format State --noheader | head -1 | awk '{print $1}'" % jobid, shell=True).strip
())
running_status=["PENDING", "CONFIGURING", "COMPLETING", "RUNNING", "SUSPENDED", "PREEMPTED"]
if "COMPLETED" in output:
  print("success")
elif any(r in output for r in running_status):
  print("running")
else:
  print("failed")
```

- Will submit jobs according to dependency
- Will wait for jobs to finish before submitting new ones
- Will run a lot of slurm commands (maybe not te best idea if large number of jobs is used)

Snakemake

- Can use python to construct file lists

```
rule all:
        input:
                expand("raw/{ens}_b2pt_{ll}_disconnected_run_full{pol}{direction}.h5",ens=ensembles,ll=['light','strange'],pol=pols,
direction=[0,1]),
                gen3pt_file_list(glob.glob('input_files/extract_disconnected_*')),
                gen2pt_file_list(glob.glob('input_files/extract_disconnected_*')),
                ['correlators/'+re.sub('r0[0-9]*','',i)+'_cumulated_connected_bin_1.h5' for i in ensembles],
                ['correlators/'+ re.sub('r0[0-9]*','',i) +"_avgs.dat" for i in ensembles],
                ["eff/2/"+re.sub('r0[0-9]*','',i)+"_eff_pruned.h5" for i in ensembles],
                "notebooks/strange_s.ipynb",
                expand("sigma_terms/2/{ens}_{typ}_bootstrap.h5",ens=ensrs,typ=['sigma','sigma0','sigmas']),
                expand("fits/data_{typ}_{var}_cov_all_nodec_latord_1_nosigma0_red_a_corr_nomn_prior.dat",typ=["sum","two"],var=range
(1,13))
```

- Use of parameters & resources:

```
rule get_eff:
        input:
                file="input_files/{ens}_eff.cfg",
                nam="correlators/{ens}_avgs.dat"
        output:
                "eff/{bin}/{ens}_eff_pruned.h5",
                "eff/{bin}/{ens}_eff_ord_mean.h5"
        params:
                par=lambda wc,input:readjson(input.file)
        resources:
                 nodes=1,tasks_per_node=1 , partition='himster2_exp', account='m2_him_exp', output=lambda wcs: f"logs/eff_{wcs.ens}_
bin_{wcs.bin}.out",runtime=240,mem_mb=0
        shell:
                "papermill effective_ff.ipynb -p rands {params.par[randseed]} -p ens {wildcards.ens} -p odir eff/{wildcards.bin}/ -p
 figdir figs/  notebooks/{wildcards.ens}_ff.ipynb"
```

HELMHOLTZ
Helmholtz-Institut Mainz