Orchestrating workflows with Nextflow

Cedric Arisdakessian

2022-09-27

Part I: Overview

Different layers in a workflow

When developing a computational workflow, you usually need to work on different aspects:

- Code (main logic of the workflow)
  - Modules = Which steps?
  - Workflow = How are steps connected to each other?
- Module/workflow parameters
- Resource requirements for each module
- Environment/dependencies (libraries, packages, binaries,...)
- Infrastructure (local computer/HPCC/Cloud)

If you try to implement everything from scratch with base R, it can become very complex and hard to manage.

What do orchestration frameworks do?

- Defines each layer separately to simplify pipeline development:
  Ex: code/infrastructure separation: when you write a module for read trimming, you don't want to worry about which platform it is going to run on.
- Handles the interaction with the infrastructure
  Ex: you don't need to learn how Amazon cloud works, it's taken care of. But you still need to provide mandatory settings:
  - Cloud service: account name (for billing), type of machine you need, ...
  - HPCC: scheduler, names of the queues, container system
- Manages processes:
  - makes sure they execute successfully, report errors otherwise
  - re-run in case a machine unexpectedly fails
  - collect outputs and pipe them to the relevant process

This is a lot to implement by yourself, and having a tested framework that does that for you is a huge help.

What is Nextflow?

Nextflow is a reactive workflow framework and a programming DSL that eases the writing of data-intensive computational pipelines.

- Individual tasks are language agnostic, but the "glue" uses the Groovy language (an extension of Java)
- Installation: `conda install -c bioconda nextflow` or `wget -qO- https://get.nextflow.io | bash`
- Depending on the configuration, you might need `docker` installed (and started)

nf-core

- List of maintained bioinformatics workflows
- Define a standard to implement bioinformatics pipeline with nextflow

nf-core project: https://nf-co.re/

Configuration

Default parameters can be changed through a flag (when calling nextflow) or in the configuration file.

2 main types of parameters:

- "biological" parameters
- computational parameters

nf-core pipelines configuration structure:

- `nextflow.config`: main config, "biological" parameters (among others)
- `conf/base.config`: computing resource specification
- `conf/test_*.config`: test configs, for testing purposes (usually not relevant in our case)
- `conf/modules.config`: defines output folders (within `output_dir`) for each task.
- Potentially more files in `conf/` depending on the developers

Example: https://github.com/nf-core/mag

https://github.com/hawaiidatascience/metaflowmics

Running a nextflow pipeline locally

Running a nf-core pipeline

```
$ nextflow run nf-core/NAME -profile PROFILE PARAMS

# for example
$ nextflow run nf-core/mag -resume -profile docker \
  --input 'MET4-COBRE-WAIMEA/reads/*_R{1,2}.fastq.gz' \
  --outdir nf-mag-outputs \
  --busco_download_path $HOME/data/busco-data
```

Running the metaflowmics pipelines on a local computer

```
$ git clone https://github.com/hawaiidatascience/metaflowmics.git
$ cd metaflowmics/metaflowmics
$ nextflow run Pipeline-16S -profile PROFILE PARAMS
# Example
$ nextflow run Pipeline-16S -resume \
  -profile local,docker \
  --outdir wells.pre-eruption \
  --reads "ikewai/IKE*_R{1,2}.fastq.gz" \
  --skip_subsampling --skip_lulu \
  --pool T
```

Part II: Using Nextflow on a HPCC

Part II: Using Nextflow on a HPCC

Things to consider:

- In Mana (UHM HPCC), you cannot execute any program in the login node, nextflow included. You have 2 options:
  - Start an interactive node and the nextflow command there. Not that if it times out, so does your whole run
  - Run the nextflow pipeline within a SLURM script
- You cannot use docker in Mana (requires admin rights). Instead, you need to use another container technology called Singularity (just another setting)
- By default, you don't have nextflow or any container technology loaded when you ssh in Mana. You will need to install nextflow locally and load the singularity module (`module load tools/Singularity/3.8.5`)
- Your personal home directory is limited. Nextflow saves all the intermediate outputs, so it can take a lot of space. Consider setting the work (`-w`) and output (`--outdir`) directories to the cmaiki-lts drive.

Setting up nextflow

Log on a sandbox node:

```
# 20 min, 1 core, 4GB mem on sandbox queue and run interactive bash session
$ srun --pty -t 20 -c 1 --mem 4G -p sandbox /bin/bash
...
srun: job 40642039 queued and waiting for resources
srun: job 40642039 has been allocated resources
```

Setup conda

```
# load conda module
$ module load lang/Anaconda3
# initialize environment
$ . $(conda info --base)/etc/profile.d/conda.sh
```

Or in your bash profile:

```
$ cat .bash_profile
function conda_on() {
    module load lang/Anaconda3
    . $(conda info --base)/etc/profile.d/conda.sh
$ conda_on
```

Create nextflow environment

```
$ conda_on
$ conda create -n nxf -c bioconda nextflow # create nextflow environment
```

Interactive pipeline run (2h)

```
$ srun --pty -t 120 -c 1 --mem 4G -p shared,exclusive,sandbox /bin/bash
$ module load tools/Singularity # container technology
$ conda_on
$ conda activate nxf # activate
$ nextflow run ...
```

Testing the setup

```
$ git clone https://github.com/hawaiidatascience/metaflowmics.git
$ cd metaflowmics/metaflowmics
$ make test CONF=mana,singularity
```

```
#!/bin/bash
#
#SBATCH --job-name=mfm16S
#SBATCH --error=stderr-mfm16S-%A.err
#SBATCH --output=stdout-mfm16S-%A.out
#
#SBATCH --partition=shared,exclusive,kill-shared,kill-exclusive
#SBATCH --nodes=1
#SBATCH --tasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --time=4-00:00:00
#SBATCH --mem=4G

source ~/.bash_profile

module load "tools/Singularity/3.8.5" # container technology
module load lang/Anaconda3
. $(conda info --base)/etc/profile.d/conda.sh
conda activate nxf
```

```
TYPE=well
cmaiki_dir=$HOME/cmaiki-lts/carisdak
input_dir=$cmaiki_dir/data/KML/16S/ikewai/per-type/${TYPE}
output_dir=$cmaiki_dir/nf-pipelines-outputs/ikewai-${TYPE}

src_dir=$HOME/apps/nf-pipelines/metaflowmics/metaflowmics/Pipeline-16S

echo "Starting metaflow|mics pipeline"
echo "Input directory:  ${input_dir}"
echo "Output directory: ${output_dir}"

nextflow run $src_dir \
    -resume -profile mana,singularity \
    -ansi-log false \
    -w $output_dir/work \
    --outdir $output_dir/results \
    --reads "$input_dir/*_R{1,2}.fastq.gz" \
    --pool T \
    --skip_subsampling --skip_lulu --skip_unifrac
```

Running a nf-core pipeline on a HPCC

Similar, but we need to adjust the configuration files since the nf-core team didn't create one for Mana. Which means:

- Saying it's running on SLURM
- Specifying which queues we want to use
- Specify how to load Singularity
- Update CPU and memory limitation

Currently, pull request to add mana configuration to nf-core pipelines.

$\Rightarrow$ you will eventually just need to specify the mana configuration for any nf-core pipeline (i.e. in your nextflow command, use the flag `-profile mana`.

In the meantime, you will need to create a file named mana.config with the following content

```
params {
  max_memory = 400.GB
  max_cpus = 96
  max_time = 72.h
}

process {
  executor = 'slurm'
  queue = 'shared,exclusive,kill-shared,kill-exclusive'
  module = 'tools/Singularity'
}

singularity {
  enabled = true
  cacheDir = "$HOME/.singularity_images_cache"
  autoMounts = true
}
```

Finally, in your nextflow command, use the parameter `-c /path/to/config` to include those parameters

What about cloud computing?

A few supported options:

- Google cloud platform (GCP)
- Amazon cloud

Advantages:

- Easy to scale up or down
- No queueing time
- Complete control over your system (you are admin)
- Can be used from your local computer, interaction with cloud services are handled by nextflow

But:

- Need to setup an account on cloud platform
- Not free (but pretty cheap)