Nextflow

Computational Pipeline Framework

Stephen Kelly

April 4, 2018

NYU Langone Medical Center

New York, New York, USA

Nextflow enables reproducible computational workflows

- Di Tommaso, P., Chatzou, M., Floden, E. W., Barja, P. P., Palumbo, E., & Notredame, C. (2017). *Nextflow enables reproducible computational workflows.* Nature Biotechnology, 35(4), 316–319. doi:10.1038/nbt.3820
- https://www.nature.com/articles/nbt.3820

nextflow

“When analyzing very large data sets, the main source of computational irreproducibility arises from a lack of good practice pertaining to software and database usage”

"In silico workflow management systems are an integral part of large-scale biological analyses"

- Ease of development
- Standardized format
- Consistent structure
- Reproducibility
- Manage complexity

Nextflow

- Domain-specific language
- parallel asynchronous execution
- adaptation of existing pipelines
- built on Groovy (Java 8)
    - no user ‘installation’ required

Key Aspects

- Executes tasks in any scripting language
  - bash, R, Python, etc.
- built-in support for Docker, Singularity, environment modules
- built-in support for AWS, HPC schedulers (SGE, SLURM, LSF, etc.)
- **decoupling of pipeline tasks from task-execution logic and environment management**
  - allows for greater portability
  - Docker + Nextflow = 100% portable pipeline

Table 1 Comparison of Nextflow with other workflow management systems

Workflow	Nextflow	Galaxy	Toil	Snakemake	Bpipe
Platform[a]	**Groovy/JVM**	**Python**	**Python**	**Python**	**Groovy/JVM**
Native task support[b]	Yes (any)	No	No	Yes (BASH only)	Yes (BASH only)
Common workflow language[c]	No	Yes	Yes	No	No
Streaming processing[d]	Yes	No	No	No	No
Dynamic branch evaluation	Yes	?	Yes	Yes	Undocumented
Code sharing integration[e]	Yes	No	No	No	No
Workflow modules[f]	No	Yes	Yes	Yes	Yes
Workflow versioning[g]	Yes	Yes	No	No	No
Automatic error failover[h]	Yes	No	Yes	No	No
Graphical user interface[i]	No	Yes	No	No	No
DAG rendering[j]	Yes	Yes	Yes	Yes	Yes
Container management					
Docker support[k]	Yes	Yes	Yes	No	No
Singularity support[l]	Yes	No	No	No	No
Multi-scale containers[m]	Yes	Yes	Yes	No	No
Built-in batch schedulers[n]					
Univa Grid Engine	Yes	Yes	Yes	Partial	Yes
PBS/Torque	Yes	Yes	No	Partial	Yes
LSF	Yes	Yes	No	Partial	Yes
SLURM	Yes	Yes	Yes	Partial	No
HTCondor	Yes	Yes	No	Partial	No
Built-in distributed cluster[o]					
Apache Ignite	Yes	No	No	No	No
Apache Spark	No	No	Yes	No	No
Kubernetes	Yes	No	No	No	No
Apache Mesos	No	No	Yes	No	No
Built-in cloud[p]					
AWS (Amazon Web Services)	Yes	Yes	Yes	No	No

Design

- ‘Channels’ and ‘Processes’
  - Channels: uni-directional pipes to pass files, values, data, etc. to processes
  - Processes: tasks to be performed in the pipeline
- Processes executed in isolation from each other

Basic Examples

Pipeline script:

Input Channel

task Process

main.nf

```
1  Channel.from( ['Sample1','Sample2','Sample3','Sample4'] ).set { samples }
2
3  process print_sample {
4      echo true
5
6      input:
7      val(sampleID) from samples
8
9      script:
10     """
11     echo "[print_sample] ${sampleID}"
12     """
13 }
```

output:

```
[2018-04-04 16:13:59]
kellys04@acc38pathlabmac01:~/projects/nextflow-demos/print-samples2$ ./nextflow run main.nf
N E X T F L O W  ~  version 0.28.0
Launching `main.nf` [cranky_brattain] - revision: e13962af7d
[warm up] executor > local
[f7/7bbaa0] Submitted process > print_sample (1)
[3e/b1a9f9] Submitted process > print_sample (4)
[da/a02b84] Submitted process > print_sample (3)
[f4/abcab6] Submitted process > print_sample (2)
[print_sample] Sample4
[print_sample] Sample3
[print_sample] Sample2
[print_sample] Sample1
```

Pipeline output

- ‘work’ directory

  stages process inputs

  & outputs

```
[2018-04-04 16:25:31]
kellys04@acc38pathlabmac01:~/projects/nextflow-demos/print-samples2$ tree work/
work/
|-- 0c
|   `-- 1fd762656b4448b5e67cef8a4c7923
|-- 10
|   `-- e14741844aec9706e5bc25824a0f75
|-- 25
|   `-- bd41cc7622fbb29405f138d3d61dea
|-- 3e
|   `-- b1a9f9d042e6c5fa74dcf2e57d593b
|-- 64
|   `-- bb6781e52ea53bb542cd1c157cc3b7
|-- 70
|   `-- 88ea030e5a155e1cd20fefb7ee20c4
|-- a8
|   `-- f1033b202aea42bcc665a2059b2e46
|-- cc
|   `-- b7578c4c6dffb56d22f5e9ff99a235
|-- da
|   `-- a02b847378a3f9437015efec909aa5
|-- e5
|   `-- aec0428a89c655de02e16ed2817ecc
|-- f4
|   `-- abcab6f2a2e5b1503c9493a3445c4e
`-- f7
    `-- 7bbaa003d7c4d22cd68ddc0115611d
```

‘work’ dir contents

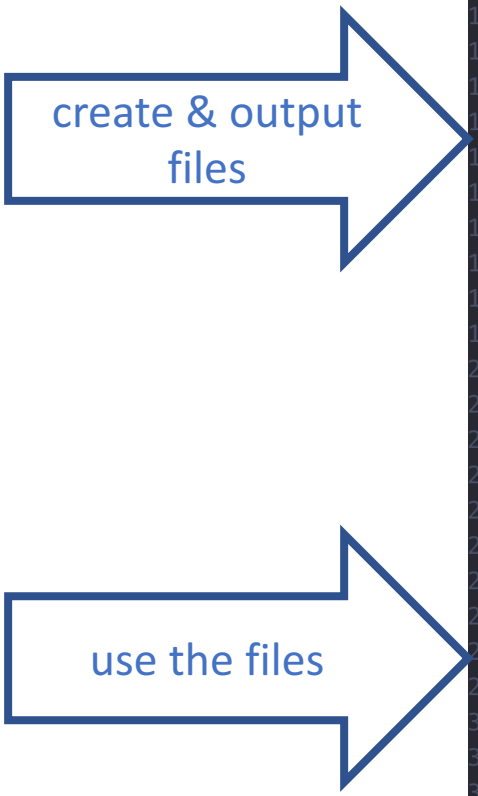
```
[2018-04-04 16:28:32]
kellys04@acc38pathlabmac01:~/projects/nextflow-demos/print-samples2$ ls -la work/3e/b1a9f9d042e6c5fa74dcf2e57d593b/
total 40
drwxr-xr-x  9 kellys04  NYUMC\Domain Users   306 Apr  4 16:15 .
drwxr-xr-x  3 kellys04  NYUMC\Domain Users   102 Apr  4 16:15 ..
-rw-r--r--  1 kellys04  NYUMC\Domain Users     0 Apr  4 16:15 .command.begin
-rw-r--r--  1 kellys04  NYUMC\Domain Users     0 Apr  4 16:15 .command.err
-rw-r--r--  1 kellys04  NYUMC\Domain Users    23 Apr  4 16:15 .command.log
-rw-r--r--  1 kellys04  NYUMC\Domain Users    23 Apr  4 16:15 .command.out
-rw-r--r--  1 kellys04  NYUMC\Domain Users  1886 Apr  4 16:15 .command.run
-rw-r--r--  1 kellys04  NYUMC\Domain Users    46 Apr  4 16:15 .command.sh
-rw-r--r--  1 kellys04  NYUMC\Domain Users     1 Apr  4 16:15 .exitcode
```

- `.command.sh`: process ‘script’ contents
- `.command.run`: Executes `.command.sh`, managing environment, staging, logging, etc.
- `.command.log`: process log file (stdout & stderr)
- Check these files for troubleshooting!

Managing file input & output

- **Don't!**

- Nextflow handles this automatically, you only need to manage process 'input' and 'output' in your pipeline



main.nf

```nf
Channel.from( ['Sample1','Sample2','Sample3','Sample4'] ).set { samples }

params.output_dir = "output"

process make_file {
    tag { "${sampleID}" }
    publishDir "${params.output_dir}/make_file", mode: 'copy', overwrite: true

    input:
    val(sampleID) from samples

    output:
    file("${sampleID}.txt") into (sample_files, sample_files2)

    script:
    """
    echo "[make_file] ${sampleID}" > "${sampleID}.txt"
    """
}
sample_files2.collectFile(name: 'sample_files.txt', storeDir: "${params.output_dir}")

process gather_files {
    publishDir "${params.output_dir}/gather_files", mode: 'copy', overwrite: true

    input:
    file("*") from sample_files.collect()

    output:
    file("output.txt")

    script:
    """
    cat * > output.txt
    """
}

```

‘work’ dir – where processing takes place

```
[2018-04-04 16:42:02]
kellys04@acc38pathlabmac01:~/projects/nextflow-demos/output-files$ ls -la work/86/f83fb4e2ee0fb5b97a796ecd368ee3/
total 64
drwxr-xr-x  14 kellys04  NYUMC\Domain Users   476 Apr  4 16:42 .
drwxr-xr-x   3 kellys04  NYUMC\Domain Users   102 Apr  4 16:42 ..
-rw-r--r--   1 kellys04  NYUMC\Domain Users     0 Apr  4 16:42 .command.begin
-rw-r--r--   1 kellys04  NYUMC\Domain Users     0 Apr  4 16:42 .command.err
-rw-r--r--   1 kellys04  NYUMC\Domain Users     0 Apr  4 16:42 .command.log
-rw-r--r--   1 kellys04  NYUMC\Domain Users     0 Apr  4 16:42 .command.out
-rw-r--r--   1 kellys04  NYUMC\Domain Users  2456 Apr  4 16:42 .command.run
-rw-r--r--   1 kellys04  NYUMC\Domain Users    35 Apr  4 16:42 .command.sh
-rw-r--r--   1 kellys04  NYUMC\Domain Users     1 Apr  4 16:42 .exitcode
lrwxr-xr-x   1 kellys04  NYUMC\Domain Users   103 Apr  4 16:42 Sample1.txt -> /Users/kellys04/projects/nextflow-demos/output-fi
les/work/a3/b834e4385c05b9585ba0f45fee2b65/Sample1.txt
lrwxr-xr-x   1 kellys04  NYUMC\Domain Users   103 Apr  4 16:42 Sample2.txt -> /Users/kellys04/projects/nextflow-demos/output-fi
les/work/71/d09165bbcbd33f53ca98f3201b51a8/Sample2.txt
lrwxr-xr-x   1 kellys04  NYUMC\Domain Users   103 Apr  4 16:42 Sample3.txt -> /Users/kellys04/projects/nextflow-demos/output-fi
les/work/70/61007db7fb0c19ebd3dcb9702d8656/Sample3.txt
lrwxr-xr-x   1 kellys04  NYUMC\Domain Users   103 Apr  4 16:42 Sample4.txt -> /Users/kellys04/projects/nextflow-demos/output-fi
les/work/38/ee02dc3fb3ab60ad149d7453620f95/Sample4.txt
-rw-r--r--   1 kellys04  NYUMC\Domain Users    80 Apr  4 16:42 output.txt
```

- input files symlinked
- output files produced

‘publishDir’ – optional output for your consumption (not the pipeline’s)

```
[2018-04-04 16:48:03]
kellys04@acc38pathlabmac01:~/projects/nextflow-demos/output-files$ tree output/
output/
|-- gather_files
|   `-- output.txt
|-- make_file
|   |-- Sample1.txt
|   |-- Sample2.txt
|   |-- Sample3.txt
|   `-- Sample4.txt
`-- sample_files.txt
```

‘collectFile’ – convenient data gathering

```
[2018-04-04 16:48:06]
kellys04@acc38pathlabmac01:~/projects/nextflow-demos/output-files$ cat output/sample_files.txt
[make_file] Sample4
[make_file] Sample2
[make_file] Sample1
[make_file] Sample3
```

Asynchronous execution

main.nf

```
Channel.from( ['Sample1','Sample2','Sample3'] ).into { samples; samples2 }

params.output_dir = "output"

process print_sample {
    tag { "${sampleID}" }
    echo true

    input:
    val(sampleID) from samples

    script:
    """
    echo "[print_sample] sample is: ${sampleID}"
    """
}

process make_file {
    tag { "${sampleID}" }
    echo true
    publishDir "${params.output_dir}/make_file", mode: 'copy', overwrite: true

    input:
    val(sampleID) from samples2

    output:
    file("${sampleID}.txt") into (sample_files, sample_files2)

    script:
    """
    echo "[make_file] ${sampleID}"
    echo "[make_file] ${sampleID}" > "${sampleID}.txt"
    """
}
sample_files2.collectFile(name: 'sample_files.txt', storeDir: "${params.output_dir}")

process gather_files {
    echo true
    publishDir "${params.output_dir}/gather_files", mode: 'copy', overwrite: true

    input:
    file("*") from sample_files.collect()

    output:
    file("output.txt")

    script:
    """
    echo "[gather_files] gathering all files..."
    cat * > output.txt
    """
}
```

```
[2018-04-04 16:58:52]
kellys04@acc38pathlabmac01:~/projects/nextflow-demos/async$ ./nextflow run main.nf
N E X T F L O W  ~  version 0.28.0
Launching `main.nf` [focused_varahamihira] - revision: da96d1de03
[warm up] executor > local
[01/5f5316] Submitted process > print_sample (Sample2)
[70/c48145] Submitted process > print_sample (Sample3)
[3c/88c8f5] Submitted process > make_file (Sample1)
[d0/0a5ec6] Submitted process > print_sample (Sample1)
[d4/f075df] Submitted process > make_file (Sample3)
[fe/67bd73] Submitted process > make_file (Sample2)
[print_sample] sample is: Sample2
[make_file] Sample1
[print_sample] sample is: Sample1
[make_file] Sample3
[make_file] Sample2
[84/e18905] Submitted process > gather_files
[print_sample] sample is: Sample3
[gather_files] gathering all files...
```

- `make_file` and `print_sample` execute independently, in parallel
- `gather_files` does not execute until all `make_file` are finished

‘profiles’ – configuration sets

nextflow.config | main.nf | Makefile

```
Channel.fromPath( 'input/fastq/*.fastq.gz' ).set { input_fastqs }
params.output_dir = "output"

process fastqc {
    tag { "${fastq}" }
    publishDir "${params.output_dir}/fastqc", mode: 'copy', overwrite: true
    echo true

    input:
    file(fastq) from input_fastqs

    output:
    file(output_html)
    file(output_zip)

    script:
    output_html = "${fastq}".replaceFirst(/.fastq.gz$/, "_fastqc.html")
    output_zip = "${fastq}".replaceFirst(/.fastq.gz$/, "_fastqc.zip")
    """
    which fastqc
    fastqc -o . "${fastq}"
    """
}

```

nextflow.config | Makefile | README.md

```
manifest {
    author = 'Stephen Kelly'
    homePage = 'https://github.com/stevekm/nextflow-demos'
    description = 'Nextflow programming demos'
    mainScript = 'main.nf'
}

report {
    enabled = true
    file = "nextflow-report.html"
}

trace {
    enabled = true
    fields =
    "task_id,hash,native_id,process,tag,name,status,exit,module,containe
    realtime,queue,%cpu,%mem,rss,vmem,peak_rss,peak_vmem,rchar,wchar,sys
    file = "trace.txt"
    raw = true
}

timeline {
    enabled = true
    file = "timeline-report.html"
}

executor.queueSize = 3

profiles {
    docker {
        docker.enabled = true
        process.$fastqc.container = "stevekm/ngs580-nf:fastqc-0.11.7"
    }
    phoenix {
        process.$fastqc.module = "fastqc/0.11.7"
        process.executor = "sge"
    }
}

```

run locally with Docker

```
[2018-04-04 17:46:14]
kellys04@acc38pathlabmac01:~/projects/nextflow-demos/profiles$ make run-docker
docker --version > /dev/null 2>&1 || { echo "ERROR: 'docker' not found" && exit 1 ; }
./nextflow run main.nf -with-dag flowchart.dot -profile docker
N E X T F L O W  ~  version 0.28.0
Launching `main.nf` [trusting_baekeland] - revision: c238bb3a0a
[warm up] executor > local
[c6/aa3530] Submitted process > fastqc (SeraCare-1to1-Positive_S2_L001_R1_001.fastq.gz)
[47/a6737c] Submitted process > fastqc (SeraCare-1to1-Positive_S2_L001_R2_001.fastq.gz)
[59/c0b1ce] Submitted process > fastqc (HapMap-B17-1267_S8_L001_R1_001.fastq.gz)
[1b/de9dfc] Submitted process > fastqc (HapMap-B17-1267_S8_L001_R2_001.fastq.gz)
/opt/FastQC/fastqc
Analysis complete for SeraCare-1to1-Positive_S2_L001_R2_001.fastq.gz
/opt/FastQC/fastqc
Analysis complete for SeraCare-1to1-Positive_S2_L001_R1_001.fastq.gz
/opt/FastQC/fastqc
Analysis complete for HapMap-B17-1267_S8_L001_R1_001.fastq.gz
/opt/FastQC/fastqc
Analysis complete for HapMap-B17-1267_S8_L001_R2_001.fastq.gz
```

```
[2018-04-04 17:56:13]
kellys04@phoenix2:~/projects/nextflow-demos/profiles$ make run-phoenix
module > /dev/null 2>&1 || { echo "ERROR: 'module' not found" && exit 1 ; }
./nextflow run main.nf -with-dag flowchart.dot -profile phoenix
N E X T F L O W  ~  version 0.28.0
Launching `main.nf` [distraught_becquerel] - revision: c238bb3a0a
[warm up] executor > sge
[84/2b9506] Submitted process > fastqc (HapMap-B17-1267_S8_L001_R1_001.fastq.gz)
[2e/7c62aa] Submitted process > fastqc (SeraCare-1to1-Positive_S2_L001_R2_001.fastq.gz)
[af/7d3df6] Submitted process > fastqc (HapMap-B17-1267_S8_L001_R2_001.fastq.gz)
/local/apps/fastqc/0.11.7/fastqc
Analysis complete for HapMap-B17-1267_S8_L001_R1_001.fastq.gz
[87/d2ecb8] Submitted process > fastqc (SeraCare-1to1-Positive_S2_L001_R1_001.fastq.gz)
/local/apps/fastqc/0.11.7/fastqc
Analysis complete for SeraCare-1to1-Positive_S2_L001_R2_001.fastq.gz
/local/apps/fastqc/0.11.7/fastqc
Analysis complete for HapMap-B17-1267_S8_L001_R2_001.fastq.gz
/local/apps/fastqc/0.11.7/fastqc
Analysis complete for SeraCare-1to1-Positive_S2_L001_R1_001.fastq.gz
```

run on HPC with ‘module’

Using custom scripts

- if present, '`bin`' directory is prepended to `$PATH` during process `script` execution

```
[2018-04-04 18:44:00]
kellys04@phoenix2:~/projects/nextflow-demos/R-Python$ ll
total 89K
drwxr-s---  3 kellys04 kellys04   72 Apr  4 17:36 .
drwxr-s--- 14 kellys04 kellys04  406 Apr  4 17:36 ..
-rw-r-----  1 kellys04 kellys04  251 Apr  4 17:36 Makefile
drwxr-s---  2 kellys04 kellys04  100 Apr  4 18:43 bin
-rw-r-----  1 kellys04 kellys04 1.7K Apr  4 17:36 main.nf

[2018-04-04 18:44:00]
kellys04@phoenix2:~/projects/nextflow-demos/R-Python$ ll bin/
total 106K
drwxr-s--- 2 kellys04 kellys04 100 Apr  4 18:43 .
drwxr-s--- 3 kellys04 kellys04  72 Apr  4 17:36 ..
-rwxr-x--- 1 kellys04 kellys04 385 Apr  4 17:36 test.R
-rwxr-x--- 1 kellys04 kellys04  81 Apr  4 17:36 test.py
-rwxr-x--- 1 kellys04 kellys04  79 Apr  4 17:36 tools.R
-rwxr-x--- 1 kellys04 kellys04  68 Apr  4 17:36 tools.py
```

More Basic Demos

- https://github.com/stevekm/nextflow-demos

Pipelines

- Clinical exome sequencing: https://github.com/NYU-Molecular-Pathology/NGS580-nf
- demultiplexing: https://github.com/NYU-Molecular-Pathology/demux-nf

Nextflow Challenges

- very different structure & syntax from other frameworks
- consideration needed for ‘data flow’ paradigm & asynchronicity
- modular pipeline components not implemented yet
  - unable to store ‘processes’ in separate files for dynamic importation; potential future feature
- Combining, joining, merging of data channels can get tricky for complex cases
  - matching sample-pairs with per-sample metadata

Nextflow Advantages

- extremely portable & lightweight
- makes pipelines easy to run, maintain, troubleshoot
- robust process isolation, input/output file management
  - failed compute jobs don't affect pipeline re-runs
- **greatly reduces overhead of environment management & task execution on HPC cluster**
  - **greatly reduces pipeline code complexity and debugging!!**

Conclusions

- Nextflow handles 95% of common pipeline use-cases easily
  - remaining cases typically require usage of advanced features or re-thinking to better fit data-flow paradigm
- Ease & portability of execution offered by Nextflow will be critical factors for ABL in the near future (HPC upgrades & migration)
  - pipelines not tied to a single compute system e.g. phoenix
- Standardized programming interface rectifies inconsistent code base
- **The advantages offered by Nextflow outweigh its challenges**

Notes

- very active developer support
  - primary developer Paolo responds quickly to posts on Google Groups, GitHub
  - Google Cloud Platform for genomics integration expected end of 2018
- Pairs well with Docker/Singularity for dependency management, Makefiles for config management & execution shortcuts
- lots of helpful extra features
  - HTML pipeline reports, email output & notifications
- Knowledge of Groovy & Java not required but it helps

Resources

- Nextflow Homepage: https://www.nextflow.io/
- Nextflow Docs: https://www.nextflow.io/docs/latest/getstarted.html
- Nextflow GitHub: https://github.com/nextflow-io/nextflow
- Nextflow Google Group: https://groups.google.com/forum/#!forum/nextflow
- Groovy docs
  - http://groovy-lang.org/documentation.html
  - http://docs.groovy-lang.org/latest/html/documentation/

Examples

- Nextflow tutorial: https://github.com/nextflow-io/hack17-tutorial
- Nextflow examples: https://github.com/nextflow-io/examples
- Pipeline examples: https://github.com/nextflow-io/awesome-nextflow
- Boilerplate example for writing new pipelines: https://github.com/stevekm/nextflow-boilerplate