PORTABLE CONTAINERS ORCHESTRATION AT SCALE WITH NEXTFLOW

Paolo Di Tommaso, CRG

HPC Knowledge Meeting, 21 June 2018, BCN
WHO IS THIS CHAP?

@PaoloDiTommaso
Research software engineer
Comparative Bioinformatics, Notredame Lab
Center for Genomic Regulation (CRG)
Author of Nextflow project
GENOMIC WORKFLOWS

• Data analysis applications to extract information from (large) genomic datasets
• Embarrassingly parallelisation, can spawn 100s-100k jobs over distributed cluster
• Mash-up of many different tools and scripts
• Complex dependency trees and configuration → very fragile ecosystem
Steinbiss et al., *Companion parasite genome annotation pipeline*, DOI: 10.1093/nar/gkw292
To reproduce the result of a typical computational biology paper requires 280 hours.

≈ 1.7 months!
CHALLENGES

• Reproducibility, replicate results over time
• Portability, run across different platforms
• Scalability ie. deploy big distributed workloads
• Usability, streamline execution and deployment of complex workloads ie. remove complexity instead of adding new one
• Consistency ie. track changes and revisions consistently for code, config files and binary dependencies
HOW ??
PACKAGE MANAGERS

- Module Environments, EasyBuild, Conda, Spack, etc.
- They provide a big help, but still a lot of problems:
  - The user still need to configure the target system
  - May not track low level dependencies eg. glibc, etc.
  - May not provide all required packages or libraries
  - May not be portable across platforms
CONTAINERS!

• Allows self-contained, pre-configured execution environments
• Very low runtime overhead
• Transparent, replicable and sharable
• Preserve against software decay
IS THIS THE SOLUTION?
SERVICES ORCHESTRATION ≠ TASKS SCHEDULING
PROBLEMS

• Containers were mainly designed for (micro)services
• How to scale large containerised workload into cluster?
• How handle task input/output files mounts?
• How manage errors and automatic fail over?
• How to deploy the same pipeline across different infra, clustering technology and container runtimes?
Orchestration & Parallelisation

Scalability & Portability

Deployment & Reproducibility

nextflow

containers

Git

GitHub

GitLab

Bitbucket
CONTAINERISATION

• Nextflow envisioned the use of software containers to fix computational reproducibility

• Each task is executed in its own container instance

• Container executions are defined in a declarative manner i.e. w/o modifying application code

• Support for multiple container engines
bwa mem reference.fa sample.fq
  | samtools sort -o sample.bam
process align_sample {

input:
file 'reference.fa' from genome_ch
file 'sample.fq' from reads_ch

output:
file 'sample.bam' into bam_ch

script:
""
    bwa mem reference.fa sample.fq \
        | samtools sort -o sample.bam
""

}
### TASKS COMPOSITION

**process align_sample**

```plaintext
input:
file 'reference.fa' from genome_ch
file 'sample.fq' from reads_ch

output:
file 'sample.bam' into bam_ch

script:
""
  bwa mem reference.fa sample.fq \\
  | samtools sort -o sample.bam
""
```

**process index_sample**

```plaintext
input:
file 'sample.bam' from bam_ch

output:
file 'sample.bai' into bai_ch

script:
""
  samtools index sample.bam
""
```
HOW PARALLELISATION WORKS

samples_ch = Channel.fromPath('data/sample.fastq')

process FASTQC {

    input:
    file reads from samples_ch

    output:
    file 'fastqc_logs' into fastqc_ch

    """
    mkdir fastqc_logs
    fastqc -o fastqc_logs -f fastq -q ${reads}
    """
}

HOW PARALLELISATION WORKS

```
samples_ch = Channel.fromPath('data/*.fastq')

process FASTQC {
    input:
        file reads from samples_ch
    output:
        file 'fastqc_logs' into fastqc_ch

    """
    mkdir fastqc_logs
    fastqc -o fastqc_logs -f fastq -q ${reads}
    """
}
```
IMPLICIT PARALLELISM

```
Channel.fromPath("data/*.fastq")
```
DEPLOYMENT SCENARIOS
LOCAL EXECUTION

- Typical scenario for development
- Dependencies can be managed using a container runtime
- Parallelisations is managed spawning posix processes
- Can scale vertically using fat server / shared mem. machine

Diagram:

```
+----------------+          +----------------+
| laptop/workstation |          | nextflow |
|                  +----------------+          +----------------+
|                  |  ↓                   |          |  ↓                   |
|  ↓               |          +----------------+          |          +----------------+
| docker/singularity |          | OS |          | local storage |
```

- nextflow
- docker/singularity
- OS
- local storage
CENTRALISED ORCHESTRATION

- Nextflow orchestrates workflow execution by submitting jobs to a compute cluster, e.g., SLURM.
- It can run in the head node or a compute node.
- Requires a shared storage to exchange data between tasks.
- Ideal for coarse-grained parallelisms.

Diagram:
- CENTRALISED ORCHESTRATION
  - Nextflow
  - Computer cluster
  - Submit jobs
  - Cluster nodes
  - NFS/Lustre
  - Shared storage for data exchange
A single job request allocates the desired compute nodes.

Nextflow deploys its own embedded compute cluster.

The main instance orchestrates the workflow execution.

The worker instances execute workflow jobs (work stealing approach).
PORTABILITY

![Diagram showing "nextflow" as a bridge between two different systems.](image-url)
PORTABILITY

process {
    executor = 'slurm'
    queue = 'my-queue'
    memory = '8 GB'
    cpus = 4
    container = 'user/image'
}
PORTABILITY

```python
process {
    executor = 'awsbatch'
    queue = 'my-queue'
    memory = '8 GB'
    cpus = 4
    container = 'user/image'
}
```
ERROR RECOVERY

• Each task outputs are saved in a separate directory
• This allows to safely record interrupted executions discarding
• Dramatically simplify debugging!
• Automatic error healing, restart the task with increased resources ie. memory, timeout, etc.
NEXTFLOW USAGE

• In production since 2014
• ~ 25 production pipelines
• ~ 33k CPU-hours and 270k jobs per month
• 1.4 mln CPU-hours and 12 mln jobs in last 4 years
• Most of this using Singularity containers
WHO IS USING NEXTFLOW?

Logos of various institutions and organizations.
Community effort to collect production ready genomic analysis pipelines built with Nextflow

Initially supported by SciLifeLab, QBiC and A*Star Genome Institute Singapore

http://nf-co.re
A QUICK COMPARISON WITH OTHER TOOLS
GALAXY vs. NEXTFLOW

- Web based platform
- Built-in integration with many tools and dataset
- Little control over tasks parallelisation
- Scalability 10→1K jobs
- Complex installation and maintenance
- Suited for training + not experienced bioinformaticians

- Command line oriented tool
- Can incorporate any tool w/o any extra adapter
- Fine control over tasks parallelisation
- Scalability 100→1M jobs
- One liner installer
- Suited for production workflows + experienced bioinformaticians
SNAKEMAKE vs. NEXTFLOW

- Command line oriented tool
- Pull model
- Rules defined using file name patterns
- Compute DAG ahead
- Built-in support for Singularity
- Custom scripts for cluster deployments
- Support for sub-workflows
- No support for source code management system
- Python based

- Command line oriented tool
- Push model
- Can manage any data structure
- Compute DAG at runtime
- All major container runtimes
- Built-in support for clusters and cloud
- Support for sub-workflows
- No (yet) support for sub-workflows
- Built-in support for Git/GitHub, etc., manage pipeline revisions
- Groovy/JVM based
CWL vs. NEXTFLOW

- Language specification
- Declarative meta-language (YAML/JSON)
- Verbose
- Committee driven
- Many vendors/implementations (and specification version)
- Language + app. runtime
- DSL on top of a general purpose programming lang.
- Concise, fluent (at least try to be!)
- Community driven
- Single implementation, quick iterations
Nextflow workflow report
[trusting_cuvier] (resumed run)

Workflow execution completed successfully!

Run times
Fri Apr 27 23:19:53 CEST 2018 - Sat Apr 28 03:18:15 CEST 2018 (completed a day ago, duration: 3h 58m 21s)

5329 succeeded 2849 cached

Nextflow command
nextflow run main.nf -profile crg --std_align=true --default_align=true --align_method=CLUSTAL0,MAFFT --
tree_method=CLUSTAL0,MAFFT_PARTHREE --seqs=/users/cn/egarriga/datasets/homfamClustalo/seqs/*,fa --
refs=/users/cn/egarriga/datasets/homfamClustalo/refs/*,ref --with-report --with-trace --resume --bg

CPU-Hours 15.6 (31.5% cached, 4.6% failed)
Launch directory /nfs/users2/cn/egarriga/projects/dpa_cp
Work directory /nfs/users2/cn/egarriga/projects/dpa_cp/work
Project directory /nfs/users2/cn/egarriga/projects/dpa_cp
Script name main.nf
Script ID 6ff767a42e50448d41927a6e5a9787fc
Workflow session 087e9bc8-e488-4311-88aa-961138c42fd6
Workflow profile crg
Workflow container cncrg/regressive-msa:v0.2.4
Container engine singularity
Nextflow version version 0.28.2, build 4782 (06-04-2018 12:25 UTC)
EXECUTION REPORT

Resource Usage
These plots give an overview of the distribution of resource usage for each process.

CPU Usage
- Raw Usage
- % Allocated

CPU Usage Chart:
- % single core CPU usage
- Bars represent different processes:
  - combine_seqs
  - evaluate
  - std_alignment
  - dpa_alignment
  - guide_trees
  - default_alignment
### Tasks

This table shows information about each task in the workflow. Use the search box on the right to filter rows for specific values. Clicking headers will sort the table by that value and scrolling side to side will reveal more columns.

<table>
<thead>
<tr>
<th>task_id</th>
<th>process</th>
<th>tag</th>
<th>status</th>
<th>hash</th>
<th>allocated cpus</th>
<th>%cpu</th>
<th>allocated memory (bytes)</th>
<th>%mem</th>
<th>vmem</th>
<th>rss</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>index</td>
<td>Homo_sapiens.GRCh38.cdna.all.fa1</td>
<td>COMPLETED</td>
<td>f4/a72585</td>
<td>2</td>
<td>195.0</td>
<td>8589934592</td>
<td>31.9</td>
<td>5272805376</td>
<td>51318</td>
</tr>
<tr>
<td>2</td>
<td>parseEncode</td>
<td>/home/pditommaso/projects/rnaseq/encode-rnf/data/metadata.tsv</td>
<td>COMPLETED</td>
<td>12/bd1fd3</td>
<td>1</td>
<td>0.0</td>
<td>-</td>
<td>0.0</td>
<td>17960960</td>
<td>53241</td>
</tr>
<tr>
<td>3</td>
<td>fastqc</td>
<td>FASTQC on SRR5210435</td>
<td>COMPLETED</td>
<td>ba/5068a0</td>
<td>2</td>
<td>46.4</td>
<td>6442450944</td>
<td>0.0</td>
<td>4088819712</td>
<td>3685</td>
</tr>
<tr>
<td>4</td>
<td>fastqc</td>
<td>FASTQC on SRR3192620</td>
<td>COMPLETED</td>
<td>fa/3efdb3</td>
<td>2</td>
<td>76.7</td>
<td>6442450944</td>
<td>0.0</td>
<td>4089171968</td>
<td>50494</td>
</tr>
<tr>
<td>5</td>
<td>fastqc</td>
<td>FASTQC on SRR3192621</td>
<td>FAILED</td>
<td>6b/1f73ec2</td>
<td>2</td>
<td>-</td>
<td>6442450944</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>fastqc</td>
<td>FASTQC on SRR3192434</td>
<td>COMPLETED</td>
<td>1e/df73c2</td>
<td>2</td>
<td>68.8</td>
<td>6442450944</td>
<td>0.0</td>
<td>4088832000</td>
<td>41535</td>
</tr>
<tr>
<td>7</td>
<td>fastqc</td>
<td>FASTQC on SRR3192433</td>
<td>COMPLETED</td>
<td>5e/4886ef</td>
<td>2</td>
<td>70.2</td>
<td>6442450944</td>
<td>0.0</td>
<td>4031012864</td>
<td>3843</td>
</tr>
</tbody>
</table>
Processes execution timeline

Launch time: 15 Jun 2016 15:03
Elapsed time: 49m 9s

downloadReference (1) 5.7s / 549.2 GB
downloadSRA (1) 59s / 1.2 GB
decompressReference (1) 1.5s /
indexReference (1) 5.7s / 80.9 MB
extractSRA (1) 13s / 150.5 MB
trim (1) 36.1s / 8.1 GB
mergeTrimEnds (1) 40.2s / 31.6 MB
filterKMC (1) 2m 14s / 8.6 GB
filterKMER (1) 38m 31s / 8.9 GB
alignReads_kmc (1) 10m 17s / 2 GB
sortAlignment_kmc (1) 1m 52s / 538.9 MB
indexAlignment_kmc (1) 8.8s / 36.2 MB
callVariants_kmc (1) 20m 16s / 99.2 MB
alignReads_khmer (1) 1m 23s / 1.8 GB
sortAlignment_khmer (1) 22s / 524.4 MB
indexAlignment_khmer (1) 2.2s / 36.2 MB
callVariants_khmer (1) 3m 21s / 89.1 MB

Created with Nextflow -- http://nextflow.io
DAG VISUALISATION
EDITORS!
WHAT'S NEXT
IMPROVEMENTS

• Better meta-data and execution provenance
• Batch jobs ie. aggregate many tasks to a single job
• Workflow composition aka sub-workflows
• More clouds support ie. Azure and GCP
APACHE SPARK

• Native support for Apache Spark clusters and execution model

• Allow hybrid Nextflow and Spark applications

• Mix the best of the two worlds, Nextflow for legacy tools/course grain parallelisation and Spark for fine grain/distributed execution eg. GATK4
CONCLUSION

• Data analysis reproducibility is hard and it's often underestimated.

• Nextflow does not provide a magic solution but enables best-practices and provide support for community and industry standards.

• It strictly separates the application logic from the configuration and deployment logic, enabling self-contained workflows.

• It allows the deployment of containerised workloads at scale, targeting different infras and container runtimes.
ACKNOWLEDGMENT

Notredame Lab, CRG

Evan Floden
Emilio Palumbo
Cedric Notredame

http://nextflow.io
Courses

Reproducible
In silico Genomics

22nd - 23rd November 2018

Organizers
Cédric Notredame (CRG, Spain)
Evan Wade Floden (CRG, Spain)
Pablo Di Tommaso (CRG, Spain)

Location
Centre for Genomic Regulation (CRG)
Barcelona, Spain

Registration deadline
5th September 2018

More information at:

Organized by:

Members of the CRG Board of Trustees:

Supported by: