Running Portable Workflow and Container Specifications at Production Scale in the Cloud

Strategy and Best Practices
DNAnexus

Source code: https://github.com/dnanexus/[dx-toolkit,dx-cwl,dxWDL]
License: Apache License 2.0
What I hope you all get out of this talk

- Three best practices for writing portable workflows so they are reusable and scale in cloud
- Useful properties of an execution environment as seen by our customers
- Potential steps forward for our community based on our experience
At Scale?

Multiple Organizations Each Processing:

Tens of Thousands of Genomes
Hundreds of Thousands of Exomes
Portable Workflow Representations

SOLVED!
Portable Workflow Executions
<table>
<thead>
<tr>
<th><strong>Local</strong></th>
<th><strong>Cluster</strong></th>
<th><strong>Cloud</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Expectations:</strong></td>
<td><strong>Expectations:</strong></td>
<td><strong>Expectations:</strong></td>
</tr>
<tr>
<td>● Quickly access CPU + files</td>
<td>● Quick-ish to access CPU + files</td>
<td>● Collaboration across regions</td>
</tr>
<tr>
<td>● Dev a workflow on an airplane</td>
<td>● Realistic sample sizes</td>
<td>● 'Limitless' scalability</td>
</tr>
<tr>
<td>● Run small tests</td>
<td>● 'Packing problem' to run jobs</td>
<td>● Enduring provenance</td>
</tr>
<tr>
<td>● Use IDE</td>
<td>● UNIXy like orchestration</td>
<td>● Realistic sample sizes</td>
</tr>
<tr>
<td><strong>Limitations:</strong></td>
<td><strong>Limitations:</strong></td>
<td><strong>Limitations:</strong></td>
</tr>
<tr>
<td>● Limited sample sizes</td>
<td>● Fixed-ish resources</td>
<td>● CPU + file request costs*</td>
</tr>
<tr>
<td>● Unrealistic test situations</td>
<td>● NFS access causes bottlenecks</td>
<td>● Network bandwidth costs</td>
</tr>
<tr>
<td>● Configure and setup cost</td>
<td>● Configure and setup cost</td>
<td>● Adapt to cloud architecture</td>
</tr>
<tr>
<td>● Limited provenance</td>
<td>● Limited provenance</td>
<td></td>
</tr>
</tbody>
</table>

* Getting better with time!
Three Best Practices for Local, Cluster, and Cloud
6 CPU Requests
7 Storage Requests

preproc
  ↓
grep
  ↓
cut
  ↓
sort
  ↓
uniq
  ↓
postproc

3 CPU Requests
4 Storage Requests

preproc
  ↓
grep
cut
  ↓
sort
  ↓
uniq
  ↓
postproc
Three Best Practices for Local, Cluster, and Cloud

1. Minimize Wall Clock Time, Maximize Reusability/Granularity
-F
patterns.txt

preproc

-grep

cut

sort

uniq

postproc

-nproc 5
(default)

More modular/reusable
Easier to understand WF
Explicitly handle defaults
Three Best Practices for Local, Cluster, and Cloud

1. Minimize Wall Clock Time, Maximize Reusability/Granularity
2. Workflow I/O > Sum of Task I/O: Differentiate Workflow from its Steps
Features of CWL, WDL, and Docker*

**Basics:**
- Filesystem encapsulation
- Task/Tool definitions
- Dependencies between tasks/tools (workflow definitions)
- Conditional execution

**Potentially 'handy' features of CWL, WDL, Docker:**
- Workflow-level processing/tasks
- Can execute arbitrary code in some settings

* Some may still be in dev
Two models for executing CWL/WDL workflows

preproc
grep
cut
sort
uniq
postproc
preproc
grep
cut
sort
uniq
postproc
preproc
Three Best Practices for Local, Cluster, and Cloud

1. Minimize Wall Clock Time, Maximize Reusability/Granularity
2. Sum Workflow I/O > Sum of Task I/O: Differentiate Workflow from its Steps
3. **Treat Workflow-Level Processing as Tasks**
Desirable features of execution environment

- Full Provenance Tracking
- Automatic Restarts & Reusability
- Versioning, Publishing, and Collaboration
- Cached Assets
- User-space Docker Support
Some thoughts going forward

- More DREAM Challenges => Improvements in CWL/WDL

- Portable Workspaces [in Git]
  - Git-like file hashes
  - Pointers to actual files
  - Includes provenance info
Thanks Portable Enthusiasts!

@DNAnexus:
- Ohad Rodeh
- Mike Lin
- Aleksandra Zalcman
- Kurt Jensen

External:
- Brad Chapman
- Michael Crusoe
- Jeff Gentry
- Brian O'Connor
- Seth Strattan

- Nihar Sheth
- John Ellithorpe
- George Asimenos
FROM pancancer/seqware_whitestar_pancancer:1.1.2
ENV SANGER_VERSION 1.0.8
LABEL SANGER_VERSION $SANGER_VERSION
USER root

### START of CGP INSTALL ###

ENV OPT /opt/wtsi-cgp
ENV PATH $OPT/bin:$PATH
ENV PERL5LIB $OPT/lib/perl5:$PERL5LIB

RUN apt-get -yqq update && \
    apt-get -yqq install libreadline6-dev build-essential \ 
    wget time curl zlib1g-dev libncurses5-dev \ 
    libgd2-xpm-dev libexpat1-dev python unzip libboost \ 
    libstreams-dev libglib2.0-dev gfortran libcairo \ 
    openjdk-7-jdk libtest-most-perl && \
    apt-get clean

RUN mkdir -p /tmp/downloads $OPT/bin $OPT/ etc $OPT/lib

WORKDIR /tmp/downloads

RUN cpanm --mirror https://cpan.metacpan.org -l $OPT / \
    rm -rf ~/.cpanm

RUN export SOURCE_JKENT_BIN=https://github.com/ENCODE- \
    curl -sSL o $OPT/bin/wigToBigWig -C --retry 10 \ 
    curl -sSL o $OPT/bin/bigWigMerge -C --retry 10

cwlVersion: v1.0

inputs:
    tumor:
        type: File
        inputBinding:
            position: 1
            prefix: --tumor
        secondaryFiles:
            - .baiFiles:
    refFrom:
        type: File
        inputBinding:
            position: 3
            prefix: --refFrom
    bbFrom:
        type: File
        inputBinding:
            position: 4
            prefix: --bbFrom

normal:
    type: File
    inputBinding:
        position: 2
        prefix: --normal
    secondaryFiles:
        - .bai

outputs:
    somatic_sv_tar.gz:
        type: File

    somatic_snv_mnv_tar.gz:
        type: File

    somatic_indel.tar.gz:
        type: File

    somatic_imputeCounts.tar.gz:
        type: File

    somatic_genotype.tar.gz:
        type: File

    somatic_verifyBamId.tar.gz:
        type: File

command {
    python /home/ seqware/CGP-Somatic- Dockstore \
        --tumor ${tumorBam} \ 
        --normal ${normalBam} \ 
        --refFrom ${refFrom} \ 
        --bbFrom ${bbFrom} \ 
        --output-dir ${outputDir}
}

output {
    Array[File] somatic_snv_mnv_tar.gz = glob(

    Array[File] somatic_cv_tar.gz = glob(

    Array[File] somatic_indel_tar.gz = glob(

    Array[File] somatic_imputeCounts_tar.gz = glob(

    Array[File] somatic_genotype_tar.gz = glob(

    Array[File] somatic_verifyBamId_tar.gz = glob(

    String outputDir = "."
Keeping Docker and CWL/WDL in sync

Problem:

Docker images may get out of sync with workflow if consistent tags aren't used.

Potential solution:

1. Dockerfiles, CWL/WDL files for a workflow all go in the same repo
2. Repo maintains tags for versions
3. All CWL/WDL references to the Docker images refer to a specific version
4. Docker images pushed to registries are tagged with the version
5. Workflow is built and tagged with the version (if supported by executor)

Ideally: these steps are automated via something like Travis CI
Four Best Practices for Local, Cluster, and Cloud

1. Locality Principle: Minimize Data Transfer, Maximize Comp Efficiency
2. Minimize, and Define Well the # of Inputs and Outputs
3. Stick to the Core Features of CWL, WDL, and Docker
4. **Stick to a Well Defined and Versioned Development Pipeline**