Comparative Analysis of Genomic Sequencing Workflow Management Systems

Cynthia Liu, Azza Ahmed, Jacob Heldenbrand, Ramshankar Venkatakrishnan, Liudmila Mainzer

INTRODUCTION

As genomic sequencing becomes widely implemented in academic and commercial settings, there is a need for new tools to manage the sheer volume of data and the complexity of sequencing analyses. The gold standard for modern genomic variant investigation, the GATK Best Practices pipeline, is a complex workflow with a plethora of different steps. In order to effectively and efficient manage workflows with many samples, workflow management systems are needed to wrap bioinformatics commands that streamline the variant calling process. Here, we compare the various aspects of three popular workflow management systems for large-scale genomic sequencing analyses: Cromwell/WDL, Nextflow, and Swift/T. Though all three serve the same general purpose, their different inbuilt functionalities lend them to different usages. Here, we present a qualitative comparison of the three and a delineation of key comparison metrics, with the hope that it will aid users in selecting the best workflow management system for their high-performance computational needs.

COMPARISON ASPECTS[1]

- **USER INTERFACE: the means by which the user interacts with the software. Possible options include command-line interface (CLI), read-eval-print-loop (REPL), and integrated development environment (IDE) .**
- **CONTAINERIZATION SUPPORT: available methods to virtualize an operating system to run across a host without separate virtual machines**
- **CHECKPOINTING: ability to save application state periodically, allowing for reboot from prior state upon failure**
- **CACHING: ability to store frequently used data in memory to reduce data retrieval time**
- **PORTABILITY: usability of software in a variety of different operating environments**
- **DISTRIBUTED EXECUTION ENGINE: software systems on computer cluster that acts as a single machine. They allow for high computational performance without the need to deal with challenges of parallel computing like task scheduling and fault tolerance.**
- **MODULARITY: ability to divide a program into separate sub-programs, allowing for design flexibility.**
- **ERROR HANDLING STRATEGY: functionalities to address and resolve errors that arise during program execution**
- **PARALLELIZATION: methods to distribute data among multiple computing nodes, allowing many instances of the same function to run at the same time.**
- **SPARK support: GATK is moving from being deployed on the grid, to cloud-based analytics computation using mapreduce in SPARK. Thus SPARK support will be required of future variant calling workflows.**

Nextflow error handling commands:

- **terminate:** terminates execution as soon as error emerges, kills pending processes (default condition)
- **finish:** orderly shutdown of workflow; waits for completion of any submitted processes
- **ignore:** ignores execution errors from processes, sends message to user that event has occurred
- **retry:** re-submit/re-execute process that returned an error condition. Can specify maxErrors and maxRetries (these are disabled as a default)

```
file alignBams[ ] =
  alignRun(sampleLines, variables, failureLog) =>
  logging(variables[“TMPDIR”], timingLog, “alignlogs”);

assert(
        size(alignBams) != 0,
        “FAILURE: The aligned bam array was empty:
         none of the samples finished properly”
);
```

Data logging and user error notifications in Swift/T from https://github.com/ncsa/Swift-T-Variant-Calling

```
#TestBWAMemSamtoolView.wdl

import "BWAMemSamtoolView.wdl" as BWAMEMSAMTOOLVIEW

workflow CallReadMappingTask {
   # Define inputs

   scatter(sample in inputsamples) {
      call BWASAMTOOLSORT.ReadMappingTask {
         input :
            sampleName = sample[0]
      }
   }
}
```

Sample Cromwell/WDL scatter task to perform read mapping in parallel on many samples from https://github.com/ncsa/MayomicsVC

```
inputFiles = Channel
   .fromPath(params.inputFiles)
   .splitText()
   .splitCsv(sep: "\t")
```

Parallelization through channels in Nextflow, where the same process is performed on everything in the channel

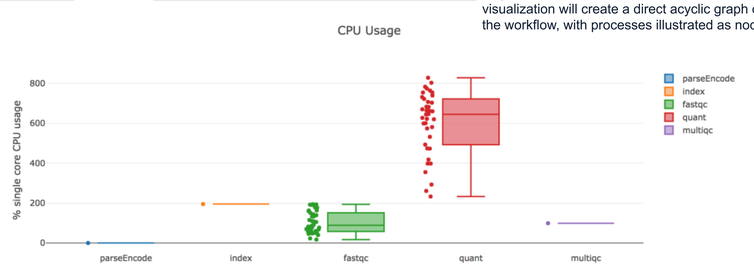
Comparison Aspect	Cromwell/ WDL[2]	Nextflow[3]	Swift/T[1]
Nature of the system	Execution engine	WL and execution	WL and execution
User interface	CLI	CLI, REPL, IDE	CLI
Containerization support	Docker	Docker, Singularity	None
Checkpointing & caching	Yes	Yes	No
Portability	LSF, HTCondor, Google JES	LSF, NQSII, HTCondor, Kubernetes, Ignite, DNAnexus	Cray aprun
Distributed execution engine	Spark	Apache Ignite/ MPI	MPI-based
Modularity	Yes	Yes	Yes
Retry on error	No	Yes	Yes, if failed QC
Error handling strategy	Continue	Continue, retry, terminate, organized finish	Continue upon failing quality control
User notifications	Easy Bash addition	Built-in	Easily implemented
Parallelization	Scatter-gather	Implicit within channels	Implicit & complete
Documentation & community	Extensive, supported by Broad Institute	Extensive, with online forums	Extensive documentation & tutorials
Ease of use	Easy, but requires Bash knowledge	Easy	Difficult, but with many unique features
Tracing & visualization	No	Yes	Some
SPARK support	Yes	No	?

Resource Usage

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

CPU Usage

% Allocated | Raw Usage



Nextflow has built in functionality to create execution, trace, and timeline reports, as well as DAG visualizations. Execution reports (right) consist of a workflow summary, a resource usage graph, and a list of tasks alongside their respective runtime metadata. The DAG visualization will create a direct acyclic graph of the workflow, with processes illustrated as nodes.

task_id	process	tag	status	hash	allocated cpus	%cpu	allocated memory (bytes)	%mem	vmem
1	index	Homo_sapiens.GRCh38.cdna.all.fa.g	COMPLETED	f4/a72585	2	195.0	8589934592	31.9	5272805376
2	parseEncode	/home/pditommaso/projects/rnaseq encode-nf/data/metadata.tsv	COMPLETED	12/bdfd13	1	0.0	-	0.0	17960960
3	fastqc	FASTQC on SRR5210435	COMPLETED	ba/5068a0	2	46.4	6442450944	0.0	4088819712

Workflow Management Systems

Cromwell WDL: A workflow management system intended for scientific workflows, Cromwell/WDL is supported by the Harvard/MIT Broad Institute, which also sets the GATK Best Practices. Intended to be a bridge between complex domain-specific languages and simple scripts, Cromwell/WDL emphasizes performing complex tasks like parallelization in a user-friendly manner suitable for non-programmers.

Nextflow: A domain-specific language and workflow management system intended for complex computational pipelines, Nextflow is based on common programming languages Groovy and Ruby. It is incredibly user-friendly with inbuilt functionalities like error handling and metadata compilation.

Swift/T: A C-like language designed for “high-performance dataflow computing”, Swift/T is intended for computation on a massive scale. Though it contains many unique features like load-balancing, Swift/T programming is less intuitive and may be overwhelming to novice programmers.

CONCLUSIONS

The varying aspects of workflow management systems lend themselves to specific ideal usages.

- Swift/T, with its ability to rapidly perform thousands of small processes, is ideal for exascale analyses.
- In contrast, Cromwell/WDL, backed by the prominent Broad Institute, is best implemented in commercial genomic analyses using the GATK Best Practices pipeline.
- Nextflow’s unique functionalities make it a viable option for both amateur programmers and commercial users who seek to build user-friendly, unbranched genomic analyses.

ACKNOWLEDGEMENTS

The authors would like to thank the Mayo Grand Challenge and the Mayo-Illinois Alliance, the Institute of Genomic Biology, IHSI, and the University of Khartoum for their invaluable contributions to this project.

REFERENCES

1. Ahmed A., Heldenbrand J., Asmann Y. et al. Managing genomic variant calling workflows with Swift/T. Manuscript in review.
2. WDL User Guide. Broad Institute. https://software.broadinstitute.org/wdl/documentation/
3. Di Tommaso P., Chatzou M., Floden EW. et al. Nextflow enables reproducible computational workflows. *Nature Biotechnology* **35**, 316–319(2017). doi:10.1038/nbt.3820