Review of PeerJ manuscript 23190 (v1 and v2)

review-peerj-23190v2.md

Review: PeerJ manuscript 23190v2

- Permalink: https://gist.github.com/stain/eedf7c3a355e347863a2381a2f55e957#file-review-peerj-23190v2-md
- Date: 2018-06-18
- Title: **BioWorkbench: A High-performance Framework for Managing and Analyzing Bioinformatics Experiments**
- Authors: Maria Luiza Mondelli, Thiago Magalhães, Guilherme Loss, Michael Wilde, Ian Foster, Marta Mattoso, Daniel S. Katz, Helio J. C. Barbosa, Ana Tereza R. de Vasconcelos, Kary Ocaña, Luiz M. R Gadelha Jr.
- Call: PeerJ (Bioinformatics tool)
- Submitted preprint: https://arxiv.org/abs/1801.03915 (previous version)
- Resources:
  - BioWorkbench Docker container: https://hub.docker.com/r/malumondelli/bioworkbench/ https://github.com/mmondelli/docker-masters/
  - BioWorkbench source code: https://github.com/mmondelli/bioworkbench https://doi.org/10.5281/zenodo.1243912
  - SwiftPhylo workflow: https://github.com/mmondelli/swift-phylo https://doi.org/10.5281/zenodo.1241164
  - SwiftGECKO workflow: https://github.com/mmondelli/swift-gecko https://doi.org/10.5281/zenodo.1241166
  - RASflow workflow: https://github.com/mmondelli/rasflow https://doi.org/10.5281/zenodo.1243176
- Review by: Stian Soiland-Reyes
- Recommendation: **Minor revisions**
- Outcome: `TODO`

Conflict of interest

- I do have a potential conflict

I have published and still collaborate with Ian Foster on BDBags and previously caGrid Taverna - however neither seem particularly relevant to this manuscript.

I am a member of the Apache Taverna (incubating) project management committee.

I am a mentor on the Nextflow Research Object Google Summer of Code 2018 project.

I am a contributor to PROV and Research Object.

Transparency

- I wish to make my name known as the author of this review

Your review

Basic reporting

The text is well-written and has a good structure.

I think the code citations should be moved to References instead of footnotes (they are now first class citable), but I understand if the GitHub hyperlinks should remain as footnotes so that the PeerJ typesetters can add them as inline hyperlinks.

I have attached some [detailed comments](#Detailed comments) line by line. In summary:

- Described usage is not "HPC" -> use "HTC"
- Fixes in Taverna details
- Fixes in Nextflow details
- How are required tool binaries installed on compute nodes?
- Shiny library vs HPFS-Prof vs new "workbench" code?

- Need deeper NCBI links
- Some grammar and phrasing suggestions

Experimental design

While the Docker container containing the Workflow script is great for reproducibility (as I partially verified under Validity), it seems hard for workflow development. In the way this is described, BioWorkbench is not a framework for writing any scientific workflow, but a framework for capturing a particular workflow, e.g. the provided Docker container captures 3 specific workflows and the Dockerfile is similarly specialized to add their dependencies.

As such I think you can say you are providing a Methodology rather than a Framework. If I am to create my own BioWorkbench that does a slightly different bioinformatics workflow I would need to make my own Dockerfile with my particular dependencies, create a brand new Swift script using the same patterns, build a new Docker image and test it inside the Docker container (or use `vim` inside the container to temporarily fix something).

Such a development methodology is however not mentioned in the paper, and it can be confusing to readers that won't know if BioWorkbench is just something you did (and we can learn from it and play with your demo), something we can repurpose (changing workflows, etc), or something general that can be reused (it can consume any workflow). I think now the truth is somewhere in the middle.

The text should more clearly make the point what maturity level BioWorkbench is at, and show what is the actual contribution. Now the text still gives the impression that BioWorkbench is presented for users, but I would argue that it is this methodology that is the contribution at this stage. I would therefore still be a bit careful about calling it a "framework", perhaps "prototype framework"?

Validity of the findings

Thanks to the updated README I was able to successfully build the Docker image from the bioworkbench recipe. The build takes a while to complete as various R packages are compiled.

I was unable to retrieve the (now automatically built) image from the Docker hub on my laptop, as I ran out of diskspace when writing swift_provenance.db.

I was also unable to retrieve the hub image on a spare server I had available (2x2 TB disk), due to the larger image size (15 GB) and that I was using the Docker LVM production configuration, which had a default max of 10 GB. Setting dm.basesize should improve this; unfortunately I did not have time to reconfigure my server.

This is unlikely to affect desktop Docker users which don't seem to hit this limit, but the authors are still advised to try to split into two Docker images, one smaller base image (tools, BioWorkbench, SwiftProf); and an extension image (using Dockerfile `FROM` that adds the larger pre-populated database.

Within the shell of the Docker image I was able to run the embedded swift-gecko and swift-phylo workflows according to their README files. As in my previous review I have not assesed the bioinformatics validity of these workflows, but confirm that they produced expected kind of outputs.

While intermediate files are kept separate in directories like `run001`, the workflow outputs are added directly to the working directory, which makes it hard to distinguish inputs, workflow, binaries and outputs. This may be improved by restructuring the workflow directories or workflow script.

As not enough reference data (e.g. for SwiftGECKO) was included, I was unable to verify any of the % speedup claims, both workflows finished rather quickly, but I did not find any obvious way to modify the number of concurrent jobs when testing on a multi-core server.

I was NOT able to run the embedded rasflow workflow as it did not contain any example sequences. Attempting to read the Swift workflow "loss_all.swift" is quite confusing, as it is a single flat linear script file that mixes shim scripting, variable declarations, control calls like snpCall() and actual code calls. For instance, the 6-step GATK sub-workflow of Figure 8 is 20 lines in the middle of a single large Swift file. The step names in the Figure 8 do not match the code, I would have expected the Figure steps to appear as functions or so. The authors are recommended to refactor the RasFlow workflow script for greater readability.

The authors argued in the rebuttal that RasFlow inputs are missing due to patient confidentiality. I think readers who try BioWorkbench would be grateful if some smaller example sequences and GFF/GTF files could be added instead, so that the RasFlow workflow also can be tested out of the box, even if it would not be able to reproduce the particular provenance data included.

I was able to access the embedded HPSW-Prof profiling web server. The user interface is somewhat confusing as there are no instructions (what goes into "Script Name"?), but I was eventually able to lower the "Date Range" to 2015 to see older runs and - after a wait - access the embedded provenance traces of executions of RASFlow "loss_all.swift". This showed figures similar to the paper's Figure 3 and Figure 6.

Unlike the Figure 6 the UI's "Level of Parallelism" for RASFlow seems to indicate only 4 concurrent executions (the figure is for SwiftPhylo). The number of cores or configured concurrent activities are not shown, but the table gives detailed execution stats per task of the workflow, e.g. "avg_duration" (unit is seconds?), percent CPU used, file system reads (unit is bytes?), and other parameters like "avg_max_rss", "sys_percent" and "user_percent" that remain unexplained.

As these are features/critizism of the previously published (and cited) HPSW-Prot this review does not go further in detail, but note that the consistent colour scheme is helpful to cross-reference tables and graphs.

The included provenance database do demonstrate the Domain Info added to the database after parsing , which includes 94,651 mutations from the 6 patients (anonymized as S1, S2, etc). I have not assessed if making these mutations public are compatible with patient confidentially, as no Ethical Approval assessment or similar data management plan for the RASOpathies study was cited.

The provenance database did not include data from the SwiftPhylo workflow, which was used for the execution speedup discussion in the paper, so I tried to add it:

```
swiftlog -import-provenance run001
```

as referenced in swift-gecko README. This did however fail for both swift-gecko and swift-phylo:

```
root@2e016c443673:~/swift-phylo# swiftlog -import-provenance run001
Traceback (most recent call last):
  File "//usr/local/swift-0.96.2/bin/swiftlog", line 170, in <module>
    arg, value = (line.split()[arg].strip(',').strip(']').split('='))
ValueError: need more than 1 value to unpack
```

This log file parsing error came from both /usr/local/swift-0.96.2/bin/swiftlog (sic) ~/rasflow/swiftlog/swiftlog (after fixing its 'python2.6' to match 'python2.7' in the image), not working with neither swift-phylo nor swift-gecko.

As mentioned above I was also unable to run RasFlow, and therefore unable to import any fresh workflow runs into the provenance database to visualize them in the web interface.

While the authors have taken significant steps to improve the packaging and documentation of the software, I am still under the impression that this is more of a demonstration snapshot that illustrate a methodology of developing and inspecting bioinformatics workflows with Swift Language and the previously published HPSW-Prof, rather than a General Availability framework (software or workflow package) for bioinformatics researchers.

From the Docker image I was able to run the new Weka instructions; after setting CLASSPATH to the weka.jar; the weka GUI does not work inside Docker. I got the same linear agression algorithm constants as in the paper's listing (1). As Weka output is quite verbose and outside my expertise I did not verify the remaining Weka calculations, but confirmed that they all executed well.

The rebuttal explains that Swift provenance model predates OPM and its successor W3C PROV, and mentions a swift_provenance-to-opm.sh script to generate OPM, which I tested; however this is not mentioned in the manuscript at all, which I find odd given that several pages are about provenance. It can be listed as future work to update this script to generate PROV instead of OPM.

General comments for the author

Hi, I am Stian Soiland-Reyes https://orcid.org/0000-0001-9842-9718 and believe in open reviews.

The complete review is also available at the (secret) URL https://gist.github.com/stain/eedf7c3a355e347863a2381a2f55e957 for easier reading and more hyperlinks.

This revision has improved significantly on most of my previous concerns. I am particularly pleased with the new completeDocker images, improved instructions and Zenodo code citations. Software used by the workflows (e.g. samtools) is now rigourisly versioned and cited, this is also a good improvement.

There are still a couple of issues with the software, I was unable to actually import the provenance to the database, which would seem central to the argument of the paper. This seems to be a bug in the log parsing, perhaps some sensitivity to how the workflow is invoked. Following the README step by step within the Docker image should work.

I was also not able to run the RASFlow workflow as no examples are provided, yet this is the prime workflow of the paper and the subject of the machine learning approach.

While the manuscript now makes it clearer that the Docker image is for demonstration purposes, I think it can still come across like if BioWorkbench is a piece of software to be used by bioinformaticians. I think the manuscript can be honest about the current maturity level of the prototype framework, and rather present the main contribution as a methodology for building and retro-inspect paralellizable workflows using Swift and provenance database.

See https://gist.github.com/stain/eedf7c3a355e347863a2381a2f55e957 for details.

Recommendation

- Minor revisions

Confidential note to the editor

(none)

Detailed comments

Some line-by-line comments:

li51, li60, abstract: "HPC (High-Performance Computing)"

The term HPC is commonly used to describe use of super-computers (e.g. with shared memory architecture or hundred thousands of cores), but the authors are referencing clusters, clouds and computational grids, and in the Result session describe running on 100 cores; which seems not indicative of HPC. I would recommend instead to use the term "HTC" (High-Througput Computing) which is more appropriate for the scaled task execution executions used within this paper.

li94..97: "Parallel execution has an optimization engine that identifies and simplifies complex parts of the workflow structure (Cohen-Boulakia)"

Taverna has no such optimization engine, and this statement is not backed by the cited Cohen-Boulakia paper, rather it describes a separate DistillFlow algorithm that can be applied to Taverna workflows.

There is implicit parallell execution in Taverna, as detailed in https://doi.org/10.1007/978-3-642-13818-8_33 or more informally http://taverna.knowledgeblog.org/2010/12/13/parallel-service-invocations/ and http://dev.mygrid.org.uk/wiki/display/tav250/Implicit+iteration#Implicititeration-Pipelining

I would rephrase, e.g.:

"Taverna has implicit iterations and parallelism, and its workflows can be further optimized to simplify complex parts using the DistillFlow optimization (Cohen-Boulakia)"

p96: "(Taverna) Provenance is collected and stored in a database"

.. and can also be exposed as W3C PROV in a portable Research Object. https://doi.org/10.1016/j.websem.2015.01.003

li97: "workflows can be shared through the myExperiment(1) platform"

instead of URL footnote, perhaps cite https://doi.org/10.1093/nar/gkq429 ?

li102: "domain-specific domain"

I suspect this should be "domain-specific language"

The description of Nextflow fails to mention that Nextflow has a wide range of executors for HTC and HPC environments: https://www.nextflow.io/docs/latest/executor.html

This means li106 needs to be rewritten as it indirectly claims the other systems don't do HPC environments.

li105: "(Nextflow) lacks details of how the provenance is tracked."

Nextflow actually has quite a rich trace report (HTML): https://www.nextflow.io/docs/latest/tracing.html

Fresh of the press - Nextflow is actually also working on standardizing to Research Object and W3C PROV provenance: See https://summerofcode.withgoogle.com/projects/#5390855418937344

li163: "BioWorkbench was developed to support users.."

This gives the indication that BioWorkbench is for General Availability (e.g. PeerJ readers can download it to start using it), but as I point out under Validity it is not yet at such a maturity level. Rephrase, perhaps use softening words like "aims". See also comments under Experimental Designs.

li189: "All expressions in a Swift script whose data dependencies are met are evaluated in parallel".

Rephrase for readability, e.g. "In Swift all expressions are evaluated in parallell as soon as their data dependencies are met"

li192: "allows the same workflow to run on different computing resources"

How do required tool binaries and their dependencies get installed on different computing nodes? E.g. the Dockerfile for BioWorkbench installs lots of tools like R packages, bowtie. Some are included in various bin/ folders as well, e.g. samtools. A compute node however would not be running the same container (or a container at all) and so would not have these tools installed by default.

Workflow systems like Nextflow and Common Workflow Language commonly use Docker, Singularity or BioConda for installing tools needed for a particular workflow, but here this seems to need manual installation on the compute infrastructure? In this case, how is the correct version etc. managed?

li214..li220: A lot of repetition of "domain data"

li218: "this layer" -- by now I have forgotten this whole sectoin is about Data Layer, so perhaps call it that again using italics.

li223..li228: How the key-values are stored in database does not seem relevant (except that it is in the database as key-values). Trim.

li239..li246: This talks about the use of "Shiny Library" -- but is this seems to be based on https://github.com/mmondelli/swift-prof as previously published in the cited HPSW-Prof - is it the same code, or has this been further developed? Either is OK, I just wanted it to be clear. The cloned or modified code for this is for some reason part of "RasFlow" https://github.com/mmondelli/rasflow/tree/master/workbench but I would have expected it to be part of either Swift-Prof or BioWorkbench.

li250: "the RASflow case study, presented below"

As the below section is instead about machine learning, I would change this to "presented in the Results section"

li276: The Weka scripts are now included in the GitHub repo for bioworkbench, so that should be cited.

Note that I was unable to run Weka inside the docker image (it can't run UIs) so perhaps they don't need to be included in the Dockerfile.

li390: a set of 40 files containing bacterial genome sequences, downloaded from NCBI

A general link to NCBI is not particularly detailed, as they hosts so many different databases. Which 40 files? Can they (or the wget/curl script to retrieve them) be added to the Docker image or GitHub repo? I tested SwiftGECKO, but it contains just a single sequence, so I was unable to test the speedup.

li535: "The result is obtained by executing Activity 2 of the workflow, which produces a log file" -- present tense here is confusing as the activity was already executed at query time. Change to "was obtained when the workflow executed Activity 2, which produced a log file that got parsed and added to the provenance database as domain data"

li555: "a reduction of ~77% of the execution time was obtained" -- using how many cores and concurrent jobs?

li559: Do I understand this right here in that the parallelism here was limited by the number of patients, and in effect each patient data is analyzed sequential fashion (but several patients concurrently)? Make this clearer in the text.