Interactive Workflows for Exploratory Data Analysis

Nourhan Elfaramawy

supervised by Prof. Matthias Weidlich,
Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany

Abstract

The analysis of scientific data is often exploratory, meaning that the exact design of a workflow to process data is subject to continuous investigation and redesign. While support for the design of such workflows is manifold, it focuses primarily on reuse, reproducibility, and traceability of analysis results. Yet, it typically relies on static models of workflows that force scientists to wait for completion and restart a workflow repeatedly to explore different design choices. This is inefficient in terms of the invested time and resources.

In this PhD project, we strive for support of user interactions in workflow execution. Our proposal is to extend common workflow models with concepts to define interaction points and possible actions, thereby providing users the flexibility to realize diverse interaction primitives, such as forwarding, repetition, and sample-based exploration. We further outline our initial results on realizing a model for interactive workflows.

Keywords

Exploratory Data Analysis, Scientific Workflows, Interactive Workflows, Snakemake

1. Introduction

In domains such as bio-informatics, remote sensing, and materials science, the analysis of large-scale data is a prerequisite for scientific progress [1]. To this end, complex pipelines of operators, also referred to as *scientific workflows* or *data analysis workflows*, are designed and executed using infrastructures for distributed computation. However, the respective analysis is typically *exploratory*, meaning that it emerges from a scientific process, in which hypotheses are designed and step-wise confirmed or invalidated. Therefore, workflows used for the analysis are also subject to continuous change.

While the importance of supporting the design and execution of workflows is widely recognized [3] [4], existing models and methods focus on reuse, reproducibility, and traceability of analysis results. Workflow engines such as Kepler [5], Galaxy [6], Pegasus [7], Snakemake [8], or Nextflow [9] offer means to specify workflows from reusable building blocks, provide technical abstractions of compute infrastructures, and include functionality for exchange and collaboration in the workflow design. However, they adopt a *static* notion of a workflow as shown in fig. 1(a): A user specifies and configures a workflow, which is subsequently executed.

A static workflow model is inherently limited in its support of interactivity for exploratory data analysis, though. The exploration of design choices and possible changes in the analysis, and hence in the workflow, can only be defined at design time, if at all. In practice, therefore, such exploration is restricted to relatively simple scenarios, such as the definition of parameter sweeps of certain operators, see Nimrod [10] for Kepler and Scalarm [11] for Pegasus. The lack of flexibility in the workflow execution has severe implications. Scientists waste their own time as well as resources of a compute infrastructure as they have to resort to submitting their workflow for execution and waiting for its completion, before repeating it all over again with potentially only minor adjustments.

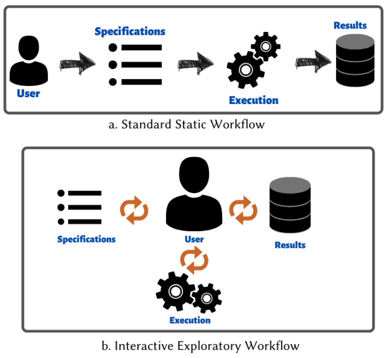

Figure 1: (a) Data analysis based on a traditional workflow that is first defined and then executed; (b) exploratory data analysis that is supported by interactive workflows.

In this paper, we propose to extend common workflow models with interaction capabilities, thereby providing support for exploratory data analysis by a human-in-the-loop model for workflow execution, see fig. 1(b). By enabling scientists to examine the intermediate data produced by a workflow, and to configure and adjust the workflow based on their observations, design choices can

VLDB'22: International Conference on Very Large Databases, September 05–09, 2022, Sydney, Australia
nourhan.elfaramawy@hu-berlin.de (N. Elfaramawy)
0000-0001-9444-5163 (N. Elfaramawy)

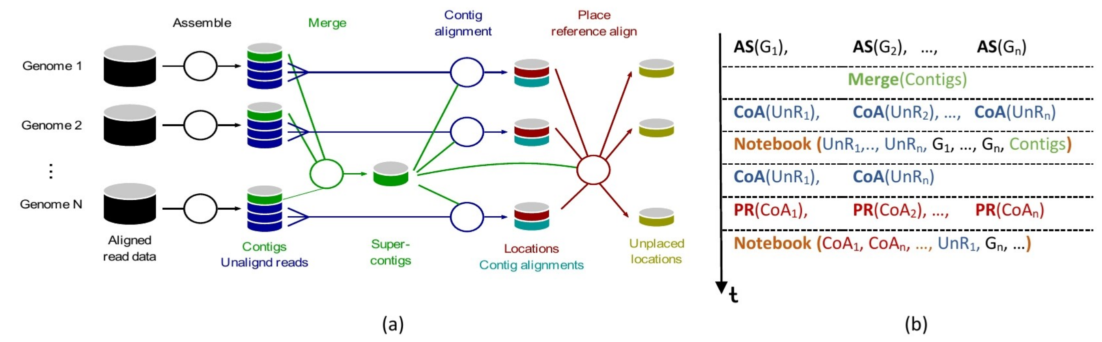

Figure 2: (a) The static PopIns workflow [2] for calling non-reference sequence insertions from many genomes jointly. The workflow includes operators that are executed separately per genome, as well as operators executed for all genomes. (b) A trace of the execution of this workflow once interactions by a user are included: A user explores the results of the *contig alignment* step in a notebook and decides to repeat the respective step before continuing with the workflow.

be explored immediately and systematically, using less compute resources compared to the traditional model.

In order to realize this objective, we set out to answer the following research questions:

(R1) How to support interactions during workflow execution?
  (i) When and where shall a user be able to interact?
  (ii) What are the actions a user may apply?

(R2) How can this dynamic interactive exploratory model be implemented in common workflow systems and infrastructures?

Below, we first introduce a motivating example (§ 2) and review related work (§ 3). Then, we outline our ideas to answer the above questions. This includes a model for interactive workflows including notions of interaction points and actions (§ 4). Moreover, we elaborate on our preliminary results in terms of realizing this model in an existing workflow engine (§ 5), before we conclude (§ 6).

2. Motivation

We illustrate the need for interactive workflows for exploratory data analysis with an example from a bioinformatics field, in particular genomics. Here, recent advances in DNA sequencing led to the wide-spread availability of large volumes of genome sequence data. Specific research questions in this area, for instance, relate to the identification of structural variants (SVs) in genomes. In particular, tools such as PopIns [2] and PopDel [12] have been developed to detect large insertion and deletion variants in whole-genome sequencing (WGS) data of hundreds to tens of thousands of individuals.

A tool such as PopIns actually implements a workflow of multiple operators that are applied to the genome of each individual separately, or that combine data from multiple individuals, as shown in fig. 2(a). In general, this workflow starts with the *assemble* (AS) operator that takes a genome (G) as input and reconstructs contigs from unalgined reads (UnR), i.e., a set of unaligned reads from a sequencing dataset of a single individual and its task is to reconstruct a set of contigs, representing candidate sequences of insertions. Then, a *merge* operator combines the contigs from different genomes, which results in so-called super-contigs. Those are used in the *contig alignment* (CoA) step which aligns the unaligned reads to the supercontigs and outputs candidate locations (approximate positions) of the supercontigs in the reference genome. The *placement of the reference alignment* (PR) step identifies precise insertion positions of the supercontigs in the reference genome.

To summarize, the output of each step in the PopIns workflow is the input of the next one, and the behaviour of the system depends on the input/output quality. Therefore, it is necessary for the scientist to examine the intermediate results and observe the system behaviour at various points during execution. This may result in repeating or skipping some steps. These design choices make genome analysis workflows good examples for exploratory data analysis. Yet, the example also illustrates that, if no ground truth is available, the design of the respective workflows cannot be optimized automatically.

To support such application scenarios effectively, we envision a model of an interactive workflow to enable an execution as sketched in fig. 2(b). The execution starts with the assemble and merge steps, followed by contig alignment. However, we envision the definition of an *interaction point*, so that the execution of the workflow is paused. Then, a frontend, here denoted as a *notebook*, shall enable the visualization of the contig alignments and provide descriptive statistics over unaligned reads.

Let us now assume that, based on some observations, a user takes the *action* to repeat the separation of unaligned reads with a different algorithm. However, the user may also update the definition of the interaction point that may pause the workflow after that specific step: It may be assigned a condition based on statistics over the unaligned reads, so that it is triggered only when this condition is met. Afterwards another interaction point is activated, enabling the user to investigate all intermediate results obtained so far, i.e., contig alignments, unaligned reads and a genome sample.

These above interaction points provide the user with the flexibility to make decisions and take actions on the workflow execution based on intermediate results. This way, a user can incorporate immediate and systematic changes at runtime, which will not only save time and computational resources, but could also act as an early indicator for technical errors during workflow execution.

3. Related Work

Scientific workflows help scientists to manage and organize their data-driven analysis [13]. There are many workflow engines that scientists rely on, such as Kepler [5], Galaxy [6], Pegasus [7], Snakemake [8], and Nextflow [9]. These engines provide ease-of-use through user interfaces, graphically or script-based, and catalogues of standardized data preprocessing techniques. As mentioned above, some engines provide some limited support for exploratory analysis, e.g., for parameter sweeps. Also, notably, dynamic control of iterations in workflows based on changes of the processed data by a user was proposed in [14]. However, there is a gap in terms of expressive models to support generic interactivity during runtime, i.e., while a workflow is executed.

Data flow optimization is related as it targets some of the challenges stemming from workflows used for exploratory analysis. For instance, meta-dataflows (MDFs), introduced in [15], to improve the task scheduling and memory allocation in exploratory analysis. Data access patterns caused by exploratory analysis may also benefit from caching layers, such as Tachyon [16].

Debugging of data processing pipelines received increased interest in the data management community recently [17]. For example, Dagger [18] provides interactivity through debugging primitives in data-driven pipelines. DataExposer [19], in turn, helps to identify properties that can be considered to be root causes of performance degradation, or system failure due to data. Yet, most of this work focuses on debugging at the data-level, rather than the control-flow level.

4. Model of Interactive Workflows

Below, we describe our take on research question (R1), i.e., how to model interactive workflows. We first propose an extension to the common workflow model (§ 4.1), before elaborating on exploration primitives (§ 4.2).

4.1. Interaction Points and Actions

As a starting point, we consider a traditional model of a workflow, see [4]. It defines a workflow as a DAG, where vertices denote operators and edges denote data dependencies between the references to the datasets consumed or produced by operators, also known as input and output ports. A state of such a workflow is then given by a binding of specific files to these input and output ports.

As hinted at already above, the question of how to model interactive workflows can be split into two parts, *when* and *where* to interact; and *what* actions to apply. We therefore propose to extend the traditional workflow model with two concepts, as follows:

Interaction points indicate that the workflow execution shall be paused for a user to explore the current state in terms of the data generated so far, which potentially involves executing some additional analysis to get insightful visualizations or to compute descriptive statistics on the intermediate results. Such an interaction point is given by an edge of the workflow DAG and, potentially, a condition. The latter may refer to a state of workflow execution (e.g., checking the number of lines in a data file) or meta-data (e.g., checking the execution time of an operator). The semantics of an interaction point (IP) are summarized as follows: Upon completing the execution of the operator that is the source of the respective edge, the workflow engine checks the condition and, if it is true, does not continue execution with the operator that is the target of the edge, but waits for user input.

Actions indicate how the user intends to continue the execution of a workflow once an interaction point is reached. To this end, we consider different types of actions, including:

- Revise interaction points: The set of interaction points defined for the workflow is updated.
- Revise workflow: The structure of the workflow in terms of operators and data dependencies is updated.
- Continuation: Workflow execution continues with the operator following the interaction point.
- Skipping: Workflow execution continues based on the workflow DAG, but skips over the specified operators when doing so, i.e., the output ports of skipped operators denote empty datasets.
- Rewind: Workflow execution continues from an earlier state, which is identified by an operator in the workflow DAG.

Naturally, the actions to revise interaction points and the workflow shall be combined with a continuation, skipping, or rewind action. Moreover, we note that the actions impose certain consistency requirements to enable proper workflow execution, e.g., in terms of reachability of operators in the workflow DAG and the realization of data dependencies.

4.2. Exploration Primitives

The extension of a model for workflows realized by interaction points and actions enables us to support various primitives often found in exploratory workflows. Below, we outline how this support is achieved for some of these exploration primitives:

Fast-forward: Based on properties of some intermediate results, a user may want to fast-forward the workflow execution to save time and compute resources. An example would be a sequence of operators to implement noise filtering, which may not be needed if the data variance stays within certain limits. This is enabled by defining an interaction point to decide on fast-forwarding (e.g., to compute the variance), which may then be realized through a skipping action.

Repetition: A user may want to repeat a certain step, or a set thereof, before advancing with the workflow execution, e.g., to fine the configuration of operators. Support for such repetition is limited in common workflow management systems. The reason being that most of them adopt an execution model based on a DAG, which prevents the definition of a cycles in the workflow structure. Using the concepts envisioned for interactive workflows, we support repetitions by defining an interaction point at which a user may decide on a rewind action or a continuation action.

Sample-based exploration: Another pattern in exploratory analysis is that a user wants to test their workflow on a subset of data, before applying it to the complete dataset in order to save time and compute resources. An example would be the calibration of some data transformations by fitting a statistical model. Here, the fit of various models may first be explored using a sample of the data. Based thereon, the model is adopted to transform the whole dataset, or the workflow is altered to incorporate a different transformation. This is realized by defining an interaction point after the sample was processed, with the possible actions being to rewind and re-execute the workflow with the complete dataset, or to revise the workflow by replacing the transformation operator.

5. Towards a Realization

Technical environment: To realize our conceptual model for interactive workflows (§ 4), we choose Snakemake, a state-of-the-art rule-based workflow management system. Here, a workflow is defined by a set of rules. Each rule denotes a task or operator and specifies how to create sets of output files from sets of input files. Then, the engine establishes the dependencies between the rules by matching file names. In Snakemake, when starting a workflow, these rules are used to create a DAG as the basis for execution. However, this also means that an adaptation of the rules and, hence, the DAG is not possible after the start of workflow execution.

However, even though the DAG cannot be altered at runtime, Snakemake provides limited support for interactivity. That is, a Jupyter notebook [20], a popular Python-based computational environment, can easily be integrated with Snakemake. Such a notebook combines code snippets, documentation, as well as plots into a single document. The integration in Snakemake is realized via dedicated rules, which, once executed, start a notebook for a user to work with. The workflow only continues execution according to the constructed DAG once the user closes the respective notebook.

Preliminary results: To test the feasibility of our ideas, we implemented parts of our proposed model for interactive workflows in Snakemake using the notebook integration, and applied it to the PopIns workflow mentioned earlier (§ 2). Specifically, we added interaction points in the workflow by rules that start a notebook, as sketched in fig. 2(b). These notebooks then enable access to the intermediate results (e.g., unaligned reads and contigs in our example). Moreover, we realized the aforementioned rewind and continuation actions, which, once triggered by a user in the notebook, enable the repetition of particular steps in the workflow. Since the DAG constructed by Snakemake is immutable, our solution for the rewind action is based on an abortion of the current workflow instance and the creation of a new instance, while the control-flow in the new instance is guided by the automated creation and deletion of dedicated files.

Applied in the context of the PopIns workflow, our prototype, despite its limited support of the envisioned model, highlights the benefits of offering interactivity in workflow execution. Users can explore and examine intermediate results at runtime, and realize common exploration primitives directly, rather than being delayed by the need to wait for workflow completion.

6. Conclusions

In this work, we outlined the need to support interactivity in workflows for scientific data. To address this need, we outlined a model for interactive workflows, which is based on interaction points and actions. We further reported on our preliminary results of realizing this model in Snakemake, a rule-based workflow engine, and its integration with Jupyter notebooks. Specifically, we adopted the implementation in the Popins workflow for structural variant calling in genomics.

Having a first version of a model for interactive workflows, our research plan involves the following phases: First, we intend to study the realization of further exploration primitives using interaction points and actions. This way, we also seek to understand whether our model shall incorporate more expressive actions. Second, we aim to ensure that our implementation is supported not only in the stand-alone execution mode of Snakemake, but can also be employed for cluster-based execution. Third, our goal is to provide implementation strategies for our model of interactive workflows for other workflow engines, such as Nextflow.

Acknowledgments

This work is funded by the German Research Foundation (DFG), Project-ID 414984028, SFB 1404 FONDA.

References

[1] U. Leser, M. Hilbrich, C. Draxl, P. Eisert, L. Grunske, P. Hostert, D. Kainmüller, O. Kao, B. Kehr, T. Kehrer, C. Koch, V. Markl, H. Meyerhenke, T. Rabl, A. Reinefeld, K. Reinert, K. Ritter, B. Scheuermann, F. Schintke, N. Schweikardt, M. Weidlich, The Collaborative Research Center FONDA, Datenbank-Spektrum (2021). doi:10.1007/s13222-021-00397-5.

[2] B. Kehr, P. Melsted, B. V. Halldórsson, Popins: population-scale detection of novel sequence insertions, Bioinform. 32 (2016) 961–967.

[3] A. Barker, J. van Hemert, Scientific workflow: A survey and research directions, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 746–753.

[4] Workflows and e-science: An overview of workflow system features and capabilities, Future Generation Computer Systems 25 (2009) 528–540. doi:https://doi.org/10.1016/j.future.2008.06.012.

[5] D. Barseghian, I. Altintas, M. B. Jones, D. Crawl, N. Potter, J. Gallagher, P. Cornillon, M. Schildhauer, E. T. Borer, E. W. Seabloom, P. R. Hosseini, Workflows and extensions to the kepler scientific workflow system to support environmental sensor data access and analysis, Ecological Informatics 5 (2010).

[6] D. Blankenberg, G. V. Kuster, N. Coraor, G. Ananda, R. Lazarus, M. Mangan, A. Nekrutenko, J. Taylor, Galaxy: A web-based genome analysis tool for experimentalists, Current Protocols in Molecular Biology 89 (2010).

[7] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, K. Wenger, Pegasus, a workflow management system for science automation, Future Generation Computer Systems 46 (2015).

[8] J. Köster, S. Rahmann, Snakemake - a scalable bioinformatics workflow engine, Bioinform. 34 (2018) 3600.

[9] P. D. Tommaso, E. W. Floden, C. Magis, E. Palumbo, C. Notredame, [nextflow, an efficient tool to improve computation numerical stability in genomic analysis] 211 (2017). doi:10.1051/jbio/2017029.

[10] D. Abramson, B. Bethwaite, C. Enticott, S. Garic, T. Peachey, Parameter space exploration using scientific workflows, in: G. Allen, J. Nabrzyski, E. Seidel, G. D. van Albada, J. Dongarra, P. M. A. Sloot (Eds.), Computational Science – ICCS 2009, Springer Berlin Heidelberg, 2009.

[11] D. Król, J. Kitowski, R. F. da Silva, G. Juve, K. Vahi, M. Rynge, E. Deelman, Science automation in practice: Performance data farming in workflows, in: 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), 2016.

[12] S. Niehus, H. Jónsson, J. Schönberger, E. Björnsson, D. Beyter, H. P. Eggertsson, P. Sulem, K. Stefánsson, B. V. Halldórsson, B. Kehr, Popdel identifies medium-size deletions jointly in tens of thousands of genomes, bioRxiv (2020).

[13] C. S. Liew, M. P. Atkinson, M. Galea, T. F. Ang, P. Martin, J. I. V. Hemert, Scientific workflows: Moving across paradigms, ACM Comput. Surv. 49 (2016).

[14] J. Dias, G. Guerra, F. Rochinha, A. L. Coutinho, P. Valduriez, M. Mattoso, Data-centric iteration in dynamic workflows, Future Gener. Comput. Syst. 46 (2015) 114–126.

[15] R. Fernandez, W. Culhane, P. Watcharapichat, M. Weidlich, V. Morales, P. Pietzuch, Metadataflows: Efficient exploratory dataflow jobs, 2018, pp. 1157–1172.

[16] H. Li, A. Ghodsi, M. Zaharia, S. Shenker, I. Stoica, Tachyon: Reliable, memory speed storage for cluster computing frameworks, SOCC '14, Association for Computing Machinery, 2014, p. 1–15. doi:10.1145/2670979.2670985.

[17] M. Brachmann, W. Spoth, Y. Yang, C. Bautista, S. Castelo, S. Feng, J. Freire, B. Glavic, O. Kennedy, H. Müeller, R. Rampin, Data debugging and exploration with vizier, 2019, pp. 1877–1880.

[18] Y. Yang, M. Youill, M. E. Woicik, Y. Liu, X. Yu, M. Serafini, A. Aboulnaga, M. Stonebraker, Flexpushdowndb: Hybrid pushdown and caching in a cloud DBMS, Proc. VLDB Endow. 14 (2021) 2101–2113.

[19] S. Galhotra, A. Fariha, R. Lourenço, J. Freire, A. Meliou, D. Srivastava, Dataexposer: Exposing disconnect between data and systems, CoRR abs/2105.06058 (2021).

[20] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, C. Willing, Jupyter notebooks – a publishing format for reproducible computational workflows, in: F. Loizides, B. Schmidt (Eds.), Positioning and Power in Academic Publishing: Players, Agents and Agendas, IOS Press, 2016, pp. 87 – 90.