Big Data Research ••• (••••) •••–•••

Contents lists available at ScienceDirect

Big Data Research

www.elsevier.com/locate/bdr

Selective and Recurring Re-computation of Big Data Analytics Tasks: Insights from a Genomics Case Study ☆

Jacek Cała *, Paolo Missier

School of Computing, Newcastle University, Newcastle upon Tyne, UK

ARTICLE INFO

Article history:
Received 18 November 2017
Received in revised form 7 June 2018
Accepted 20 June 2018
Available online xxxx

Keywords:
Re-computation
Knowledge decay
Big data analysis
Genomics

ABSTRACT

The value of knowledge assets generated by analytics processes using Data Science techniques tends to decay over time, as a consequence of changes in the elements the process depends on: external data sources, libraries, and system dependencies. For large-scale problems, refreshing those outcomes through greedy re-computation is both expensive and inefficient, as some changes have limited impact. In this paper we address the problem of refreshing past process outcomes *selectively*, that is, by trying to identify the subset of outcomes that will have been affected by a change, and by only re-executing fragments of the original process. We propose a technical approach to address the selective re-computation problem by combining multiple techniques, and present an extensive experimental study in Genomics, namely variant calling and their clinical interpretation, to show its effectiveness. In this case study, we are able to decrease the number of required re-computations on a cohort of individuals from 495 (blind) down to 71, and that we can reduce runtime by at least 60% relative to the naïve blind approach, and in some cases by 90%. Starting from this experience, we then propose a blueprint for a generic re-computation meta-process that makes use of process history metadata to make informed decisions about selective re-computations in reaction to a variety of changes in the data.

1. Introduction

In Data Science applications, the insights generated by resource-intensive data analytics processes may become outdated as a consequence of changes in any of the elements involved in the process. Changes that cause instability include updates to reference data sources, to software libraries, and changes to system dependencies, as well as to the structure of the process itself. We address the problem of efficiently restoring the currency of analytics outcomes in the presence of instability. This involves a trade-off between the recurring cost of process update and re-execution in the presence of changes on one side, and the diminishing value of its obsolete outcomes, on the other. Addressing the problem therefore requires knowledge of the impact of a change, that is, to which extent the change invalidates the analysis, as well as of the cost involved in upgrading the process and running the analysis again. Additionally, it may be possible to optimise the re-analysis given prior outcomes and detailed knowledge of, and control over, the analysis process.

1.1. Motivation: genomics data processing

In this paper we focus specifically on Genomics data processing, as it is a relevant and paradigmatic case study for experimenting with general re-computation strategies. Next Generation Sequencing (NGS) pipelines are increasingly employed to analyse individuals' exomes (the coding region of genes, representing about 1% of the genome), and more recently whole genomes, to extract insight into suspected genetic diseases, or to establish genetic risk factors associated with some of the most severe human diseases [1–3]. NGS pipelines provide an ideal testbed to study the re-computation problem, as they are relatively unstable and are used to process large cohorts of individual cases. They are also resource-intensive: exome files are of the order of 10 GB each, and a batch of 20–40 exomes is required for the results to be significant. Each 1TB+ input batch requires over 100 CPU-hours to process. Specific performance figures for our own pipeline implementation, which runs on the Azure cloud, can be found in [4].

While the cost and execution time associated to a single execution of these pipelines is decreasing over time [5,4], recent advances in preventive and personalised medicine [6] translate into ambitious plans to deploy genomics analysis at population scale. At the same time, although relatively stable *best practices* are available

☆ This article belongs to Special Issue: Medical Data Analytics.

* Corresponding author.
E-mail addresses: Jacek.Cala@ncl.ac.uk (J. Cała), Paolo.Missier@ncl.ac.uk (P. Missier).

https://doi.org/10.1016/j.bdr.2018.06.001
2214-5796/

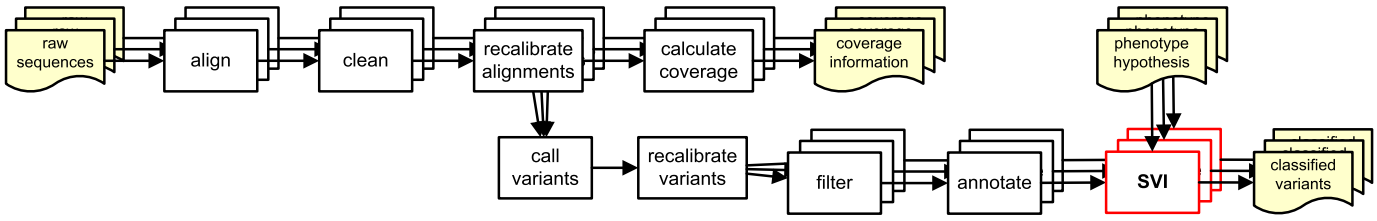

Fig. 1. The Next Generation Sequencing pipeline; highlighted is the variant classification step.

to describe the general structure of the analysis process,[1] their implementations make use of algorithms and tools that are subject to frequent new releases, as well as of reference databases that undergo regular revisions.

In this setting, failing to react to important changes results in missed opportunities to improve on an individual's genetic diagnosis. On the other hand, over-reacting to each and every change is impractical and inefficient, as in many cases the benefits of refresh may be marginal. Using genomics data processing as a case study, we are therefore motivated to explore techniques for *selective* and *incremental re-computation* that optimise the use of the available computing resources vis-à-vis the expected benefit of knowledge refresh on a population of prior outcomes.

1.2. Reacting to changes: a meta-process

To clarify the meaning of *selectivity* and *incremental re-computation* in this context, consider: a collection C of *cases*, e.g., a cohort of individuals' genomes; an analysis process P, e.g. an NGS pipeline; a collection of executions of P on each input $x_i \in C$, which generate corresponding outcomes y_i with processing cost c_i; and a set $D = \{d_1 \dots d_m\}$ of versioned dependencies, i.e., software libraries or reference databases. When a new version D'_j of a dependency $D_j \in D$ becomes available, we expect the change $D_j \rightarrow D'_j$ to have different impact on different outputs y_i computed at some earlier time: some of these outputs will be unaffected, while others will be partially or completely invalidated, as we will show in examples later.

We are going to define *impact* in terms of a change on a specific output y_i in terms of some type-specific *diff* functions that compute the differences between two versions y_i, y'_i of an output. Assuming that expected impact can be estimated, we define the *scope* of the change as the subset of $C' \subseteq C$ of inputs x_i such that the change will have non-zero impact on the corresponding output y_i, and the *selectivity* of the change as $1 - \frac{|C'|}{|C|}$. Those $x_i \in C$ that are within the scope of a change are candidates for re-computation, and it may be possible to prioritise them using knowledge of the cost c_i of their earlier processing, the quantified extent of impact, along with domain-specific knowledge of their relative importance (for instance, more severe genetic diagnoses). Such considerations, however, are beyond the scope of this paper.

Instead, here we study techniques to (i) estimate the scope of a change, without having to recompute each output, and (ii) perform *incremental* re-computation: given a *white box* specification of P, for instance as a script or as a workflow, we want to efficiently identify the minimal fragment of P that is affected by the change, in order to optimise the re-computation of the x_i that are within the scope of the change. We define such techniques within the framework of the `ReComp` *meta-process*. `ReComp` takes as input a history of prior analysis and a change event, as indicated above, and controls the incremental re-execution of the underlying process P on selected inputs that are within the scope of the change.

Not all scenarios involving C, P, and changes in P's dependencies are equally suitable for optimisation using `ReComp`, however. Specifically, `ReComp` is most effective when changes have high selectivity (only few of the cases are affected), when process P is a *white box*; and when the change affects only a few of P's components, providing scope for incremental re-computation. In the next section we select our target case study following these three requirements, by analysing three scenarios involving different reference data and software tool changes within the realm of Genomics. Firstly, however, we must briefly describe NGS pipelines.

1.3. Variant calling and interpretation

Fig. 1 depicts the anatomy of the NGS pipeline implementation available from our lab. It consists of two main phases: (i) exome analysis and variant calling and annotation [4], and (ii) variant interpretation [7]. The first phase closely follows the guidelines issued by the Broad Institute.[2] It takes a batch of raw input exomes and, for each of them, produces a corresponding list of variants, or *mutations*, defined relative to the current reference human genome (in the order of tens of thousands). Particularly critical in this phase are the choices of reference genome, currently at version `h19`, and the choice and version of the *variant caller*. Currently we use FreeBayes [8], one of several such algorithms [9]. At the end of this phase, each variant will have been annotated using a variety of statistical predictors of the likelihood that the variant contributes to a specific genetic disease.

Only a very small fraction of these variants are deleterious, however. The second phase, which we have called Simple Variant Interpretation (SVI in the figure), aims to identify those the few tens of variants that may be responsible for an individual's phenotype, i.e., the manifestation of a suspected genetic disease. In addition to using the predictors, SVI also makes use of databases that associate phenotype descriptions with sets of genes that are known to be broadly implicated in the phenotypes, such as OMIM GeneMap.[3] It also uses databases of known variants and their deleteriousness such as NCBI ClinVar,[4] HGMD,[5] and possibly others.[6]

In more detail, the SVI portion of the pipeline consists of three main steps (Fig. 2): (1) mapping the user-provided clinical terms that describe a patient's phenotype to a set of relevant genes (*genes-in-scope*), (2) selection of those variants that are in scope, that is, the subset of the patient's variants that are located on the genes-in-scope, and (3) annotation and classification of the variants-in-scope according to their expected pathogenicity. Classification consists of a simple traffic-light system {red, green, and amber} to denote pathogenic, benign and variants of unknown or uncertain pathogenicity, respectively. In this process, the class of a

[1] https://software.broadinstitute.org/gatk/best-practices.

[2] https://software.broadinstitute.org/gatk/best-practices.

[3] http://data.omim.org.

[4] https://www.ncbi.nlm.nih.gov/clinvar.

[5] http://www.hgmd.cf.ac.uk.

[6] http://grenada.lumc.nl/LSDB_list/lsdbs.

Please cite this article in press as: J. Cała, P. Missier, Selective and Recurring Re-computation of Big Data Analytics Tasks: Insights from a Genomics Case Study, Big Data Res. (2018), https://doi.org/10.1016/j.bdr.2018.06.001