Informatics in Medicine Unlocked 32 (2022) 101024

Contents lists available at ScienceDirect

Informatics in Medicine Unlocked

journal homepage: www.elsevier.com/locate/imu

A taxonomy of tools and approaches for distributed genomic analyses

Wilmer Garzón [a,b,*], Luis Benavides [a], Alban Gaignard [c], Richard Redon [c], Mario Südholt [b]

[a] Deparment of Computer Engineering, Escuela Colombiana de Ingeniería "Julio Garavito", AK. 45 No. 205 - 59, Bogotá, Colombia
[b] Deparment of Computer Science, IMT Atlantique, 4 Rue Alfred Kastler, Nantes, France
[c] L'Institut du Thorax, Université de Nantes, CNRS, INSERM, Nantes, France

ARTICLE INFO

Keywords:
Distributed biomedical analyses
Fully distributed collaborations
Reproducibility
Scalability
Multi-site analyses
Distributed workflow analyses

ABSTRACT

The amount of biomedical data collected and stored has grown significantly. Analyzing these extensive amounts of data cannot be done by individuals or single organizations anymore. Thus, the scientific community is creating global collaborative efforts to analyze these data. However, biomedical data is subject to several legal and socio-economic restrictions hindering the possibilities for research collaboration. In this paper, we argue that researchers require new tools and techniques to address the restrictions and needs of global scientific collaborations over geo-distributed biomedical data. These tools and techniques must support what we call Fully Distributed Collaborations (FDC), which are research endeavors that harness means to exploit and analyze massive biomedical information collaboratively while respecting legal and socio-economical restrictions. This paper first motivates and discusses the requirements of FDCs in the context of a research collaboration on the development of diagnostic and predictive tools for the risk of intracranial aneurysm formation and rupture (the ICAN project). The paper then presents a taxonomy classifying the current tools and techniques for biomedical analysis with respect to the proposed requirements. The taxonomy considers three key architectural features to support FDC scenarios: data and computation placement, Privacy and Security, and Performance and Scalability. The review reveals new research opportunities to design tools and techniques for multi-site analyses encouraging scientific collaborations while mitigating technical and legal constraints.

1. Introduction

Following the completion of the human genome project in 2003, the emergence of next-generation sequencing techniques has resulted in an immense increase in data production in the genomics field [116]. Due to technological progress, the same phenomenon is observed in other biomedical fields, such as functional phenotyping by bioimaging. Exploiting and interpreting these extensive amounts of data often cannot be done anymore by individuals or single organizations. It has to be performed collaboratively. Today's predominant architectural model for biomedical collaborative analyses consist of centralizing the underlying data and performing analyses through supercomputers or cluster infrastructures located at a single or a small number of organizations [107]. Nevertheless, this collaborative model is very restrictive and rigid. Recently, the need for more widely distributed collaborations has been noted [12,13,74,98]. Several arguments favor a higher degree of distribution: more and more organizations dispose of high-performance infrastructures for large-scale analyses, biomedical local data should be kept private, and massive data transfers are too time-consuming.

Such distributed cooperations, which we call fully-distributed collaborations (FDCs) in the following, are research endeavors that harness means to exploit and analyze massive biomedical information collaboratively over geo-distributed infrastructures. Thus, FDCs require tools and techniques for collaboration that can use advanced distributed (data and computation) architectures to cope with complex socio-technical constraints and heterogeneous networks. FDCs promise to enable more powerful biomedical analyses defined in distributed workflows operating over large volumes of shared public and private data. FDC analyses will promote cooperation among geo-distributed research groups or organizations, each typically subject to, possibly local, constraints stemming from legal frameworks, security constraints, sensitive private data, and locally-available infrastructures.

Approaches and tools for fully distributed collaborative biomedical analyses are rare today. Most existing approaches and tools process data at individual or a small number of locations using efficient frameworks for large-scale computations, such as MapReduce [21,55,105].

Abbreviations: FDCs, Fully Distributed Collaborations; BATTs, Biomedical Analytical Tools and Techniques; SWFMS, Scientific Workflow Management System.

* Corresponding author. Deparment of Computer Engineering, Escuela Colombiana de Ingeniería "Julio Garavito", AK. 45 No. 205 - 59, Bogotá, Colombia.
E-mail address: wilmer.garzon@escuelaing.edu.co (W. Garzón).

https://doi.org/10.1016/j.imu.2022.101024
Received 16 June 2022; Accepted 15 July 2022
Available online 31 July 2022

MapReduce-based frameworks like Hadoop and Spark are highly efficient for distributed processing across a large number of machines orchestrated by a central node. But, these frameworks have limitations when processing geographically distributed data while maintaining quality attributes such as scalability, consistency, and performance [41]. Similarly, workflow systems, another popular tool used in the biomedical field, allow scientists to define analyses in terms of tasks as well as task dependencies and dataflows [76]. They also frequently support portability across different execution environments like grids and clusters [102,142]. But again, current workflow systems do not support FDC scenarios because they are designed to cope with current state-of-the-art infrastructure. They are also not designed to cope with the complex requirements of international biomedical cooperation, lacking mechanisms for decentralization, distributed computations, and security/privacy requirements [98]. Fog and edge computing have also been proposed as infrastructures to address research collaboration [4,18,122,128]. They enable placing computations and data closer to the owners, but suffer the same limitations as the techniques discussed above. They lack the means for defining complex scientific collaborations while preserving data privacy and socioeconomic restrictions. Thus, they address only one side of the solution through support for computation placement and data gathering (sensors).

In order to design the next-generation tools to address future biomedical challenges,[1] we need extensive knowledge of the state-of-the-art on biomedical tools and data analytic techniques Some authors address this problem by studying and categorizing biomedical tools based on the underlying framework they use (a technology point of view) or according to the biomedical problem they solve (a biomedical point of view) [21,55,105]. However, these approaches are not comprehensive enough to address the problems stated above in the context of FDC scenarios.

In order to close this gap, this article investigates tools and approaches for distributed biomedical analyses. We have restricted the study of biomedical analysis techniques to genomics due to its importance in biomedical research, the vast amount of unprocessed data that has been generated in recent years, and because we think it is representative enough of the practices of research collaborations in the entire biomedical field. We argue that Fully Distributed Collaborations techniques and approaches are general enough to apply to any biomedical data type, such as clinical data, imaging data, or biological samples.

Concretely, in this paper, we present the following contributions:

- We motivate the FDCs as an improvement of centralized biomedical analyses. We show current limitations and constraints in the context of the ICAN project [6], a collaborative project between French medical institutions providing clinical records, medical images, and genetic data collected from notably through biological samples [15].
- We present a taxonomy of existing Biomedical Analytical Tools and Techniques (BATTs). It classifies current BATTs from three different perspectives: (i) biomedical problems that are being solved, (ii) the tool support provided for biomedical analyses, and (iii) support for distributed cooperation, notably in terms of types of distribution offered, interoperability properties, and reproducibility properties.
- We investigate architectural support for computation and data placement, privacy and security properties, as well as scalability and performance properties.
- Finally, we discuss lessons learned and some open issues in the domain of biomedical analytic tools and techniques.

This paper is structured as follows. Sec. 2 presents a motivational case study and some reasons for processing biomedical data in FDC scenarios. Sec. 3 details the methodology of the literature search conducted in this review. Sec. 4 presents a comprehensive taxonomy of models, frameworks, and tools for biomedical analyses. In Sec. 5, we discuss lessons learned and major open challenges identified based on our survey. Finally, our conclusions are presented in Sec. 6.

2. Motivation

The amount of biological and clinical data collected and stored continues to grow rapidly [116]. Part of this data is exploited, but most of it is not being processed or analyzed. For example, in the case of the European Bioinformatics Institute (EBI) the total unprocessed data exceeded 160 PB in 2018, growing more than 200% from 2015 [32]. The availability of such amounts of unprocessed data presents new research opportunities and challenges for biomedical researchers. For instance, many research collaborations in the biomedical community have benefited from data sharing of ever larger data sets. Genome-Wide Association (GWA) studies [89], a prime example, serve for the analysis of large sets of genomic information from many individuals in order to identify genetic variants that may be associated with specific traits (*e.g.*, characterizing a specific human disease). GWA studies are performed using large amounts of genomic data shared via public repositories [90].

Contrary to a frequent assumption concerning distributed research collaborations involving biomedical data, the information is not always accessible and available to all parties involved in the collaboration. Consider, for example, the ICAN project [15] that involves 34 french hospitals and research centers working together on understanding the pathology of intracranial aneurysms. As part of this cooperation, all parties share clinical records, imaging data, and biological samples to extract DNA data. These three data sets are then jointly analyzed at the two processing sites. However, even though the project is a french national effort financed by the government, data sharing is severely restricted by legal and technical issues. These restrictions impose limitations on the collaboration patterns available for biomedical researchers. Current tools and models for biomedical cooperations rarely support such distributed collaborations. We argue that computational tools must support more sophisticated collaboration patterns while respecting legal, socio-economic, and technical restrictions that abound in biomedical cooperations. This section motivates the need for more sophisticated tools by analyzing the restrictions biomedical data is subject to, starting from the ICAN project.

2.1. The ICAN Project

The IntraCranial ANeurysms (ICAN) project aims to develop diagnostic and predictive tools for the risk of intracranial aneurysm formation and rupture from three types of data: clinical data, imaging data, and biological samples [15]. Fig. 1 shows the parties involved in the project. The figure shows that most sites provide (the three types of) data, while only hospitals and research institutions in two cities (Nantes and Rennes) ensure their storage and processing.

Fig. 2 details the process performed on computing platforms (different clusters) located in Nantes and Rennes. The medical images and their metadata are uploaded using the SHANOIR neuroimage data sharing platform [8], and these are managed and stored on an imaging platform (Neurinfo) in Rennes. Subsequently, the images are transferred to Nantes. In Nantes, the analysis occurs at two places: the university hospital and on a computing cluster (BiRD) of a research institution. The BiRD platform also enables the analysis of transferred images and processes genetic data. Genetic data is obtained on-site through high-throughput sequencing and array genotyping techniques from biological samples sent by all hospitals. Thus, although the ICAN project involves multiple hospitals, the data analysis process is performed at only two sites.

As discussed below, the project structure described above already poses several technical, legal, and socio-economic challenges, foremost related to data storage, data sharing, and computational requirements.

[1] See for example [54] for a discussion of new challenges arising in the field of next-generation sequencing technologies.

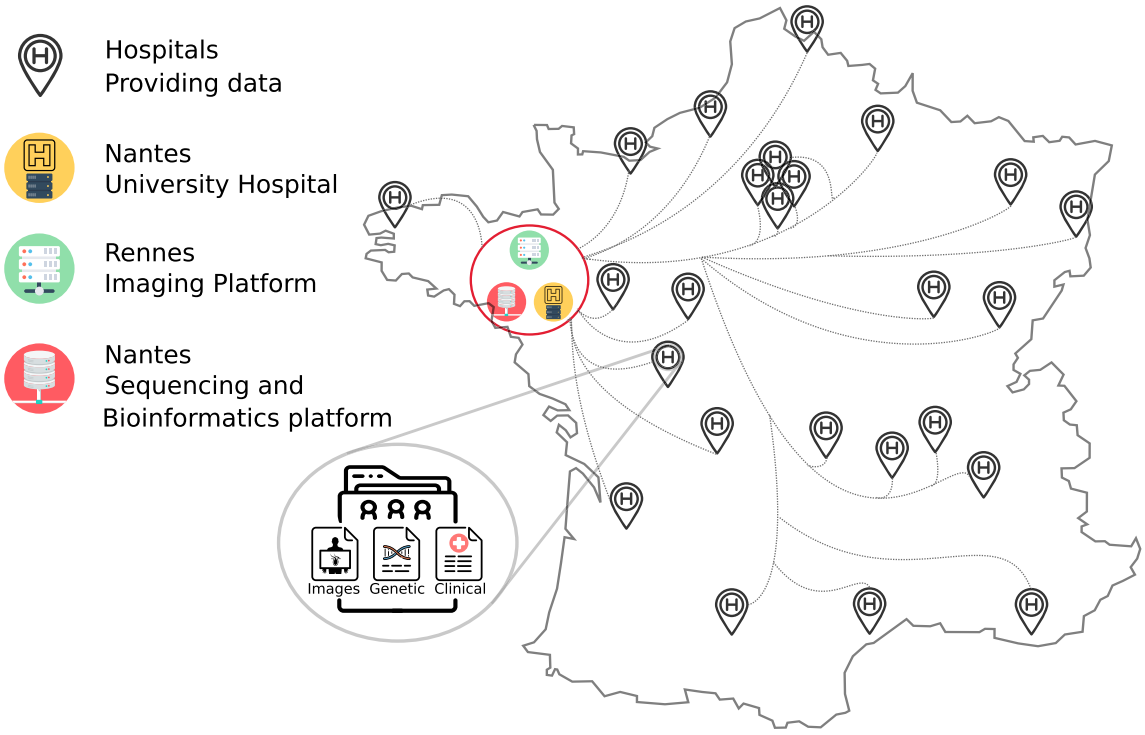


Fig. 1. Hospitals, data flows, and processing sites of the ICAN project.

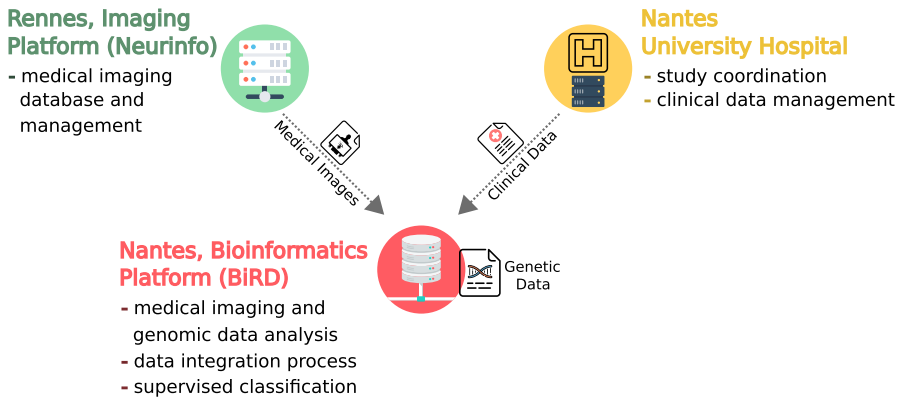


Fig. 2. Data analyses process performed in Nantes and Rennes in the ICAN project.

In addition, replicating the study abroad or sharing data with other countries would require overcoming legal constraints and considering alternatives to centralizing storage and processing.

2.2. Technical challenges

Biomedical cooperations are subject to major technical challenges. The ICAN project requires, for example, huge data volumes to be processed and long-running complex computations to be performed. This is one of the reasons that all data is centralized at two sites. The other sites do not dispose of the necessary hardware and software infrastructures. The data volumes grow at least linearly in relation with the number of involved patients. In this context storage and communication bottlenecks arise easily [95], leading to major questions related to the localization/placement of computation and data, notably for performance reasons.

Second, the ICAN project uses a simple distributed collaboration architecture because data is generated at all sites but then centralized and processed at only two sites out of 34. More widely distributed architectures between the participating sites would have led to larger distributed executions of analyses performing less data movements and thus could have been more efficient and cost-effective. They could, however, not be employed because of insufficient computational facilities at many sites. In general, current computational infrastructures and data storage methods lack support for the efficient distributed processing of massive biomedical data [46].

Third, because of its mostly centralized execution architecture, the ICAN project harnesses a simple architecture in terms of security and privacy checks that relies on basic infrastructure security services, such as authentication. However, in general, biomedical analyses must satisfy security and privacy properties that are much stronger than those applicable to other domains and have to be enforced in heterogeneous (computational and regulatory) environments [26,35,96].

2.3. Legal and socio-economic constraints

Biomedical data and the projects using them are most frequently subject to much stronger legal constraints than other data types. For instance, within the ICAN project French regulations do not allow sharing data (easily) between clinical services and research groups, even when both are part of the same university hospital and even when a researcher is working in both parts. Furthermore, French regulations impose stringent constraints on privacy preservation if data is shared with third parties, *e.g.*, public cloud infrastructures can only be used in exceptional circumstances and after undergoing specific accreditation procedures. Currently, the ICANpartners are considering an extension to an international partnership across and beyond the European Union.

However, for more deeply integrated cooperations, the regulatory situation becomes much more complicated because of the different laws governing data privacy in general and health-related data in different countries. Since governments are aware of the potential benefits of such cooperations, they have made efforts to regulate data privacy without affecting the flexibility of the research initiatives [47]. For example, in the United States, the National Science Foundation (NSF) has defined policies for data sharing, encouraging researches to share data and results [92]. However, these efforts are often not compatible on the international level. Likewise, the European Union has also recognized the importance of open data initiatives with corresponding agreements to support collaboration, notably the reproduction and reuse of experiments located at different sites [123]. These initiatives as well as similar ones in other locations support data sharing; however, national regulations are often limiting these efforts. For example, in the European Union, the General Data Protection Regulation (GDPR) [29] is a standard regulating the sharing of EU data. Similarly, the California Consumer Privacy Act (CCPA) [71] defines privacy guarantees and protection mechanisms for California residents.

Apart from these formal legal constraints, two strong socio-economic motivations also hinder data sharing in biomedical settings, notably large financial and reputational benefits for institutions and researchers. These impediments to collaboration and data sharing, which apply to any sufficiently important discovery, can be mitigated by technical means, notably through strong encapsulation and protection means for data and computations that preserve data confidentiality even during the execution of analyses.

2.4. Multi-site analyses based on FDC scenarios

In order to motivate FDCs, let us consider multi-site analyses that are executed in an international context. Such cooperation is currently in preparation, for example, as part of an extension to the ICAN project. Fig. 3 presents a corresponding analysis workflow: while it does not stem from a real project, it is defined in terms of realistic constraints that apply to international cooperations.

The figure shows a workflow involving four sites, three in France, and one in Colombia, where each site disposes of private data, the computing and storage infrastructures at the different sites are heterogeneous. The left part shows a distributed multi-site workflow composed of nine steps, each represented by a different color. The workflow trains a collaborative learning model by aggregating local models [80]. The analysis is defined as a workflow over computations and data movements. On the right-hand side, the four sites are separated into two groups according to region-specific rules governing aspects of cooperations, such as data-sharing restrictions, policies for data privacy protection, and data ownership. The sites located in France may share data as in the ICAN project. Similarly, sites located in Europe may cooperate under standard rules, such as the GDPR. In this scenario, a collaborative analysis also has to be compatible with the Colombian data protection law. The law allows the transfer of personal data only to countries with defined data protection standards [30], such as members of the European Union. Therefore, the proposed scenario takes advantage of computation facilities at each site while respecting strict regulations on biomedical data sharing in both countries.

In addition, each site has different computing and network access capacities. Sharing data on a central site is also inappropriate due to technical restrictions such as computing capacity and bandwidth limitations. For example, the memory capacity required to train a model is notably affected by the size of the data, and the execution time further increases rapidly with increasing data size [141]. Moreover, transferring raw data versus trained models makes a significant difference in terms of network capacity. 1.7 TB of raw data may lead to a trained (compressed) model of 400 MB [19], a notable reduction. Therefore, FDC scenarios that build models in distributed fashions are often crucial to mitigate technical limitations. The proposed workflow shares local models and partial data to be aggregated collaboratively. This alleviates technical limitations by reducing required memory and network capacities compared to expensive training on all aggregated data at a single site.

In this context, Fig. 3 presents a model aggregation workflow across four sites. Let us assume that S_1 is in charge of aggregation tasks because it has a higher computing capacity than the sites S_2 and S_3. In contrast, S_3 has no computing capacity; therefore, it shares its data with S_2, assuming a regional cooperation agreement. Finally, the different steps respect the data sharing restrictions that apply to the collaboration between S_1, S_2, and S_3 (as shown by the data flows between them). Once the global collaboration with S_4 is started, the technical restrictions have to be met.

The collaborative analysis illustrated in Fig. 3 also addresses socio-economic constraints. For example, the two countries can share models only if the privacy and confidentiality of the data is ensured. A strategy using encapsulation of data reduces information leakage risks during data sharing and processing. Such capabilities of FDC systems also promote trust between the parties, alleviating legal and socio-economic restrictions on biomedical data.

Generally, such workflows have to respect numerous constraints, including data ownership, bandwidth limits, infrastructure heterogeneity, and availability, as well as security risks. An FDC-based workflow should facilitate the automation of such complex workflow under the corresponding restrictions. Support for FDCs requires mechanisms to deal with such constraints, and BATTs should thus provide them. From these considerations, we propose three architectural key features to support FDC scenarios:

- **Data and Computation Placement.** Frequently moving complete datasets between sites is inefficient due to constraints on network capacity and bandwidth costs. Alternatively, data can be processed locally and only partially transferred. Similarly, different sites may share computational resources to optimize computations.
- **Privacy and Security.** Distributed biomedical analyses must ensure strong security and privacy properties, notably confidentiality and integrity properties throughout the complete processes. However, delegating security checks to third parties, *e.g.*, cloud providers, is frequently insufficient because of non-compliance with legal constraints and robust privacy requirements. Third parties provide other mechanisms to address security and confidentiality, such as confidentiality agreements, but these do not grant data privacy and confidentiality when data is moved to third-party facilities.
- **Performance and Scalability.** In multi-site scenarios, computational resources and requirements may differ at each site. Thus, scaling biomedical computations represents a serious challenge. Again, until now, BATT designers have frequently opted to rely on infrastructure scaling capabilities to address the issue, *e.g.*, elastic computing in the cloud.

We consider the issues discussed in this section as fundamental for FDCs. The taxonomy presented in the following investigates how current biomedical tools and techniques address these concerns.

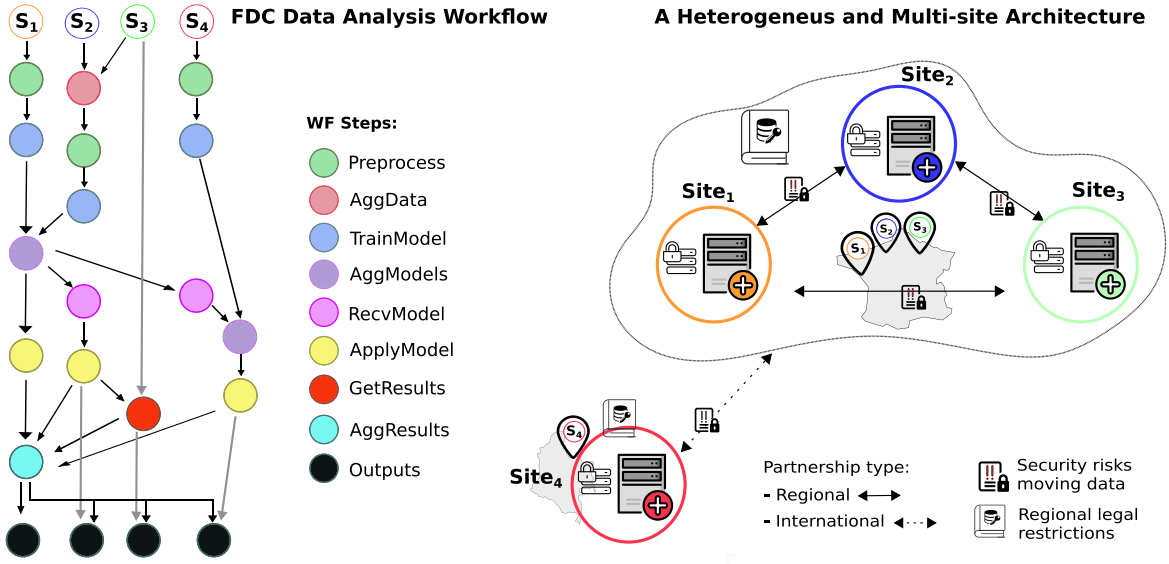


Fig. 3. Scenario for distributed processing analysis in Fully Distributed Collaborations across four sites.

3. Literature review methodology

This systematic review focuses on tools and approaches for biomedical data analyses. Genomics being a representative field of the practices used by biomedical researchers when analyzing vast amounts of data, we have restricted the scope of the search to techniques and tools used in biomedical genomics and cooperative analysis. We are interested in investigating what tools, techniques, and heuristics that are used by researchers to analyze biomedical data in multi-site scenarios. Based on Fully Distributed Collaborations, we defined the following research questions:

1. What techniques, methods and tools are being used by researchers to analyze genomic data, in particular analyses with data resulting from Next-Generation Sequencing (NGS) technologies?
2. How do researchers define workflows of analyses of biomedical data, and what tools are they using to execute them collaboratively? Concretely, we wanted to evaluate three aspects: workflow specification languages, reproducibility of experiments, and interoperability mechanisms.
3. What are the technical features of the current tools and techniques used for scientific collaboration? Moreover, how do they fulfill the key features for fully-distributed collaborations? We were particularly interested in how these tools manage data and computation placement, privacy and security, and scalability and performance.

To address the research questions, we first searched for scientific papers indexed in the following databases: ScienceDirect, PubMed, IEEE, Google Scholar, Scopus, and ACM. The literature search was limited to articles published during the last ten years. We then realized a cross-reference study and applied several filters to define the final set of research work.

The search words used to address the first question were "biomedical applications" and "genomic analysis tools". Based on these results, we selected the papers addressing any of the phases defined in the data analysis protocol for genomics proposed in Ref. [37], namely: de novo genome assembly, sequencing reads mapping and comparison, gene expression analysis, and single nucleotide polymorphisms (SNPs) identification (see subsection 4.1 for a detailed explanation of the protocol). This first search resulted in 87 papers.

For the second question, we used terms such as "biomedical workflows", "distributed workflows", "workflow analyses", and "biomedical workflow analyses". We filtered out the articles that had no link to research in biomedical fields. We then applied a second filter based on the most popular tools among biomedical practitioners. This selection was supported by the number of projects using a particular tool on collaborative research websites such as *myExperiment*,[2] *BioSharing*,[3] and *WorkflowHub*.[4] As a result of this search, we selected ten workflow systems to complement the selection of articles from the first search. To complement these two sets of research works, we also searched for literature reviews addressing tools and techniques for distributed biomedical analyses. The terms used during the search were "distributed biomedical tools", "genomic analysis", and "distributed biomedical analyses". We then filtered out works unrelated to scientific collaborations, data analysis, or biomedical collaborations. We ended up with a set of 53 papers.

Finally, we have performed a cross-reference analysis from these sets of articles, looking for papers being cited or citing the papers in the sets. We have then filtered out the referencing papers using the same criteria applied to the original set. This yielded a set of 40 papers in total. Note that the third research question did not generate a specific search on the databases, it was answered from the analysis of the selected literature.

In total, 50 papers were analyzed and discussed, corresponding to selected applications and relevant workflow systems. The articles are classified in the taxonomy presented in Section 4.

4. A taxonomy of biomedical tools

This section presents a taxonomy of existing Biomedical Analytical Tools and Techniques (BATTs). We investigate and classify current BATTs with respect to three different criteria that are essential to collaborative research:

- First, we study which biomedical problems are being solved with genomic data analysis techniques and what tools support those analyses.
- Then, we investigate how some workflow systems support collaborative research in terms of workflow specification tasks, the ability to reproduce experiments, and interoperability properties.

[2] www.myexperiment.org/workflows.
[3] BioSharing.org.
[4] workflowhub.eu.

- Finally, we classify BATTs according to three architectural features identified in the motivation section: support for data and computation placement, privacy and security properties, as well as scalability and performance properties.

The taxonomy is based on tools and techniques used in data analysis experiments on genomic data due to its importance in biomedical research and due to the accelerated growth in the amount of unprocessed data in recent years. However, our FDC approach is general enough to applicable to other types of biomedical data analyses.

4.1. Biomedical problems, data analytic techniques, and tools

In this section, we classify the data analytic techniques and tools used by biomedical researchers to solve common problems when processing genomic data. Such processes can be generally represented by the workflow shown in Fig. 4. In the figure, high-throughput sequencing data analyses consist of three processing stages.

In the first stage, researchers prepare the tissue samples using biophysical and biochemical methods (for more details on these methods, see Ref. [66]). These samples are then sequenced to create libraries of millions of short sequences, known as reads, that are stored in computer files (see Ref. [37] for a detailed explanation). In the following stage of *data sequencing analysis* the reads are assembled to generate a sequence. If the studied organism is not associated with any known genome, then "de novo" assembly techniques together with *sequence comparison* techniques are used to create the sequence. In contrast, if a reference genome is available, the *sequence reads mapping* techniques are used to generate the complete sequence. Finally, researchers have enough organized data to analyze and test biomedical hypotheses using data analytic techniques in the third stage. For example, researchers may study the relation of specific genes with particular diseases employing *Single Nucleotide Polymorphism (SNP) identification* or *gene expression analysis*.

We are interested in the computational methods used in the second and third stages, that is, techniques used to study biomedical hypotheses based on previously extracted and organized data. In Table 1, we classify the Biomedical Analytical Tools and Techniques (BATTs) according to the five problems addressed by the activities presented in the second and third stages of Fig. 4, namely genome sequencing using "de novo" techniques, sequencing using reads mapping, sequencing by comparison, gene expression analysis, and SNP identification. A similar classification is presented in Ref. [21], which we extend by adding data analysis techniques and the corresponding BATTs used to apply such techniques. In Table 1, we present a specific biomedical problem, what the analytical techniques are used to solve the problem, and then what tools are used to address that particular problem as well as a corresponding research paper.

As discussed below, the BATTs we consider to analyze data using sequential or parallel cluster-based strategies. The applications we review here reveal opportunities to improve processing strategies, especially how data and computations are distributed, notably in the context of FDC scenarios. A detailed explanation of the biomedical techniques is out of the scope of this paper. Interested readers may follow the individual references.

4.2. Support for research collaborations

Nowadays, global multi-party collaborations are crucial in multiple biomedical domains, such as cancer treatment [68] and human genome identification [31]. Such collaborations require large geo-distributed analyses to be applied to huge amounts of distributed data. Furthermore, they should be reproducible to support independent validation by the biomedical community. In order to support such collaborations, BATTs must provide mechanisms and abstractions for the application of algorithms in distributed data and support the participation of independent and partially competing researchers. Therefore, we argue that the appropriate tools must possess at least the following three corresponding characteristics:

- They must provide a *workflow description language*, providing a common interaction language for users with different backgrounds.
- They must provide explicit means to *reproduce* experiments. Thus, the definition must include enough information for other teams to replicate the experiment.
- The tools must provide *interoperability* mechanisms such as Application programming interfaces (APIs).

This section investigates how scientific workflow management systems (SWFMSs), popular tools in biomedical research, meet these research collaboration characteristics.

4.2.1. Workflow description language

The most common way of defining a definition of data-driven biomedical analysis is using a workflow description languages (WDLs). A workflow definition includes defines atomic tasks that perform specific calculations on available data [7,147] and the execution order of (atomic and complex) tasks. The tasks may be executed sequentially or concurrently depending on the WDL and the infrastructure capabilities. The workflow is represented using a directed graph. The nodes represent the tasks, and the edges represent the control flow. Biomedical analyses also require the definition of data flows between tasks. Most data flow definitions are implicit, that is, the atomic task assumes that the data will be ready for processing via the corresponding data sources when the task is invoked. The input data for a specific task may be prepared by the previous task. This is, however, not always the case: in general, the edges represent data flows (or better data dependencies) only implicitly. Finally, some workflow description languages provide a graphical user interface (see Galaxy [52] and Kepler [9]), while others, such as Snakemake [67] and Nextflow [121], use scripting languages.

Table 2 shows the most important workflow systems, classified with respect to the type of graph supported for the definition of the workflows, their user interface, and the specification language used or generated by the tool. The column *Workflow graph* indicates if the tool supports the definition of workflows using Directed acyclic graphs (DAG) or Directed Cyclic Graphs (DCG). The user interface column indicates how the workflow definition is specified via a textual interface or a graphical one to represent the tasks and the control flow. Finally, the *Spec. Language* column indicates the language used for the specification or the language generated by the graphical tool to specify and store the workflow. For example, Snakemake provides a scripting language whose syntax is similar to Python. Taverna [133] supports data flows expressed in terms of the Simple Conceptual Unified Flow Language (SCUFL), an XML-based language. SCUFL provides three main abstractions: processors, data links, and restrictions applied during the workflow execution. The Nextflow [121] specification is based on the Groovy programming language, and it is a Java-syntax-compatible. Finally, Wings [50] uses the Web Ontology Language (OWL) to model tasks and workflow constraints. OWL was proposed by the World Wide Web Consortium (W3C) and is a computational logic-based language for representing and sharing computational ontologies.

4.2.2. Experiment reproducibility

Experiment reproducibility is an important feature in any research field. It is essential for global research collaborations studying biomedical questions. However, designing reproducible experiments with biological data has proven to be challenging. Some researchers even claim that life sciences are experiencing a reproducibility crisis [49]. The non-reproducibility of experiments nowadays may lead to the

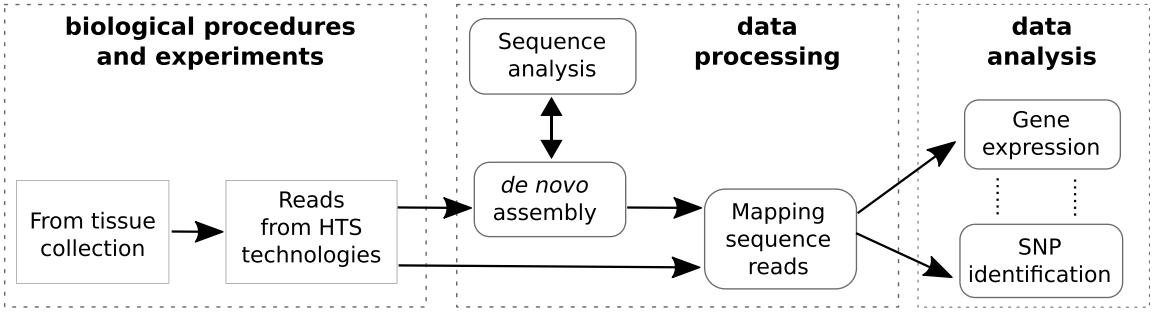


Fig. 4. High-Throughput Sequencing (HTS) data analysis protocol (suggested in Ref. [37]).

Table 1
Characterization of BATTs by analysis techniques applied on genomics data.

Problem	Analytic Technique	BATT
De novo genome assembly	String graphs	CloudBrush [23]
		SA-BR-Spark [42]
	De Bruijn graph	Contrail [110]
		DNA fragment [138]
		Spaler [2]
Sequencing reads mapping	seed-and-extend	CloudBurst [109]
		BlastReduce [108]
		CloudAligner [88]
	Burrows-Wheeler	SparkBWA [3]
		Halvade [38]
		SEAL [100]
	Several mappers	S–MART [152]
		DistMap [94]
		MetaSpark [150]
	Statistical methods	SparkSeq [131]
		MrMC-MinH [101]
Sequence comparison	Local alignment	SparkSW [146]
		CloudSW [140]
		DSA [139]
	Alignment-free	Alfree [151]
		HAFS [22]
		CAFE [82]
		Strand [28]
		FractalMapReduce [5]
	k-mer methods	BioPig [91]
		Nephele [28]
	BLAST alignment	CloudBLAST [5]
		SparkBLAST [20]
		Biodoop [72]
Gene expression analysis	Statistical methods	ExAtlas [113]
		Myrna [69]
		Sparkhit [58]
		SEQSpark [144]
	Gene data sets	YunBe [145]
	Clustering methods	VariantSpark [93]
		Sparkhit
SNP identification	Haplotype blocks	CloudTSS [60]
	Bayesian approaches	SOAPsnp [73]
		Crossbow [70]
	Sequencing methods	Crossbow
		GATK [84]
	p-value test	BlueSNP [57]

Table 2
Workflow description characteristics of the SWFMs.

SWFM	Workflow Graph	User Interface	Spec. Language
Galaxy	cyclic graph	graphic	N/A
Kepler	cyclic graph	graphic	XML-based
Knime	acyclic graph	graphic	N/A
Nextflow	acyclic graph	textual	groovy-based
Pegasus	acyclic graph	textual	XML-based
Swift	cyclic graph	textual	objective-C-based
Taverna	acyclic graph	both	SCUFL-based
Triana	cyclic graph	graphic	XML-based
Snakemake	acyclic graph	textual	python-based
Wings	acyclic graph	graphic	OWL-based

rejection of otherwise sound research papers. For example, in 2007, researchers from the M.D. Anderson Cancer Center[5] found inconsistencies due to data handling errors in an acclaimed paper on cancer treatment from the previous year [61].

Since then, the research community has put a lot of pressure on researchers to ensure that their experiments and results are reproducible [59]. Funding agencies, in particular, have established policies to promote reproducibility, such as NIH guidance for addressing rigor and reproducibility in their funded projects [56]. Similarly, the European Union's Horizon 2020 program has implemented measures to make research data accessible for reproduction by others.

Executing analyses in distributed environments increases difficulties for the reproducibility of experiments; for instance, distributed systems consist of many components working together, and in those systems cause challenging problems such as heterogeneity, scale, and instability [129,130]. In the case of FDC-based analyses, reproducibility is particularly challenging because multiple sites, each potentially with different, *e.g.*, computing capacity, dynamically-changing topologies, security, and privacy parameters. These technical challenges limit the capacity of current tools to support the reproducibility of geo-distributed analyses.

This section investigates the level of reproducibility provided by the workflow systems that have been considered above. We start from two different reproducibility perspectives advocated by various research initiatives, namely, *computational reproducibility* and *experiment reproducibility*. We then will describe the approaches and present a taxonomy to classify and understand the current state of the tools.

Let us start discussing *computational reproducibility*. Here, we focus on three relevant features: data provenance management, support for the export of workflow components for reuse in another systems, and the ability to document or modify analyses phases by means of annotations.

Several papers have evaluated the computational reproducibility in scientific workflows [27,48,53,81,106]. These studies have concluded that for the reproduction of data analysis experiments, researchers need at least three things: the workflow description, the data, and the description of the technical configuration of the experiment. The findings seem natural and sound; however, we argue that the last requirement, *technical configuration of the experiment*, needs some discussion. In particular, it is essential to note that the information of "technical configuration" is required for performance comparisons and for the detection of possible sources of errors. However, the reproduction of the analysis must be independent of the hardware and middleware configuration, as is often only implicit in current work. Workflows, as well as algorithms, describe a computation performed over input data, and are, in principle, independent from the hardware and middleware on which

[5] https://www.mdanderson.org/.

they are executed. Thus, it is important to differentiate the configuration of the underlying hardware and middleware (for example, the operating system), from that of the active components of the analysis, for example, a deep learning engine.

Other researchers have studied reproducibility from a more abstract perspective, empathizing experiment (analysis) reproducibility [39]. These researchers advocate for a more general set of requirements for reproducibility: data provenance, workflow exchange mechanisms, and workflow annotations. Data provenance information documents where data came from and what transformations it has suffered. The exchange mechanisms describe the standards and methods supported to define workflows, and workflow annotations provide information about the execution of the scientific analysis. We think that this last perspective is more suitable for the study of experiment reproducibility over FDCs. In particular, because it addresses the problem from the perspective of knowledge transfer and abstract computations (workflows). In other words, it highlights the design of the scientific analyses and the corresponding algorithmic implementation. Interestingly, these issues are similar to those presented in the software development community, where design documentation and algorithm definition are crucial for reproducibility. For instance, design documentation via annotations in the workflow, metadata to track transformation in scientific data sets, and standard platforms for "code/experiment" sharing have already been studied by the software engineering community [27,39,62]. Industry standards improve communication among researchers and facilitate exchanging knowledge through platforms such as *myExperiment*, *BioSharing*, and *WorkflowHub*.

Table 3 presents the three features thato characterize the support for experiment reproducibility in the SWFMSs. The first column indicates if the system provides some data provenance mechanism. SWFMSs labeled with 'Yes' comply in some way with practices and recommendations, like those proposed by World Wide Web Consortium (W3C) [86], or they provide some proprietary provenance strategy. For example, Taverna represents metadata at the dataflow level based on the PROV[6] language defined by W3C. However, the information stored does not provide details of intermediate data during the workflow analysis [115]. On the other hand, Kepler records all details during the workflow execution, including data evolution and step details. Provenance features also enable researchers to debug transformations, *e.g.*, to search for errors. However, there are still many open challenges regarding data provenance; for example, incomplete data provenance records limit the reproducibility of workflows [64].

The column "WF Exchange" shows the supported formats for exchanging workflow definitions or their components. Pegasus abstractly describes the workflow using a DAX format (Directed Acyclic Graph in XML) up to version 4.0, and the following versions use a serialization format YAML. Galaxy is based on a format that is not readable and a more recent, experimental one. In Taverna, the workflow and its components are represented using the SCUFL language based on the XML format. Kepler exports the workflow components to the Kepler Archive (KAR) file based on the JAR file format from Java. Knime exports to format KNWF; it is a KNIME Workflow data based on an XML/JSON format. Nextflow has its own format that represents the specification of the workflow based on Groovy programming language. Swift, through the Swift language, represents the workflows in a format based on C-like syntax. Triana defines the workflows using files based on XML representing the name, specifications, and parameters. Snakemake uses Snakefiles, Python-based rules including input and output data between them. Wings expresses the workflow components and dependencies in workflow templates using the Resource Description Framework (RDF) and the Semantic Web Rule Language (SWRL).

Table 3
Computational reproducibility characteristics in the SWFMs.

SWFM	Data Provenance	WF Exchange	Annotations
Galaxy	Yes	GA(v19), Format2 (>v19)	Yes
Kepler	Yes	KAR	Yes
Knime	No	KNWF	No
Nextflow	No	NF	No
Pegasus	No	DAX(v4.0) and YAML(v5.0)	No
Swift	Yes	Swift	Yes
Taverna	Yes	SCUFL	Yes
Triana	No	XML	No
Snakemake	No	Snakefiles	No
Wings	Yes	RDF/SWRL	Yes

[6] See https://www.w3.org/TR/prov-overview/for information on PROV.

Finally, the table shows if a platform supports annotations in the workflow. Annotations help scientists to understand the design of the experiment. For example, Taverna and Galaxy provide means to annotate the workflow using free text or labels, in contrast to Nextflow, which does not offer an annotation mechanism.

The table shows that support for experiment reproducibility is still limited in the current workflow systems. In fact, in 2017, Kanwal et al. studied reproducibility and provenance tracking in workflows for genomic analyses and pointed out that there is still an incomplete understanding of reproducibility requirements, and thus a lack of support on current tools. They also remarked the complexity of reproducibility of distributed workflows as an open research problem. Similarly, other researchers have identified *workflow decay* [36,147] due to volatile third-party resources, missing example data, missing information about execution environments, and insufficient descriptions of workflows as a complication for experiment reproducibility.

4.2.3. Workflow System's interoperability

We define interoperability as the ability of workflow systems to communicate with other systems to exchange data, share functionality, or delegate responsibility. SWFMSs may interoperate by providing language bindings or by exposing an API at runtime. Programming language bindings allow scientists to use their preferred programming languages to create complex analytical routines to extend, enhance, or leverage native functionality. On the other hand, an exposed API helps scientists to interconnect several analytical engines to delegate responsibilities to specialized hardware and software. For example, a scientific workflow may execute most of its computations in local infrastructure and delegate a specialized machine learning computation to a GPU cluster deployed in the cloud.

In this section, we investigate interoperability mechanisms by identifying each tool's different programming language bindings and by studying the presence and reach of an API exposed to be invoked at runtime. In particular, we have studied the presence of a REST API (API based on REST-defined services) and what functionality is exposed. For example, some tools provide APIs to support execution monitoring, while others allow full control of the workflow life cycle (definition, debugging, deployment, and execution). During the discussion, we use the word API to refer mostly to the programming interface provided via libraries for specific languages (language bindings) and REST API to refer to the functionality exposed at runtime using REST web services.

Table 4 presents the classification of SWFMSs with respect to their interoperability mechanisms. The second column, *Language Binding*, enumerates the programming language that is supported by each tool. A system with more language bindings is supposed to provide better interoperability properties. For example, Galaxy provides Python, PHP, Java, and JavaScript libraries, allowing researchers to define their analytical experiments in their preferred language. Knime, on the other hand, supports the inclusion of Python scripts as custom code in the workflow steps defined in their graphical user interface but does not provide language bindings to use the tool from a programming language. Similarly, Taverna and Triana enable customizing Java code through the graphical interface during the workflow design. In contrast,

Table 4
Interoperability properties in the SWFMs.

SWFM	Language Binding	REST API capabilities
Galaxy	Python, PHP, Java, JavaScript	Low-level: interact with data, run tools and WFs, handling histories
Kepler	Java, R	No
Knime	Python scripts via UI	Interaction by UI: ETL to visualization tools, and score a model for predict analysis
Nextflow	No	No
Pegasus	Python, Java, R	Monitoring, defining and executing workflows.
Swift	No	No
Taverna	Java scripts via UI	Monitoring and executing workflows
Triana	Java scripts via UI	No
Snakemake	Python	No
Wings	Java, Matlab, and Python scripts via UI	No

Nextflow and Swift do not provide libraries to extend to programming languages, and they use their proprietary specification language to define workflows.

The third column, *REST API*, provides information on interoperability with external sources through REST services. Galaxy's API supports the interaction with external programs or libraries at runtime. The Galaxy API allows interaction with data sets, executing and monitoring workflows, and monitoring relevant information. Secure communication is achieved by supporting the HTTPS protocol. Galaxy is migrating the current version of the API to an improved version supporting the standard of FastAPI.[7] Knime provides interoperability mechanisms, although limitedly, through the licensed component Knime Server. It enables calling web services as part of the workflow design by means of a graphical interface. Knime performs ETL (Extract, Transform, Load) operations via REST services and integrates with external visualization tools. Finally, Pegasus and Taverna provide functionalities through REST web services to support monitoring, design, and specification of workflows.

The table shows that interoperability mechanisms are limited in current workflow systems. Other researchers have identified these shortcomings; see the research presented in Refs. [1,44,114]. Sarah et al. [27] also identified interoperability between workflow systems as an open challenge, suggesting a standard intermediate model between the specification and execution layers to interoperate between different workflow systems.

4.3. Distributed architectural features

We now investigate the support of BATTs for the architectural features required to address fully distributed collaborations, as introduced in section 2.4. Concretely, we will investigate how the current tools address *data and computation placement*, *privacy and security*, and *performance and scalability*.

4.3.1. Data and computation placement

Analyzing large amounts of data scattered over several sites requires transferring the data to a single site for analysis, delegating computations to the sites where data is stored, or a hybrid solution that partially distributes storage and computations. The choice of the strategy to use depends entirely on the specific context of the collaboration.

The problem of allocating storage, bandwidth, and computational resources to optimize the performance of a specific scientific workflow has been amply studied generally for cluster and grid computing settings. A common strategy is to address the problem of data placement in order to minimize total data transfer cost and optimize the execution time (computation time) in a given architecture, as proposed, for instance, by Van Huang and Chuanhe [125] as well as by Cope et al. [33]. In contrast, Li et al. [74] claim that such cost minimization strategies are insufficient and propose a two-stage approach, first pre-allocating datasets to specific data centers during workflow build-time, and then dynamically distributing newly generated datasets at runtime. Other authors have proposed other heuristics to address the placement problem [24,43,136,148]. However, finding the *optimal* data placement strategy is an NP-hard problem [79]. Studies on computation placement are rare: very few approaches address the problem of selecting the best computation facility for a given task, and most rely on cloud environments guaranteeing the availability of (almost) arbitrary computational resources required by collaborative scenarios. Similarly, studies relying on MapReduce and workflow systems lack the functionality necessary for multi-site processing [41,98].

Therefore, current approaches are not appropriate for FDCs for biomedical analyses for two reasons:

- They do not support geo-distributed architectures that constitute a much more complex setting for computation and data placement [77,78,98,99].
- They do not consider the larger set of constraints of biomedical applications that involve legal and socio-economic constraints.

This problem is manifest because most BATTs harness partitioning data simply by distributing equal chunks among multiple nodes. Data is often loaded onto a single node and then split into chunks stored and distributed on a cluster of nodes using randomization strategies [135]. Table 5 starts by classifying the BATTs according to the paradigm and data location type offered. For example, Halvade or BioPig use such strategies based on Hadoop. Others use different frameworks, notably Spark, but use similar data and computation placement strategies. Sparkhit [58] can process data from different clusters located in three geographic regions, but the data needs to be moved to one place. MetaSpark [150], SparkBlast [20], VariantSpark [93], and SparkSW [146] are also based on Spark architecture. By contrast, a few BATTs, like S-MART [152] or CAFE [82], harness sequential analysis without any distributed processing. Therefore, all these BATTs lack the functionality to support fully distributed and collaborative work, especially to process data across different geographic sites.

Other tools as workflow systems have more explicit data and computation placement features. For instance, Pegasus [40] supports three data placement approaches, shared file systems, remote ones and non-shared ones. In addition, Pegasus uses DAGMan [117] and HTCondor [120] to model task-based workflows submitted to a pool of resources in HPC clusters. Kepler [9] includes prebuilt components (referred to as actors) to model external data sources and grid facilities. Finally, Taverna uses the SCUFL language to explicitly define the data flow between processors to model the passing of data between services associated with bioinformatic atomic tasks. Pegasus and Triana implement the strategy replica location service (RLS) [143] that allows access to information about copies in different physical locations to support scalability, reliability, and security during distributed executions. The same approach is used by the Giggle (GIGa-scale Global Location Engine)framework [25].

The second column of Table 5 shows the classification of BATTs according to their data location and computation strategies. The data placement column indicates if the corresponding BATT supports local data processing (label loc). If, for example, the local data placement strategy may distribute data in a cluster, such as in Hadoop, we add a plus symbol (+). If the tool supports even more sophisticated methods, such as dynamic allocation offered by Spark, we add a doubled plus symbol (label ++). Finally, the label ext shows if the BATT can handle external data sources located, for example, in cloud repositories like Amazon's S3 cloud. The column *Placement - Computing* describes how computing resources are allocated. If the tool allocates computing resources at definition or configuration time, we classify it as employing

[7] https://fastapi.tiangolo.com/.

Table 5
A categorization of BATTs and SWFMs according to our architectural features proposed.

Name	Paradigm	D/C Placement		Privacy and Security	Perform & Scalability	
		Data	Computation		Architecture	Abstraction
Taverna	Workflow	loc+, ext	dynamic	+	distribut	explicit
Galaxy	Workflow	loc+, ext	dynamic	+	distribut	explicit
Kepler	Workflow	loc++	dynamic	++	distribut	explicit
Knime	Workflow	local	static	delegated	distribut	explicit
Nextflow	Workflow	loc+, ext	dynamic	delegated	distribut	delegated
Pegasus	Workflow	loc+, ext	static	++	distribut	delegated
Swift	Workflow	loc++, ext	dynamic	delegated	distribut	delegated
Triana	Workflow	loc++	dynamic	delegated	distribut	delegated
Snakemake	Workflow	loc+, ext	dynamic	delegated	distribut	delegated
Wings	Workflow	loc+	static	delegated	distribut	minimal
Sparkhit	MapReduce	loc++	dynamic	delegated	PCluster	explicit
Crossbow	MapReduce	loc+	static	delegated	PCluster	delegated
MetaSpark	MapReduce	loc++	static	delegated	PCluster	explicit
CloudBurst	MapReduce	loc+	static	delegated	PCluster	explicit
Halvade	MapReduce	loc+	static	delegated	PCluster	delegated
DistMap	MapReduce	loc+	static	delegated	PCluster	delegated
Myrna	MapReduce	loc+	static	delegated	PCluster	delegated
SparkBLAST	MapReduce	loc++	static	delegated	PCluster	delegated
K-mulus	MapReduce	loc	static	delegated	PCluster	delegated
CloudSW	MapReduce	loc++	static	delegated	PCluster	explicit
SOAPsnp	Sequential	loc	local	N/A	standalone	N/A
S–MART	Sequential	loc	local	N/A	standalone	N/A
CAFE	Sequential	loc	local	N/A	standalone	N/A

static allocation. If the allocation or reallocation of resources is done at execution time, we classify the tool as having dynamic allocation. The dynamic and static labels are related to each workflow system's workflow scheduling strategy. For example, Swift, Galaxy, and Triana systems exploit different dynamic scheduling strategies based on publish/subscribe patterns or adaptive methods during workflow execution [76]. In contrast, systems like Pegasus provide mechanisms resulting in static scheduling [17], without automated strategies to adapt computing resources during execution across multiple nodes.

Finally, even though FDCs are not dependent on any particular hardware or software solution. It is important to discuss its use in the context of fog and edge infrastructures to address some of the problems of international collaborations. For example, fog and edge-based solutions have been proposed to address the problem of computation and data placement in collaborations [4,18,128]. These approaches have the advantage of keeping the data close to the original owner using hardware, such as smartphones or edge servers, to perform some calculations. However, these technical solutions address only distribution and location, omitting legal and socioeconomic restrictions on biomedical data. Furthermore, several of today's research efforts involving final users require the patient to agree to share their medical records with a research institution. The devices are used as sensors to capture the information (see, for example, the app-based study to identify cardiac arrhythmias [122]). Therefore, these efforts will benefit from tools and techniques that support FDC requirements because they can enable new forms of collaboration. For example, a patient participating in a global research effort without disclosing any sensitive information and even performing fog-based computations without risking data leakage among participant devices.

4.3.2. *Privacy and security*

Distributed biomedical analyses are subject to many security risks and, frequently, to a much higher risk of privacy issues than other domains since the potential loss of personal and sensitive information implies more severe consequences. Security and privacy-related properties that have to be satisfied comprise, but are not limited to, authentication, authorization, integrity during access and control of biomedical data. BATTs, therefore, have to provide means for the stringent enforcement of security and privacy-preservation properties, or at least be able to harness corresponding means that are provided by their environments.

In Table 5 (on page §), we classify BATTs according to the security mechanisms provided or delegated (either in the cloud or locally). If a BATT provides simple security mechanisms, we mark it with a single plus symbol (+). Taverna, for example, provides simple security mechanisms, such as authentication based on web services. Similarly, Galaxy offers limited capabilities such as libraries to make a secure connection through a web API.

If the tool provides a more advanced security mechanism, we mark it with two-plus symbols (++). For example, some BATTs provide explicit mechanisms to secure data during storage, transfer, and processing. For instance, in Kepler, the Security Analysis Package (SAP) provides information security mechanisms such as input validation, data integrity, and remote access validation. The package triggers an alarm when a potential alteration of the information is detected [65]. However, some researchers have pointed out that the SAP package is not yet part of the current release of Kepler due to its runtime overhead [104]. Similarly, in Pegasus, the Scientific Workflow Integrity Project[8] (SWIP), proposed by NSF, seeks to ensure integrity and security during workflow execution. The SWIP project implements some cryptographic mechanisms to check provenance metadata to detect input and output data changes during the workflow execution.

Many BATTs delegate security to the hosting (cloud or local) infrastructure provider; this is indicated in the table by delegated. For example, Snakemake, Knime, Sparkhit, and MetaSpark do not offer mechanisms to analyze data securely. The security aspects are delegated to the infrastructure provider. In contrast, Taverna and Galaxy endow the API with authentication strategies through REST services, although the premise is that the processing nodes are part of trusted environments.

Overall, the table shows that only a few BATTs provide specialized advanced security and privacy mechanisms. Moreover, security and privacy issues have also to be handled at other levels, such as storage, sharing, processing, and publication of results. The first BATTs were designed to work with distributed file systems, such as GPFS [111] or PVFS [103], and had to provide simple interfaces to high-performance environments implemented on top of grids [75]. In such scenarios,

[8] https://cacr.iu.edu/projects/swip/index.html.

information security and privacy were often ensured by not sharing data and strongly limiting access to research facilities. Later on, the exponential growth of biomedical data and the emergence of accessible cloud infrastructures motivated BATT systems to redesign their architecture. However, most of them tried to achieve this by minimal changes to adopt the cloud as a preferred execution environment and delegating security to the cloud providers. For example, tools implementing a MapReduce strategy, such as CloudBurst or MetaSpark, delegate security features to the infrastructure provider. In addition, privacy and security are major concerns of its own in cloud environments. There are still many open challenges to comprehensive security during FDC analyzes involving shared computing facilities and data storage. Some initiatives, though limited, address security with a higher priority, such as the SWIP project. However, these challenges are most frequently present because of a lack of support for security concerns in most workflow systems [62,87].

4.3.3. Architecture and quality attributes

Flexible and robust architectures are needed to support FDC scenarios. In this section, we study the architectural features of BATTs, focusing on two distinctive characteristics, namely the support for distributed architectures and the mechanisms provided to address complex quality attributes such as performance and scalability. Concretely, we classify them first according to their execution architecture, identifying those that support stand-alone execution, distributed execution on clusters of computers, and distribution over several geographically separated sites.

We then study how those tools support complex quality attributes, identifying if they provide explicit mechanisms to address them or if they delegate the final configuration to the deployment phase, where engineers rely on the capabilities of the underlying infrastructure. Here, we are interested in the explicit support for distribution from within the tool. For example, as part of a complex workflow, a complex algorithm may be configured at deployment time to be executed in high-performance computing services on Amazon Web Services (AWS). However, we are interested in tools providing explicit means to model and manipulate such high-performance computation facilities.

Table 5 presents our findings. First, the column Architecture shows if the tools are stand-alone, run on a parallel cluster, or support complex distributed architectures. By complex distributed architectures, we mean complex workflows deployed on top of the complex geo-distributed infrastructure. The next column shows if complex quality attributes are supported explicitly by abstractions provided by the tool or if they are delegated to the deployment phase and the underlying computing infrastructure. The remainder of this section provides additional insights on this classification.

4.3.3.1. Stand-alone solutions on a single machine. Stand-alone applications are used to process small datasets on a single machine. For example, CAFE implements alignment-free sequence analysis on single-machine architecture. Decades ago, DNA sequences were aligned using algorithms based on dynamic programming, but this strategy turned out to be inefficient over time due to the length of the investigated sequences. More generally, due to a large amount of data to be analyzed in many cases, stand-alone applications are rarely used nowadays. Therefore, implementations based on parallel approaches such as MPI techniques, GPU computing, or MapReduce frameworks are preferred nowadays. Similarly, other biomedical applications have migrated to parallelization techniques, for example, short reads assembly or SNP identification [57,88,108]. However, tools supporting such architectures do not promote distributed cooperations. Table 1 is complemented by Table 6, where we present the main processing architecture differentiating the BATTs that use sequential computations and those that support processing on parallel clusters.

Table 6
Sequential and cluster-based BATTs for genomic data (complements Table 1).

Architecture	BATTs
Stand-alone	SOAPsnp, S–MART, Alfree, CAFE, ExAtlas
Cluster-based	CloudBrush, SparkSeq, BioPig, VariantSpark, CloudTSS, BlueSNP, SA-BR-Spark, Contrail, Spaler, CloudBurst, BlastReduce, CloudAligner, SparkBWA, Halvade, SEAL, DistMap, MetaSpark, MrMC-MinH, SA-BR-Spark, SparkSW, CloudSW, DSA, HAFS, Strand, FractalMapReduce, Nephele, CloudBLAST, SparkBLAST, Biodoop, Myrna, Sparkhit, SEQSpark, Crossbow, GATK

4.3.3.2. Cluster-based distribution. Most current BATTs are implemented on top of popular frameworks such as Hadoop and Spark. These tools are extensively used for cluster-based processing of large data sets. For instance, Halvade executes pipelines parallelly on a multi-node architecture or in a multi-core configuration using Spark on a cluster and storing data in files using the Hadoop Distributed File System (HDFS). Similarly, DistMap and Myrna use Hadoop to execute statistical models on multiple processors in cluster scenarios. Several other tools follow similar approaches providing parallel execution over a computing cluster, including CloudTSS, SEQSpark, VariantSpark, CloudBLAST, CloudSW, or CloudBurst.

The MapReduce paradigm has also been integrated into workflow systems. For example, Kepler over Hadoop provides an architecture that supports the execution of MapReduce applications during the workflow execution in Kepler [127]. Similarly, Hi-WAY executes scientific workflows using Hadoop YARN [16]. Other SWFMSs employ naive ad hoc strategies to parallelize tasks and distribute data over available resources [17]. These strategies are specific for each system and are usually provided by the task management layer of each workflow system. The corresponding BATTs that we have surveyed are categorized in Table 6 as *cluster-based architecture*.

4.3.3.3. Distribution of computations over several sites. One may argue that some of the tools harnessing parallel infrastructures can support fully distributed collaborations by exploiting, in addition, direct manipulation of the underlying distributed infrastructures such as those based on MapReduce. However, we have identified several limitations as part of our analysis. Most importantly, the tools rely on the configuration at deployment time to address complex collaborations, resulting in a complex deployment phase that restricts flexibility and reproducibility. For example, Taverna provides components supporting remote execution of locally defined workflows that are published and controlled by web services. Taverna can be executed on clusters, grids, and clouds, and it can be made to interoperate with other workflows like Galaxy. However, those configurations require extensive technical knowledge by the deployer and flexible features from the underlying infrastructure.

Kepler supports the use of programs written in R or C for remote execution, harnessing distributed execution threads via web and grid services. In the same way, Nextflow [121] and Snakemake [67] support GRID platforms, *e.g.*, SGE (Sun Grid Engine) or LSF (Load Sharing Facility). In addition, Snakemake supports the interaction with other tools via web services executing jobs in distributed environments, such as clusters or batch systems. But again, most of these tools rely on the configuration knowledge of the deployer. Pegasus has taken a more explicit approach for distribution and supports execution on individual machines, remote clusters, distributed infrastructures, and clouds. Nevertheless, Pegasus does support such architectures with explicit abstractions or components. For example, it incorporates HTCondor to enable the management of resources in dedicated or distributed computers. Also, Pegasus incorporates the Glideins component that allows adding machines from different domains and HPC centers.

Researchers have noted that BATTs must adapt their architectures for processing in multi-site scenarios, such as multi-cloud technologies. Nevertheless, it is necessary to have an orchestrator between the workflow layers and the cloud architectures. This way, the technical

challenges of distributed computing, such as resource allocation, virtualized systems, fault tolerance, and task monitoring, can be supported [112,149].

The first column refers to a platform supported based on MapReduce or workflow system. Data placement covers three strategies: only local data, allocation by a simple partition strategy (+), and dynamic allocation of data (++). Next, computing placement allocates resources to process data before execution (static) or reallocate during execution (dynamic). Privacy and Security cover three mechanisms: a simple (delegated in the cloud or locally), complement delegate security with own functionalities (+), and advanced strategies (++). Finally, the architecture implemented and the abstraction level supported by each one, explicit or delegated.

4.4. Distributed workflow systems

The first column represents the workflow scheduling architecture: central workflow (centralized) or supported by multiple schedulers (distributed). Data management indicates the strategies during data processing, moving it to a centralized site, mediated by a distributed data management system, and transferring data point-to-point in a P2P fashion. The partitioning process generates workflow fragments where each is defined to be executed on a specific site or time. The last column, Distribution Level distinguishes between three possible options: Limited, Partial, or Fully Distributed.

In this section, we study the use of workflow systems for distributed collaborative scenarios. Table 7 presents additional information about SWFMSs, classifying them by workflow scheduling approach, data management strategy, support of workflow partitioning, and fully distributed collaboration.

4.4.1. Workflow scheduling strategies

Workflow scheduling strategies allocate tasks to computational resources such as processing nodes during workflow execution [76]. Yu and Buyya [142] present two scheduling strategies for workflow systems. In contrast to centralized scheduling architectures, decentralized ones enable multiple schedulers to manage tasks. They highlight the importance of scheduling schemes to achieve scalability and performance in workflow systems. They survey Galaxy, Nextflow, Swift, and Triana as examples of decentralized architectures. For instance, Galaxy implements the GridWay framework to provide multiple schedulers, and Swift integrates the Karajan workflow engine [132]. In Ref. [76], centralized architectures are identified as having bottlenecks in the master node, and peer-to-peer (P2P) systems are proposed to mitigate this problem (see Ref. [45] for the implementation of a dynamic workflow management strategy for processes as services over the internet). The first column in Table 7 shows the systems' workflow scheduling strategies based on our assessment of the review.

Table 7
Classification of SWFMs according to the level of distribution offered by each one.

SWFM	WF Scheduling	Data Management	WF Partitioning	Distribution Level
Taverna	centralized	centralized	Yes	Partial
Galaxy	centralized	centralized	Yes	Partial
Kepler	centralized	all	N/A	Partial
Knime	centralized	centralized	Yes	Limited
Nextflow	centralized	centralized	Yes	Partial
Pegasus	centralized	mediated	Yes	Limited
Swift	decentralized	all	N/A	Fully
Triana	decentralized	peer-to-peer	N/A	Fully
Snakemake	centralized	centralized	Yes	Partial
Wings	centralized	mediated	No	Partial

4.4.2. Data management

Another critical feature for FDC support is the data management strategy. Yu et al. [142] study three mechanisms: centralized, mediated, and P2P-based data management. In a centralized data management strategy, a master node manages the data; mediated strategy, the underlying system is responsible for data management; a P2P-based scenario, data is distributed among all available nodes without an intermediary. Some SWFMSs offer several strategies, for example, Kepler. Triana implements a decentralized architecture applying two distribution policies for parallel and pipeline execution [118]. Finally, others propose strategies to search and discover services for data but leave aside the privacy and socio-technical restrictions on biomedical data [63]. The data management strategies are summarized in the second column of Table 7.

4.4.3. Workflow partitioning

Clustering is a technique using several tasks to partition a workflow horizontally. This technique may improve the performance in distributed scenarios. For instance, Pegasus [40] implements different clustering techniques that improve the execution time significantly for short tasks. Following Liu et al. [76], the partitioning process generates workflow fragments where each one is programmed to be executed on a specific site.

The taxonomy presented in this section concludes with the classification of the workflow systems with respect to their support for fully-distributed collaborations. As discussed during the section, existing tools and approaches for distributed biomedical analysis limit functionality that support the three architectural features for analyzing data in FDC scenarios: data and computation placement, privacy and security, and performance and scalability. Additionally, the functionality provided by workflow systems is also limited around interoperability and reproducibility. Therefore, we identified an opportunity in designing systems to analyze data in multi-site environments, such as FDC scenarios, due to the limited support of current ones in collaborative research and geo-distributed processing.

As discussed in this section, existing tools and approaches for distributed biomedical analysis limit the functionality of the three main features for analyzing data in FDC scenarios: data and computation placement, privacy and security, as well as performance and scalability. Additionally, the functionality provided by workflow systems is also limited with respect to interoperability and reproducibility. The last column in Table 7 classifies current workflow systems in three categories according to the functionality provided by each one focused on FDC scenarios: limited, partial, and fully distributed.

To conclude, we have identified opportunities for designing systems to analyze data in multi-site environments, notably using FDC scenarios, opportunities arising from the limited support of current BATTs for collaborative research and geo-distributed processing.

5. Lessons learned and open challenges

Most of the analytical tools and techniques reviewed provide only mechanisms for partial collaboration among geo-distributed sites. Most are based on collaborative strategies where one of the sites acts as a central master and the others are slaves nodes. There are still numerous open issues, including explicit data and computation placement management, dynamic scheduling of resources, robust security and privacy features, and flexible data placement strategies. Additionally, ethical and legal constraints must be considered during the execution of biomedical analyses [34,83]. This section explores these open issues in detail.

5.1. Data and computation placement

To address the problem of data and computation placement, tools must place data and computations in the optimal location according to

specific criteria, *e.g.*, putting data in the infrastructure best suited to analyze it or moving data while preserving data locality and security constraints. This problem is NP-hard in computational complexity, and it is still actively studied [79].

Developing heuristics with efficient implementations for data and computation placement is an ongoing endeavcor. As we have shown before, current BATTs offer only some basic strategies such as data analysis in a centralized setting, uniform data partition among multiple nodes, and computation based on cluster distribution on homogeneous nodes. Therefore, other research directions provide relevant abstractions to handle multi-site biomedical data analysis.

A recent and exciting field of research is the application of machine learning techniques to optimize distributed computations on scattered data that cannot be shared. The application of learning models on these data without tmoving them to a central site, on the contrary, consider sharing local model parameters or trained local models without revealing sensitive data [80]. A prime example, federated machine learning [85,137], has been applied to analyze large genomic data sets spread over multiple sites worldwide. However, there are many challenges, such as communication latency, systems heterogeneity, heterogeneous data, privacy preservation. Nowadays, massively distributed machine learning [97,126] seems a promising future research domain.

5.2. Privacy and security

Research collaborations must comply with strict security policies imposed by governments over biomedical data. These policies may be enforced in a national context (as those implemented by national governments), supranational (*e.g.*, policies enforced by the European Union), or even globally. However, as we have shown before, the current set of tools and techniques used to analyze biomedical data address the problem of security and privacy only partially. They either delegate security to the computing infrastructure (*e.g.*,a cloud provider) or provide very basic mechanisms. Therefore, biomedical researchers must address security by means external to the tools they use. In order to support FDCs, BATTs have to better support privacy concerns. Thus, several researchers are investigating more sophisticated security mechanisms. For example, secure containers encapsulating confidential data allow data to be used in complex computations without exposing participating stakeholders [12]. Other approaches investigate data watermarking techniques augmented with encryption and cryptography operations [14]. Other approaches propose privacy-preserving strategies in workflow scheduling for geographically distributed processing through partitioning strategies for Wide Area Network (WAN) [134]. However, the security in BATTs is still a great challenge; in workflow systems is an unexplored matter, especially by security and privacy risks encompassing large parts of complex workflows, as mentioned by the authors from the NSF's SWIP project.

5.3. Scalability and performance

The exponential growth of biomedical data has forced scientists to scale their experiments and applications. Several techniques have been used to address the problem of scaling up these analytical experiments. A common practice consists in implementing dedicated high-performance computing systems (HPC). Most of the world's supercomputers belong to this category. Another common technique to improve performance and scalability is deploying scientific experiments on clusters of computers, on on-premise facilities, or in the cloud. Some authors propose to mimic the architectures for massive computations and data storage proposed by internet corporations such as Amazon, Facebook, or Google to create data-intensive scalable computing (DISC) facilities for scientific applications [11,124].

Each strategy has its benefits and drawbacks. For example, high-performance computing (HPC) systems offer centralized computing resources focused on high-performance applications with many floating-point operations. However, it requires all the data to be available for the computation, and it also requires a significant investment in infrastructure. These infrastructures may also be costly to maintain, and they can pose problems regarding data ownership and confidentiality. On the other hand, private and hybrid clouds for the deployment of scientific experiments constitute flexible means to configure multiple computing scenarios. However, data ownership, security, and legal restrictions hinder the implementation of massive biomedical experiments on the cloud. On the other hand, DISC systems, *e.g.*, based on Hadoop and Spark, enable processing large volumes of data and the distribution and scaling of resources in clusters or clouds. They include fault-tolerance mechanisms and are generally controlled by the owner of the DISC system. However, these infrastructures may be costly to maintain and pose data ownership, confidentiality, and data localization problems.

None of the above-discussed systems will support Fully Distributed Collaborations as presented in section 4.3. Instead, they all propose some essentially centralized solution. They lack the means to address the three FDC requirements: decentralized computations and flexible data placement, compliance with security and data protection policies, and performance and scalability based on the interests of participating sites. Thus, numerous research opportunities exist to develop solutions addressing these requirements while providing performance and scalability benefits.

We found some computational approaches that may support basic, Fully Distributed patterns; for example, hybrid strategies, exploiting the MapReduce benefits in the cloud [23,58,140,146]. Similarly, some workflow systems also have adopted combinations of these paradigms [76,119,133]. However, as seen in the review, these solutions address some primitive form of technical flexibility, and few provide explicit abstractions to handle all the desired requirements. Our taxonomy also revealed another opportunity, for example, to strive to implement flexibly configurable systems that enable high scalability. The current methods to address high scalability and performance require some form of centralized control.

5.4. Need for fully distributed collaborations

This section has identified several research opportunities in data and computation placement, data privacy and security, and performance and scalability. These areas of investigation make sense for fully distributed collaborations only when they are considered together. In the discussion, we explain how to achieve fully distributed collaborations.

First, the CAP theorem [51] applied in the same logic to distributed systems indicates that it is impossible simultaneously to have the three properties in a distributed system: consistency, availability, and partition tolerance. Therefore, a trade-off has to be enacted between the three properties to achieve a practical system supporting a distributed database in an FDC system. Similarly, the scalability trilemma [10] states that highly distributed systems cannot simultaneously combine: decentralization, scalability, and security. Therefore, again, there should be a balance between the levels of support for each of these three properties. The classification presented in Tables 5 and 7 categorizes the tools and approaches into properties such as resource location, privacy and security, and scalability. It is clear that many covers two of the three properties; therefore, there is an open question about solving them together. There is still a great challenge to achieve all three simultaneously because researchers seek to compensate for two of the properties indicated in the CAP theorem or the scalability trilemma.

Nevertheless, not only technical constraints will affect the development of FDCs. Legal and socio-economic misconceptions should be overcome before having a full implementation. We expect that in FDCs a scientists won't have to disregard security or legal restriction to achieve high scalability and performance. On the contrary, an FDC solution will coordinate dynamically the execution of complete workflows exploiting and combining heterogeneous computing facilities.

6. Conclusions

The exponential growth of biomedical data generated worldwide presents an excellent opportunity for researchers to design studies and experiments, such as genomics data. However, such an opportunity is difficult to exploit, especially in the biomedical domain, due to legal constraints imposed by governments, socio-economic factors involving public and private organizations, as well as technical constraints, such as limited bandwidth, limited storage capacity, and limited computational power. Most organizations and researchers often cannot pursue large-scale experiments on their own because obtaining data from other sources is frequently made difficult due to privacy and security issues, and available resources for computation and data storage may be insufficient at a given site.

As a solution to this problem, we have advocated Fully Distributed Collaboration (FDC) in this article. FDCs constitute collaborative research endeavors where the confidentiality and ownership of biomedical data are satisfied while providing means to analyze and exploit the information collaboratively, by sharing data or arranging computations according to data locations. The characteristics of the FDCs mitigate technical, legal, and also socio-economic constraints by adopting secure mechanisms, expressive definition languages, and a rich set of architectural features that promote reliable collaborations.

In this paper, we have presented a comprehensive survey of existing biomedical tools and approaches, focused on genomic data, evaluating them against requirements for FDCs. We have proposed a taxonomy to classify tools for their support of three general properties: data and computation placement, privacy and security needs, as well as performance and scalability characteristics. We have also characterized and evaluated data-analysis techniques for biomedical purposes in the context of five biomedical use cases for genomic data processing. Note that our FDC approach can be used for any biomedical data type.

We have observed that two approaches currently support large-scale biomedical data analyses. First, the use of efficient centralized frameworks for processing large volumes of data, *e.g.*,MapReduce. Second, the biomedical community has widely adopted scientific workflow systems thanks to the ability of modeling different data analyses graphically. Unfortunately, these two approaches lack functionality for multi-site processing, secure data placement, explicit confidentiality management, and fully-distributed workflow specification, among others. This means that traditional tools and techniques lack many important features required for FDCs.

Based on these limitations, we have presented different future research opportunities. First, we have discussed research opportunities on decentralized machine learning algorithms with constraints of data sharing. We have also advocated studying such algorithms in heterogeneous and dynamic infrastructures characterized by different configurations and computing capabilities. Regarding data security, we have proposed to develop the idea of secure containers further, where raw data is protected through encapsulation from being read while maintaining their capacity to participate in complex computations. Concerning performance and scalability, we have proposed research studies on the dynamic reconfiguration of cloud infrastructures and the dynamic adaptation of infrastructure according to the scientific workflow's computational requirements. Finally, we have discussed the critical issue of the development and usage of machine learning solutions in biomedical studies.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

I would like to express my gratitude to supervisors Mario Südholt and Luis Daniel Benavides for their support and guidance during my research work. Additionally, to the co-authors Alban Gaignard and Richard Redon for their contribution to the paper from their biomedical experience.

My doctoral research is collaboratively financed by the Escuela Colombiana de Ingeniería (Bogotá - Colombia) and the IMT Atlantique (Nantes - France). I thank both universities for the agreement established to fund my doctoral thesis research.

References

[1] Abouelhoda M, Issa SA, Ghanem M. Tavaxy: integrating taverna and galaxy workflows with cloud computing support. BMC Bioinfo 2012;13:77. https://doi.org/10.1186/1471-2105-13-77.

[2] Abu-Doleh A, Catalyurek UV. Spaler: spark and GraphX based de novo genome assembler. In: 2015 IEEE international conference on big data (big data). IEEE; 2015. https://doi.org/10.1109/bigdata.2015.7363853.

[3] Abuín JM, Pichel JC, Pena TF, Amigo J. SparkBWA: speeding up the alignment of high-throughput DNA sequencing data. PLOS ONE 2016;11:e0155461. https://doi.org/10.1371/journal.pone.0155461.

[4] Al-Zoubi K, Wainer G. Modelling fog amp; cloud collaboration methods on large scale. In: 2020 winter simulation conference. WSC); 2020. p. 2161–72. https://doi.org/10.1109/WSC48552.2020.9384058.

[5] Almeida JS, Grüneberg A, Maass W, Vinga S. Fractal MapReduce decomposition of sequence alignment. Algorithm Mol Biol 2012;7. https://doi.org/10.1186/1748-7188-7-12.

[6] ANR. IntraCranial ANeurysms: from familial forms to pathophysiological mechanisms – I-CAN. 2019. http://www.agence-nationale-recherche.fr/Project-ANR-15-CE17-0008. [Accessed 10 October 2019].

[7] Atkinson M, Gesing S, Montagnat J, Taylor I. Scientific workflows: past, present and future. 2017. https://doi.org/10.1016/j.future.2017.05.041.

[8] Barillot C, Bannier E, Commowick O, Corouge I, Baire A, Fakhfakh I, Guillaumont J, Yao Y, Kain M. Shanoir: applying the software as a service distribution model to manage brain imaging research repositories. Front ICT 2016;3:25. URL: https://www.frontiersin.org/article/10.3389/fict.2016.00025.

[9] Barseghian D, Altintas I, et al. Workflows and extensions to the kepler scientific workflow system to support environmental sensor data access and analysis. Ecol Inf 2010;5:42–50. https://doi.org/10.1016/j.ecoinf.2009.08.008.

[10] Bez M, Fornari G, Vardanega T. The scalability challenge of ethereum: an initial quantitative analysis. In: 2019 IEEE international conference on service-oriented system engineering (SOSE). IEEE; 2019. https://doi.org/10.1109/sose.2019.00031.

[11] Bondiombouy C, Valduriez P. Query processing in multistore systems: an overview. Int J Cloud Comput 2016;5:309–46.

[12] zahra Boujdad F, Sudholt M. Constructive privacy for shared genetic data. In: Proceedings of the 8th international conference on cloud computing and services science. SCITEPRESS - Science and Technology Publications; 2018. https://doi.org/10.5220/0006765804890496.

[13] Boujdad FZ, Gaignard A, et al. On distributed collaboration for biomedical analyses. In: 2019 19th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGRID), IEEE; 2019. https://doi.org/10.1109/ccgrid.2019.00079.

[14] Boujdad FZ, Niyitegeka D, Bellafqira R, Gouenou C, Emmanuelle G, Südholt M. A hybrid cloud deployment architecture for privacy-preserving collaborative genome-wide association studies. In: ICDF2C 2021 - 12th EAI international conference on digital forensics & cyber crime; 2021.

[15] Bourcier R, Chatel S, et al. Understanding the pathophysiology of intracranial aneurysm: the ICAN project. Neurosurgery 2017;80:621–6. https://doi.org/10.1093/neuros/nyw135.

[16] Bux M, Brandt J, Witt C, Dowling J, Leser U. Hi-way: execution of scientific workflows on hadoop yarn. In: 20th international conference on extending database technology, EDBT 2017, 21 march 2017 through 24 march 2017, Open Proceedings. Org; 2017. p. 668–79. https://doi.org/10.5441/002/edbt.2017.87.

[17] Bux M, Leser U. Parallelization in scientific workflow management systems. 2013. arXiv preprint arXiv:1303.7195.

[18] Canali C, Lancellotti R, Mione S. Collaboration strategies for fog computing under heterogeneous network-bound scenarios. In: 2020 IEEE 19th international symposium on network computing and applications. NCA); 2020. p. 1–8. https://doi.org/10.1109/NCA51143.2020.9306730.

[19] Cano I, Weimer M, Mahajan D, Curino C, Fumarola GM. Towards geo-distributed machine learning. 2016. arXiv preprint arXiv:1603.09035.

[20] de Castro MR, dos Santos Tostes C, et al. SparkBLAST: scalable BLAST processing using in-memory operations. BMC Bioinf 2017;18. https://doi.org/10.1186/s12859-017-1723-8.

[21] Cattaneo G, Giancarlo R, et al. MapReduce in computational biology - a synopsis. 10.1007%2F978-3-319-57711-1_5. In: Advances in artificial life, evolutionary computation, and systems chemistry. Springer International Publishing; 2017. p. 53–64. URL:.

[22] Cattaneo G, Petrillo UF, Giancarlo R, Roscigno G. An effective extension of the applicability of alignment-free biological sequence comparison algorithms with hadoop. J Supercomput 2016;73:1467–83. https://doi.org/10.1007/s11227-016-1835-3.

[23] Chang YJ, Chen CC, Chen CL, Ho JM. A de novo next generation genomic sequence assembler based on string graph and MapReduce cloud computing framework. In: BMC genomics, BioMed central; 2012. S28. https://doi.org/10.1186/1471-2164-13-S7-S28.

[24] Chen Z, Hu J, Min G, Chen X. Effective data placement for scientific workflows in mobile edge computing using genetic particle swarm optimization. Concurrency Comput: Pract Ex 2019;e5413doi. https://doi.org/10.1002/cpe.5413.

[25] Chervenak A, Deelman E, Foster I, Guy L, Hoschek W, Iamnitchi A, Kesselman C, Kunszt P, Ripeanu M, Schwartzkopf B, Stockinger H, Stockinger K, Tierney B. Giggle: a framework for constructing scalable replica location services. In: ACM/IEEE SC 2002 conference (SC'02), IEEE; 2002. https://doi.org/10.1109/sc.2002.10024.

[26] Claerhout B, DeMoor G. Privacy protection for clinical and genomic data: the use of privacy-enhancing techniques in medicine. Int J Med Inf 2005;74:257–65.

[27] Cohen-Boulakia S, Belhajjame K, et al. Scientific workflows for computational reproducibility in the life sciences: status, challenges and opportunities. Future Generat Comput Syst 2017;75:284–98. https://doi.org/10.1016/j.future.2017.01.012.

[28] Colosimo ME, Peterson MW, Mardis S, Hirschman L. Nephele: genotyping via complete composition vectors and MapReduce. Source Code Biol Med 2011;6. https://doi.org/10.1186/1751-0473-6-13. URL:.

[29] Commission, E., Council. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data. http://data.europa.eu/eli/reg/2016/679/2016-05-04; 2016.

[30] Congress of Colombia. Colombian data protection law. URL: https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=49981. [Accessed 16 September 2021].

[31] Consortium DS, Consortium DM, Mahajan A, et al. Genome-wide trans-ancestry meta-analysis provides insight into the genetic architecture of type 2 diabetes susceptibility. Nature genetics 2014;46:234. https://doi.org/10.1038/ng.2897.

[32] Cook CE, Lopez R, et al. The european bioinformatics institute in 2018: tools, infrastructure and training. Nucleic Acids Res 2018;47:D15–22. https://doi.org/10.1093/nar/gky1124.

[33] Cope JM, Trebon N, Tufo HM, Beckman P. Robust data placement in urgent computing environments. In: 2009 IEEE international symposium on parallel & distributed processing. IEEE; 2009. p. 1–13. https://doi.org/10.1109/IPDPS.2009.5160914.

[34] Corpas M, Kovalevskaya NV, McMurray A, Nielsen FG. A fair guide for data providers to maximise sharing of human genomic data. PLoS Comput Biol 2018;14:e1005873. https://doi.org/10.1371/journal.pcbi.1005873.

[35] De Moor G, Claerhout B, De Meyer F. Privacy enhancing techniques. Method Inf Med 2003;42:148–53.

[36] De Roure D, Belhajjam K, Missier P, Gómez-Pérez JM, Palma R, Ruiz JE, Hettne K, Roos M, Klyne G, Goble C. Towards the preservation of scientific workflows. In: iPRES 2011-8th international conference on preservation of digital objects. National Library Board Singapore and Nanyang Technology University; 2011. p. 228–31.

[37] De Wit P, Pespeni MH, et al. The simple fool's guide to population genomics via rna-seq: an introduction to high-throughput sequencing data analysis. Mol Eco Res 2012;12:1058–67. https://doi.org/10.1111/1755-0998.12003. URL:.

[38] Decap D, Reumers J, Herzeel C, Costanza P, Fostier J. Halvade: scalable sequence analysis with MapReduce. Bioinformatics 2015;31:2482–8. https://doi.org/10.1093/bioinformatics/btv179.

[39] Deelman E, Gannon D, et al. Workflows and e-science: an overview of workflow system features and capabilities. Future Generat Comput Syst 2009;25:528–40. https://doi.org/10.1016/j.future.2008.06.012.

[40] Deelman E, Vahi K, et al. Pegasus, a workflow management system for science automation. Future Generat Comput Syst 2015;46:17–35. https://doi.org/10.1016/j.future.2014.10.008.

[41] Dolev S, Florissi P, et al. A survey on geographically distributed big-data processing using MapReduce. IEEE Transact Big Data 2019;5:60–80. https://doi.org/10.1109/tbdata.2017.2723473.

[42] Dong G, Fu X, Li H, Pan X. An accurate sequence assembly algorithm for livestock, plants and microorganism based on spark. Int J Pattern Recognit Artif Intell 2017;31:1750024. https://doi.org/10.1142/s0218001417500240.

[43] Ebrahimi M, Mohan A, Kashlev A, Lu S. Bdap: a big data placement strategy for cloud-based scientific workflows. In: 2015 IEEE first international conference on big data computing service and applications. IEEE; 2015. p. 105–14. https://doi.org/10.1109/BigDataService.2015.70.

[44] Elmroth E, Hernández F, Tordsson J. Three fundamental dimensions of scientific workflow interoperability: model of computation, language, and execution environment. Future Generat Comput Syst 2010;26:245–56.

[45] Fakas GJ, Karakostas B. A peer to peer (P2P) architecture for dynamic workflow management. Inf Software Technol 2004;46:423–31.

[46] Fan J, Han F, Liu H. Challenges of big data analysis. Nat Sci Rev 2014;1:293–314. https://doi.org/10.1093/nsr/nwt032. URL:.

[47] Federer LM, Lu YL, et al. Biomedical data sharing and reuse: attitudes and practices of clinical and scientific research staff. PLOS ONE 2015;10:e0129506. https://doi.org/10.1371/journal.pone.0129506.

[48] Freire J, Bonnet P, Shasha D. Computational reproducibility: state-of-the-art, challenges, and database research opportunities. In: Proceedings of the 2012 ACM SIGMOD international conference on management of data; 2012. p. 593–6.

[49] Frye SV, Arkin MR, et al. Tackling reproducibility in academic preclinical drug discovery. Nat Rev Drug Discovery 2015;14:733–4. https://doi.org/10.1038/nrd4737.

[50] Gil Y, Ratnakar V, et al. Wings: intelligent workflow-based design of computational experiments. IEEE Intell Syst 2011;26:62–72. https://doi.org/10.1109/mis.2010.9.

[51] Gilbert S, Lynch N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. SIGACT News 2002;33:51–9. https://doi.org/10.1145/564585.564601. URL:.

[52] Goecks J, Nekrutenko A, Taylor J, Team TG. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biol 2010;11:R86. https://doi.org/10.1186/gb-2010-11-8-r86.

[53] Goodman SN, Fanelli D, Ioannidis JPA. What does research reproducibility mean? Sci Translat Med 2016;8. https://doi.org/10.1126/scitranslmed.aaf5027. 341ps12–341ps12.

[54] Goodwin S, McPherson JD, McCombie WR. Coming of age: ten years of next-generation sequencing technologies. Nature Rev Genet 2016;17:333.

[55] Guo R, Zhao Y, Zou Q, et al. Bioinformatics applications on Apache spark. GigaScience 2018. https://doi.org/10.1093/gigascience/giy098.

[56] of Health NI, et al. Guidance: rigor and reproducibility in grant applications. 2017.

[57] Huang H, Tata S, Prill RJ. BlueSNP: R package for highly scalable genome-wide association studies using hadoop clusters. Bioinformatics 2012;29:135–6. https://doi.org/10.1093/bioinformatics/bts647.

[58] Huang L, Krüger J, Sczyrba A. Analyzing large scale genomic data on the cloud with sparkhit. Bioinformatics 2017;34:1457–65. https://doi.org/10.1093/bioinformatics/btx808.

[59] Huang Y, Gottardo R. Comparability and reproducibility of biomedical data. Briefings Bioinfo 2012;14:391–401. https://doi.org/10.1093/bib/bbs078.

[60] Hung CL, Lin YL, Hua GJ, Hu YC. CloudTSS: a TagSNP selection approach on cloud computing. In: Communications in computer and information science. Springer Berlin Heidelberg; 2011. p. 525–34. https://doi.org/10.1007/978-3-642-27180-9_64.

[61] Hutson S. Data handling errors spur debate over clinical trial. 618–618 Nature Med 2010;16. https://doi.org/10.1038/nm0610-618a.

[62] Karim MR, Michel A, et al. Improving data workflow systems with cloud services and use of open data for bioinformatics research. Briefings Bioinfo 2017;19:1035–50. https://doi.org/10.1093/bib/bbx039.

[63] Khan A, Kim T, Byun H, Kim Y. Scispace: a scientific collaboration workspace for geo-distributed hpc data centers. Future Generat Comput Syst 2019;101:398–409.

[64] Khan FZ, Soiland-Reyes S, Sinnott RO, Lonie A, Goble C, Crusoe MR. Sharing interoperable workflow provenance: a review of best practices and their practical application in cwlprov. GigaScience 2019;8:giz095.

[65] Kim D, Vouk MA. Assessing run-time overhead of securing kepler. Procedia Comput Sci 2016;80:2281–6. https://doi.org/10.1016/j.procs.2016.05.412.

[66] Kim JH. Genome data analysis. Springer Singapore; 2019. URL: https://www.springer.com/gp/book/9789811319419.

[67] Koster J, Rahmann S. Snakemake–a scalable bioinformatics workflow engine. Bioinfo 2012;28:2520–2. https://doi.org/10.1093/bioinformatics/bts480.

[68] Kuhn K, et al. The cancer biomedical informatics grid (cabig): infrastructure and applications for a worldwide research community. Medinfo 2007;1:330.

[69] Langmead B, Hansen KD, Leek JT. Cloud-scale RNA-sequencing differential expression analysis with myrna. Genome Biol 2010;11:R83. https://doi.org/10.1186/gb-2010-11-8-r83.

[70] Langmead B, Schatz MC, et al. Searching for SNPs with cloud computing. Genome Biol 2009;10:R134. https://doi.org/10.1186/gb-2009-10-11-r134.

[71] Legislature CS. The California consumer privacy act of. 2018. https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180SB1121.

[72] Leo S, Santoni F, Zanetti G. Biodoop: bioinformatics on hadoop. In: 2009 international conference on parallel processing workshops. IEEE; 2009. https://doi.org/10.1109/icppw.2009.37.

[73] Li R, Li Y, Fang X, Yang H, Wang J, Kristiansen K, Wang J. SNP detection for massively parallel whole-genome resequencing. Genome Res 2009;19:1124–32. https://doi.org/10.1101/gr.088013.108.

[74] Li X, Zhang L, et al. A novel workflow-level data placement strategy for data-sharing scientific cloud workflows. IEEE Transact Serv Comput 2016. https://doi.org/10.1109/TSC.2016.2625247.

[75] Liu J, Pacitti E, Valduriez P, Mattoso M. Parallelization of scientific workflows in the cloud. 2014.

[76] Liu J, Pacitti E, Valduriez P, Mattoso M. A survey of data-intensive scientific workflow management. J Grid Comput 2015;13:457–93. https://doi.org/10.1007/s10723-015-9329-8.

[77] Liu J, Pacitti E, Valduriez P, Mattoso M. Scientific workflow scheduling with provenance data in a multisite cloud. In: Transactions on large-scale data-and knowledge-centered systems XXXIII. Springer; 2017. p. 80–112.

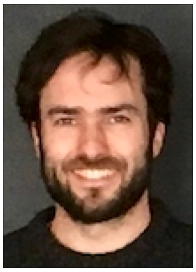
[78] Liu J, Pineda L, Pacitti E, Costan A, Valduriez P, Antoniu G, Mattoso M. Efficient scheduling of scientific workflows using hot metadata in a multisite cloud. IEEE Transact Knowl Data Eng 2019;31:1940–53. https://doi.org/10.1109/tkde.2018.2867857.

[79] Liu X, Datta A. Towards intelligent data placement for scientific workflows in collaborative cloud environment. In: 2011 IEEE international symposium on

parallel and distributed processing workshops and phd forum. IEEE; 2011. p. 1052–61. https://doi.org/10.1109/IPDPS.2011.259.
[80] Liu Y, Zhang L, Ge N, Li G. A systematic literature review on federated learning: from a model quality perspective. 2020. arXiv preprint arXiv:2012.01973.
[81] Lu S, Zhang J. Collaborative scientific workflows supporting collaborative science. Int J Bus Process Integrat Manag 2011;5:185. https://doi.org/10.1504/ijbpim.2011.040209.
[82] Lu YY, Tang K, et al. CAFE: aCcelerated Alignment-FrEe sequence analysis. Nucleic acids research 2017;45:W554–9. https://doi.org/10.1093/nar/gkx351.
[83] Malin BA, Emam KE, O'Keefe CM. Biomedical data privacy: problems, perspectives, and recent advances. 2013.
[84] McKenna A, Hanna M, Banks E, et al. The genome analysis toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res 2010;20:1297–303. https://doi.org/10.1101/gr.107524.110.
[85] McMahan B, Moore E, Ramage D, Hampson S, y Arcas BA. Communication-efficient learning of deep networks from decentralized data. In: Singh A, Zhu J, editors. Proceedings of the 20th international conference on artificial intelligence and statistics. Fort Lauderdale, FL, USA: PMLR; 2017. p. 1273–82. URL: http://proceedings.mlr.press/v54/mcmahan17a.html.
[86] Moreau L, Missier P, Cheney J, Soiland-Reyes S. Prov-n: the provenance notation. 2013.
[87] Nagappan M, Vouk MA. A model for sharing of confidential provenance information in a query based system. In: International provenance and annotation workshop. Springer; 2008. p. 62–9. https://doi.org/10.1007/978-3-540-89965-5_8.
[88] Nguyen T, Shi W, Ruden D. CloudAligner: a fast and full-featured MapReduce based tool for sequence mapping. BMC Res Notes 2011;4. https://doi.org/10.1186/1756-0500-4-171.
[89] NHGRI-EBI. GWAS catalog. 2019. https://www.ebi.ac.uk/gwas/. accessed 20-Sept-2019.
[90] NIH-BMIC. NIH data sharing repositories. 2019. https://www.nlm.nih.gov/NIHbmic/nih_data_sharing_repositories.html. accessed 20-Sept-2019.
[91] Nordberg H, Bhatia K, Wang K, Wang Z. BioPig: a hadoop-based analytic toolkit for large-scale sequence data. Bioinformatics 2013;29:3014–9. https://doi.org/10.1093/bioinformatics/btt528.
[92] NSF, 2019. Chapter XI - Other Post Award Requirements and Consideration. https://www.nsf.gov/pubs/policydocs/pappg19_1/pappg_11.jsp\#XID4. [Online; accessed 20-June-2019].
[93] O'Brien AR, Saunders NFW, et al. VariantSpark: population scale clustering of genotype information. BMC Genom 2015;16. https://doi.org/10.1186/s12864-015-2269-7.
[94] Pandey RV, Schlötterer C. DistMap: a toolkit for distributed short read mapping on a hadoop cluster. PLoS ONE 2013;8:e72614. https://doi.org/10.1371/journal.pone.0072614.
[95] Papageorgiou L, Eleni P, et al. Genomic big data hitting the storage bottleneck. EMBnetjournal 2018;24:e910. https://doi.org/10.14806/ej.24.0.910.
[96] Parks R, Chu CH, Xu H. Healthcare information privacy research: iusses, gaps and what next? AMCIS; 2011.
[97] Peteiro-Barral D, Guijarro-Berdiñas B. A survey of methods for distributed machine learning. Prog Artif Intell 2013;2:1–11.
[98] Pineda-Morales L, Costan A, Antoniu G. Towards multi-site metadata management for geographically distributed cloud workflows. In: 2015 IEEE international conference on cluster computing. IEEE; 2015. p. 294–303. https://doi.org/10.1109/cluster.2015.49.
[99] Pineda-Morales L, Liu J, Costan A, Pacitti E, Antoniu G, Valduriez P, Mattoso M. Managing hot metadata for scientific workflows on multisite clouds. In: 2016 IEEE international conference on big data (big data). IEEE; 2016. p. 390–7.
[100] Pireddu L, Leo S, Zanetti G. SEAL: a distributed short read mapping and duplicate removal tool. Bioinformatics 2011;27:2159–60. https://doi.org/10.1093/bioinformatics/btr325.
[101] Rasheed Z, Rangwala H. A map-reduce framework for clustering metagenomes. In: 2013 IEEE international symposium on parallel & distributed processing, workshops and phd forum. IEEE; 2013. https://doi.org/10.1109/ipdpsw.2013.100.
[102] Rodriguez MA, Buyya R. Scientific workflow management system for clouds. In: Software architecture for big data and the cloud. Elsevier; 2017. p. 367–87. https://doi.org/10.1016/b978-0-12-805467-3.00018-1.
[103] Ross RB, Thakur R, et al. Pvfs: a parallel file system for linux clusters. In: Proceedings of the 4th annual Linux showcase and conference; 2000. p. 391–430.
[104] Rynge M, Vahi, et al. Integrity protection for scientific workflow data: motivation and initial experiences. In: Proceedings of the practice and experience in advanced research computing on rise of the machines (learning). ACM; 2019. p. 17. https://doi.org/10.1145/3332186.3332222.
[105] Salloum S, Dautov R, et al. Big data analytics on Apache spark. Int J Data Sci Anal 2016;1:145–64. https://doi.org/10.1007/s41060-016-0027-9. URL:.
[106] Santana-Perez I, Pérez-Hernández MS. Towards reproducibility in scientific workflows: an infrastructure-based approach. Scientific Program 2015:1–11. https://doi.org/10.1155/2015/243180. 2015.
[107] Schadt EE, Linderman MD, et al. Computational solutions to large-scale data management and analysis. Nature Rev Genet 2010;11:647–57. https://doi.org/10.1038/nrg2857.
[108] Schatz MC. BlastReduce: high performance short read mapping with MapReduce. University of Maryland; 2008. http://cgis.cs.umd.edu/Grad/scholarlypapers/papers/MichaelSchatz.pdf.
[109] Schatz MC. CloudBurst: highly sensitive read mapping with MapReduce. Bioinfo 2009;25:1363–9. https://doi.org/10.1093/bioinformatics/btp236.
[110] Schatz MC, Sommer D, Kelley D, Pop M. De novo assembly of large genomes using cloud computing. In: Proceedings of the cold spring harbor biology of genomes conference; 2010.
[111] Schmuck FB, Haskin RL. Gpfs: a shared-disk file system for large computing clusters. In: FAST; 2002.
[112] Senturk IF, Balakrishnan P, et al. A resource provisioning framework for bioinformatics applications in multi-cloud environments. Future Generat Comput Syst 2018;78:379–91. https://doi.org/10.1016/j.future.2016.06.008.
[113] Sharov AA, Schlessinger D, Ko MSH. ExAtlas: an interactive online tool for meta-analysis of gene expression data. J Bioinfo Comput Biol 2015;13:1550019. https://doi.org/10.1142/s0219720015500195.
[114] da Silva RF, et al. A characterization of workflow management systems for extreme-scale applications. Future Generat Comput Syst 2017;75:228–38. https://doi.org/10.1016/j.future.2017.02.026.
[115] Soiland-Reyes S, Alper P, Goble C. Tracking workflow execution with tavernaprov. In: PROV: three tears later: Provenance Week 2016; 2016.
[116] Stephens ZD, Lee SY, et al. Big data: astronomical or genomical? PLOS Biology 2015;13:e1002195. https://doi.org/10.1371/journal.pbio.1002195.
[117] Tannenbaum T, Wright D, Miller K, Livny M. Condor: a distributed job scheduler. In: Beowulf cluster computing with windows; 2001. p. 307–50.
[118] Taylor I, Shields M, Wang I, Harrison A. The triana workflow environment: architecture and applications. In: Workflows for e-Science. Springer; 2007. p. 320–39. https://doi.org/10.1007/978-1-84628-757-2_20. URL:.
[119] Taylor IJ, Deelman E, et al. Workflows for e-Science: scientific workflows for grids, ume 1. Springer; 2007. https://doi.org/10.1007/978-1-84628-757-2.
[120] Thain D, Tannenbaum T, Livny M. Distributed computing in practice: the condor experience. Concurr Comput: Pract Exp 2005;17:323–56. https://doi.org/10.1002/cpe.938.
[121] Tommaso PD, Chatzou M, et al. Nextflow enables reproducible computational workflows. Nature Biotechnol 2017;35:316–9. https://doi.org/10.1038/nbt.3820.
[122] Turakhia MP, Desai M, Hedlin H, Rajmane A, Talati N, Ferris T, Desai S, Nag D, Patel M, Kowey P, Rumsfeld JS, Russo AM, Hills MT, Granger CB, Mahaffey KW, Perez MV. Rationale and design of a large-scale, app-based study to identify cardiac arrhythmias using a smartwatch: the apple heart study. Am Heart J 2019;207:66–75. https://doi.org/10.1016/j.ahj.2018.09.002. https://www.sciencedirect.com/science/article/pii/S0002870318302710.
[123] Union I. Communication from the commission to the european parliament, the council, the european economic and social committee and the committee of the regions. A new skills agenda for europe. 2014 [Brussels].
[124] Valduriez P, Mattoso M, Akbarinia R, Borges H, Camata J, Coutinho A, Gaspar D, Lemus N, Liu J, Lustosa H, et al. Scientific data analysis using data-intensive scalable computing: the scidisc project. In: LADaS: Latin America data science workshop, CEUR-WS. Org; 2018.
[125] Van Hung T, Chuanhe H. An effective data placement strategy in main-memory database cluster. In: 2011 second international conference on networking and distributed computing. IEEE; 2011. p. 93–8. https://doi.org/10.1109/ICNDC.2011.27.
[126] Verbraeken J, Wolting M, Katzy J, Kloppenburg J, Verbelen T, Rellermeyer JS. A survey on distributed machine learning. ACM Comput Surv (CSUR) 2020;53:1–33.
[127] Wang J, Crawl D, Altintas I. Kepler + hadoop. In: Proceedings of the 4th workshop on workflows in support of large-scale science - WORKS '09. ACM Press; 2009. https://doi.org/10.1145/1645164.1645176.
[128] Wang R, Li M, Peng L, Hu Y, Hassan MM, Alelaiwi A. Cognitive multi-agent empowering mobile edge computing for resource caching and collaboration. Future Generat Comput Syst 2020;102:66–74. https://doi.org/10.1016/j.future.2019.08.001. URL: https://www.sciencedirect.com/science/article/pii/S0167739X19318783.
[129] Wang Y. Automating experimentation with distributed systems using generative techniques. Ph.D. thesis. University of Colorado at Boulder; 2006.
[130] Wang Y, Carzaniga A, Wolf AL. Four enhancements to automated distributed system experimentation methods. In: Proceedings of the 30th international conference on Software engineering; 2008. p. 491–500.
[131] Wiewiórka MS, Messina A, et al. SparkSeq: fast, scalable and cloud-ready tool for the interactive genomic data analysis with nucleotide precision. Bioinformatics 2014;30:2652–3. https://doi.org/10.1093/bioinformatics/btu343.
[132] Wilde M, Hategan M, et al. Swift: a language for distributed parallel scripting. Parallel Comput 2011;37:633–52. https://doi.org/10.1016/j.parco.2011.05.005.
[133] Wolstencroft K, Haines R, et al. The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. Nucleic Acids Res 2013;41:W557–61. https://doi.org/10.1093/nar/gkt328.
[134] Xiao Y, Zhou AC, Yang X, He B. Privacy-preserving workflow scheduling in geo-distributed data centers. Future Generat Comput Syst 2022;130:46–58.
[135] Xie J, Yin S, et al. Improving mapreduce performance through data placement in heterogeneous hadoop clusters. In: 2010 IEEE international symposium on parallel & distributed processing, workshops and phd forum (IPDPSW). IEEE; 2010. p. 1–9. https://doi.org/10.1109/IPDPSW.2010.5470880.
[136] Xie T. Sea: a striping-based energy-aware strategy for data placement in raid-structured storage systems. IEEE Transact Comput 2008;57:748–61. https://doi.org/10.1109/TC.2008.27.
[137] Xing EP, Ho Q, Dai W, Kim JK, Wei J, Lee S, Zheng X, Xie P, Kumar A, Yu Y. Petuum: a new platform for distributed machine learning on big data. IEEE Transact Big Data 2015;1:49–67. https://doi.org/10.1109/tbdata.2015.2472014.

[138] Xu B, Gao J, Li C. An efficient algorithm for DNA fragment assembly in MapReduce. Biochem Biophys Res Commun 2012;426:395–8. https://doi.org/10.1016/j.bbrc.2012.08.101.
[139] Xu B, Li C, Zhuang H, et al. DSA: scalable distributed sequence alignment system using SIMD instructions. In: 2017 17th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGRID), IEEE; 2017. https://doi.org/10.1109/ccgrid.2017.74.
[140] Xu B, Li C, Zhuang H, et al. Efficient distributed smith-waterman algorithm based on Apache spark. In: 2017 IEEE 10th international conference on cloud computing (CLOUD). IEEE; 2017. https://doi.org/10.1109/cloud.2017.83.
[141] Yu HF, Hsieh CJ, Chang KW, Lin CJ. Large linear classification when data cannot fit in memory. In: ACM Transactions on Knowledge Discovery from Data (TKDD); 2012. p. 1–23. 5.
[142] Yu J, Buyya R. A taxonomy of workflow management systems for grid computing. J Grid Comput 2005;3:171–200. https://doi.org/10.1007/s10723-005-9010-8.
[143] Yuan D, Yang Y, Liu X, Chen J. A data placement strategy in scientific cloud workflows. Future Generat Comput Syst 2010;26:1200–14. https://doi.org/10.1016/j.future.2010.02.004.
[144] Zhang D, Zhao L, Li B, et al. SEQSpark: a complete analysis tool for large-scale rare variant association studies using whole-genome and exome sequence data. The American J Human Genet 2017;101:115–22. https://doi.org/10.1016/j.ajhg.2017.05.017.
[145] Zhang L, Gu S, Liu Y, Wang B, Azuaje F. Gene set analysis in the cloud. Bioinformatics 2011;28:294–5. https://doi.org/10.1093/bioinformatics/btr630.
[146] Zhao G, Ling C, Sun D. SparkSW: scalable distributed computing system for large-scale biological sequence alignment. In: 2015 15th IEEE/ACM international symposium on cluster, cloud and grid computing, IEEE; 2015. https://doi.org/10.1109/ccgrid.2015.55.
[147] Zhao J, Gomez-Perez JM, Belhajjame K, Klyne G, Garcia-Cuesta E, Garrido A, Hettne K, Roos M, De Roure D, Goble C. Why workflows break—understanding and combating decay in taverna workflows. In: 2012 ieee 8th international conference on e-science. IEEE; 2012. p. 1–9.
[148] Zhao Q, Xiong, et al. A new energy-aware task scheduling method for data-intensive applications in the cloud. J Network Comput Appl 2016;59:14–27. https://doi.org/10.1016/j.jnca.2015.05.001.
[149] Zhao Y, Li Y, Raicu I, Lu S, Tian W, Liu H. Enabling scalable scientific workflow management in the cloud. Future Generat Comput Syst 2015;46:3–16. https://doi.org/10.1016/j.future.2014.10.023.
[150] Zhou W, Li R, Yuan S, et al. MetaSpark: a spark-based distributed processing tool to recruit metagenomic reads to reference genomes. Bioinformatics 2017. https://doi.org/10.1093/bioinformatics/btw750. btw750.
[151] Zielezinski A, Vinga S, Almeida J, Karlowski WM. Alignment-free sequence comparison: benefits, applications, and tools. Genome Biol 2017;18. https://doi.org/10.1186/s13059-017-1319-7.
[152] Zytnicki M, Quesneville H. S-MART, a software toolbox to aid RNA-seq data analysis. PLoS ONE 2011;6:e25988. https://doi.org/10.1371/journal.pone.0025988.

Wilmer Garzón Alfonso is a doctoral researcher from IMT Atlantique (Nantes - France) and Escuela Colombiana de Ingeniería (Bogotá - Colombia). He received a B.Sc. in computer science in 2006 and his B.Sc. in Mathematics in 2008 from Escuela Colombiana de Ingeniería. He studied a Master of Science in Computer Engineering at the University of Puerto Rico. He currently is assistant professor at Escuela Colombiana de Ingeniería. His areas of interest include data analytics, data mining, learning techniques, distributed systems applications, and formal methods.



Alban Gaignard is a CNRS research engineer at l'Institut du Thorax in Nantes, France. He holds a Ph.D. in Computer Science from the University of Nice-Sophia Antipolis since 2013. His research interests cover knowledge representations (knowledge graphs, data integration) and distributed systems (workflows, provenance, large scale computing infrastructures) in the context of life sciences.

Luis Daniel Benavides Navarro holds a PhD in computer science from the University of Nantes (France), a Master in computer science from the Vrije Universiteit Brussel (Belgium), a Specialization degree in software construction from the Universidad de Los Andes (Colombia) and an Electrical Engineering degree from Universidad de Los Andes. He currently serves as an associate professor at Escuela Colombiana de Ingeniería. His areas of interest include Enterprise Architecture, Software Engineering, Distributed Computing, Distributed Programming Languages, and Models for the development of distributed, complex and concurrent applications.

Mario Südholt is a full professor in Computer Science at IMT Atlantique, France, where he holds the chair in Software Engineering and Programming Languages. He holds an MSc from U. of Koblenz, Germany, a PhD from TU Berlin, Germany, and a habilitation from U. of Nantes, France. His research interests focus on large-scale distributed software systems, their formal definition, compositional properties, efficient implementation, and corresponding support in programming languages. He applies his results mainly to problems involving information systems, the Cloud and the health sector. He is a member of the steering committee of the scientific journal Programming and co-authored more than 100 peer-reviewed publications.

Richard Redon is Research Director at the INSERM, and the Lab Director of l'institut du thorax at Nantes University. RR is recognized as an international expert in the analysis of structural variation in the human genome. He has also contributed to the development of NGS-based approaches to address the genetics of rare diseases, and has been leading international genome-wide association studies based on array genotyping and whole-genome sequencing and aiming to identify genetic risk factors for cardiovascular diseases.