BiobankCloud: a Platform for the Secure Storage, Sharing, and Processing of Large Biomedical Data Sets

Alysson Bessani[5], Jörgen Brandt[2], Marc Bux[2], Vinicius Cogo[5], Lora Dimitrova[4], Jim Dowling[1], Ali Gholami[1], Kamal Hakimzadeh[1], Micheal Hummel[4], Mahmoud Ismail[1], Erwin Laure[1], Ulf Leser[2], Jan-Eric Litton[3], Roxanna Martinez[3], Salman Niazi[1], Jane Reichel[6], Karin Zimmermann[4]

[1] KTH - Royal Institute of Technology,
{jdowling, gholami, mahh, maism, erwinl, smkniazi}@kth.se
[2] Humboldt University
{leser, bux, joergen.brandt}@informatik.hu-berlin.de
[3] Karolinska Institute
{Jan-Eric.Litton, Roxanna.Martinez}@ki.se
[4] Charite
{Michael.Hummel, Lora.Dimitrova, Karin.Zimmermann}@charite.de
[5] LaSIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal
{bessani, vielmo}@lasige.di.fc.ul.pt
[6] Uppsala University
{jane.reichel}@jur.uu.se

Abstract. Biobanks store and catalog human biological material that is increasingly being digitized using next-generation sequencing (NGS). There is, however, a computational bottleneck, as existing software systems are not scalable and secure enough to store and process the incoming wave of genomic data from NGS machines. In the BiobankCloud project, we are building a Hadoop-based platform for the secure storage, sharing, and parallel processing of genomic data. We extended Hadoop to include support for multi-tenant studies, reduced storage requirements with erasure coding, and added support for extensible and consistent metadata. On top of Hadoop, we built a scalable scientific workflow engine featuring a proper workflow definition language focusing on simple integration and chaining of existing tools, adaptive scheduling on Apache Yarn, and support for iterative dataflows. Our platform also supports the secure sharing of data across different, distributed Hadoop clusters. The software is easily installed and comes with a user-friendly web interface for running, managing, and accessing data sets behind a secure 2-factor authentication. Initial tests have shown that the engine scales well to dozens of nodes. The entire system is open-source and includes pre-defined workflows for popular tasks in biomedical data analysis, such as variant identification, differential transcriptome analysis using RNA-Seq, and analysis of miRNA-Seq and ChIP-Seq data.

1 Introduction

Biobanks store and catalog human biological material from identifiable individuals for both clinical and research purposes. Recent initiatives in personalized medicine created a steeply increasing demand to sequence the human biological material stored in biobanks. As of 2015, such large-scale sequencing is under way in hundreds of projects around the world, with the largest single project sequencing up to 100.000 genomes[7]. Furthermore, sequencing also is becoming more and more routine in a clinical setting for improving diagnosis and therapy especially in cancer [1]. However, software systems for biobanks traditionally managed only metadata associated with samples, such as pseudo-identifiers for patients, sample collection information, or study information. Such systems cannot cope with the current requirement to, alongside such metadata, also store and analyze genomic data, which might mean everything from a few Megabytes (e.g., genotype information from a SNP array) to hundreds of Gigabytes per sample (for whole genome sequencing with high coverage).

For a long time, such high-throughput sequencing and analysis was only available to large research centers that (a) could afford enough modern sequencing devices and (b) had the budget and IT expertise to manage high performance computing clusters. This situation is changing. The cost of sequencing is falling rapidly, and more and more labs and hospitals depend on sequencing information for daily research and diagnosis/treatment. However, there is still a pressing need for flexible and open software systems to enable the computational analysis of large biomedical data sets at a reasonable price. Note that this trend is not restricted to genome sequencing; very similar developments are also happening in other medical areas, such as molecular imaging [2], drug discovery [3], or data generated from patient-attached sensors [4].

In this paper, we present the BiobankCloud platform, a collaborative project bringing together computer scientists, bioinformaticians, pathologists, and biobankers. The system is designed as a "platform-as-a-service", i.e., it can be easily installed on a local cluster (or, equally well, in a public cloud) using Karamel and Chef[8]. Primary design goals are flexibility in terms of the analysis being performed, scalability up to very large data sets and very large cluster set-ups, ease of use and low maintenance cost, strong support for data security and data privacy, and direct usability for users. To this end, it encompasses (a) a scientific workflow engine running on top of the popular Hadoop platform for distributed computing, (b) a scientific workflow language focusing on easy integration of existing tools and simple rebuilding of existing pipelines, (c) support for automated installation, and (d) role-based access control. It also features (e) HopsFS, a new version of Hadoop's Distributed Filesystem (HDFS) with improved throughput, supported for extended metadata, and reduced storage requirements compared to HDFS, (f) Charon, which enables the federation of clouds at the file system level, and (g) a simple Laboratory Information Management Service with an

[7] See `http://www.genomicsengland.co.uk/`.

[8] `http://www.karamel.io`

integrated web interface for authenticating/authorizing users, managing data, designing and searching for metadata, and support for running workflows and analysis jobs on Hadoop. This web interface hides much of the complexity of the Hadoop backend, and supports multi-tenancy through first-class support for *Studies*, *SampleCollections* (DataSets), *Samples*, and *Users*.

In this paper, we give an overview on the architecture of the BiobankCloud platform and describe each component in more detail. The system is currently under development; while a number of components already have been released for immediate usage (e.g., Hops, SAASFEE), a first overall platform release is planned for the near future. The system is essentially agnostic to the type of data being managed and the types of analysis being performed, but developed with genome sequencing as most important application area. Therefore, throughout this paper we will use examples from this domain.

2 Related Work

Our platform covers a broad set of technologies providing components that solve problems in the areas of security in Hadoop, sharing data, parallel data processing, and data management. Here, we focus on related work in the area of parallel data analytics and discuss some general and some domain-specific solutions. In general, research in platforms for large-scale data analysis is flourishing over the last years. General purpose data parallel processing systems like Spark [5] or Flink [6] support efficient and distributed data analysis, but focus on providing SQL support for querying data. This leads to some design choices that make supporting scientific analysis pipelines (or scientific workflows) rather cumbersome. In contrast, specialized scientific workflow management systems like Taverna, Kepler, or Galaxy, typically neglect the Big Data aspect, i.e., they are not able to scale workflow executions efficiently over large clusters [7].

The lack of a flexible and open platform for running complex analysis pipelines over large data sets led to the development of a number of highly specialized systems. For instance, tools like Crossbow [8] perform one specific operation (sequence alignment) in a distributed manner, but cannot be used for any other type of analysis. Also for the most important types of complex analysis, like sequence assembly or mutation detection and evaluation, specialized systems exist. Some very popular systems, like GATK [9], only support parallelism on a single machine and cannot scale-out to make use of additional computational resources. Technically more advanced solutions like Adam [10], Halvade [11], Seal [12], and PigSeq [13] show much better scaling, but are not general purpose solutions but specifically developed for one type of analysis. Appropriate models of data security and privacy are not supported in any of the systems we are aware of. There are frameworks that enhance Hadoop with access control, such as Apache Ranger that supports attribute-based access control and is general and expressive enough to be applied to most Hadoop services. However, attribute-based access control does not have high enough throughput to support authorization of all HDFS operations at the NameNode or all application requests at the ResourceManager. Overall, we see a clear lack of a flexible, open, secure, and scalable platform for scientific data analysis on large data sets.

3 Architecture

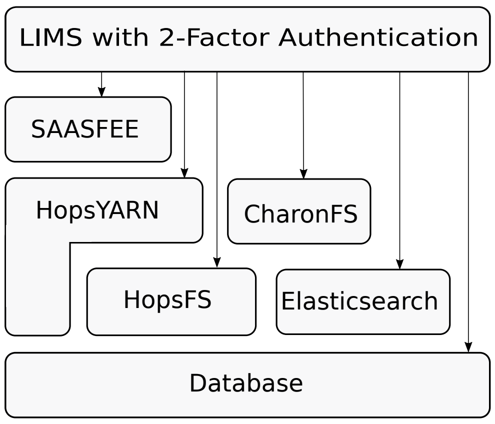


Fig. 1: BiobankCloud Architecture

Our platform has a layered architecture (see Figure 1). In a typical installation, users will access the system through the web interface with 2-factor authentication. From there, she can access all services, such as the enhanced file system HopsFS (see section 5), the workflow execution engine SAASFEE (see section 6), the federated cloud service CharonFS (see section 8), and an Elasticsearch instance to search through an entire installation. SAASFEE is built over YARN, while CharonFS can use HopsFS as a backing store. HopsFS and Elasticsearch use a distributed, in-memory database for metadata management. Note that all services can also be accessed directly through command-line interfaces.

Data Sets for Hadoop

The web interface has integrated a LIMS to manage the typical data items inside a biobank, and to provide fine-grained access control to these items. These items are also reflected in the Hadoop installation. Specifically, BiobankCloud introduces **DataSets** as a new abstraction, where a DataSet consists of a related group of directories, files, and extended metadata. DataSets can be indexed and searched (through Elasticsearch) and are the basic unit of data management in BiobankCloud; all user-generated files or directories belong to a single DataSet. In biobanking, a sample collection would be a typical example of a DataSet. To allow for access control of users to DataSets, which is not inherent in the

DataSet concept, we introduce the notion of **Studies**. A Study is a grouping of researchers and DataSets (see Figure 2) and the basic unit of privacy protection (see below).

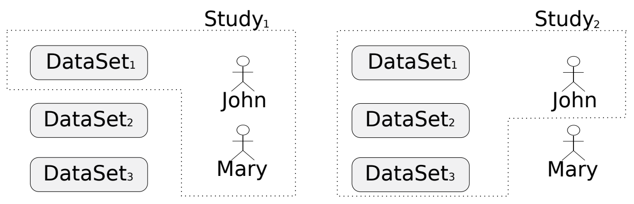


Fig. 2: Study1 has John and Mary as users and includes DataSet1, while Study2 has only John as as a user and includes DataSet1, DataSet2, and DataSet3.

4 Security Model

The BiobankCloud environment deploys strong security features for concerns such as confidentiality, integrity, and non-repudiation [14] of data access. This includes authentication, authorization, and auditing. The system allows defining different roles with different access privileges. In designing the system, we applied the Cloud Privacy Threat Modeling [15] approach to identify the privacy requirements of processing sensitive biomedical data. This model implements a data management policy that aligns with the European data protection directive. The project further implements tools to ensure the correct allocation of legal responsibilities for the data processed within the platform.

Figure 3 shows the different components of the employed security mechanisms. All BiobankCloud services are protected behind the firewall and only accessible through the secure interfaces over HTTPS channels.

4.1 2-Factor Authentication

The authentication services map the person accessing the platform to a user identity. We provide 2-factor authentication using smart mobile devices or Yubikey[9] hardware tokens to support different groups of users. Users send authentication requests via a Web browser to the authentication service that runs instances of the time-based one-time password (TOTP) and Yubikey one-time password (YOTP) protocols.

In a mobile scenario, a user supplies an one-time generated password by a commercial authenticator in addition to a simple password that was decided during the account registration. The login page authenticates the user to the

[9] Yubikey Manual, `http://www.yubico.com`

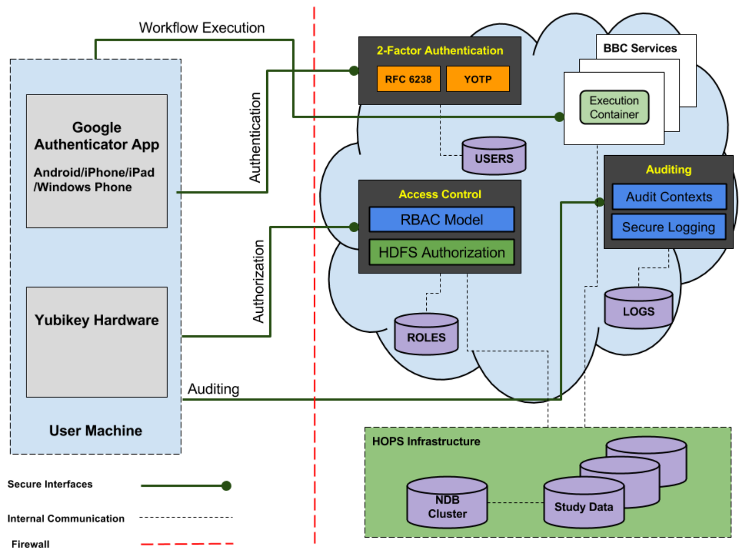


Fig. 3: Secure access to the BiobankCloud via a web front-end.

platform using the TOTP module (Time-Based One-Time Password) as an implementation of the RFC 6238. In contrast, a Yubikey user would enter the Yubikey device into a USB port. The user enters the simple password that was decided during the account registration and pushes the Yubikey button. The Yubikey login page authenticates the user to the platform via the YOTP module.

4.2 Role-based Access Control

The access control component ensures authorized access to all data and services within a platform installation. Therefore, the system defines several roles[10] which are granted certain access rights to certain studies. Examples are a DataOwner (users who may create new data sets), a DataScientist (users who can run workflows on the data), or an auditor (users with access to audit trails for auditing). DataSets technically can be shared between studies, and users may have different roles in different studies. We use the access control mechanism of HopsFS to implement the Study- and DataSet-based authorization model.

4.3 Auditing Service

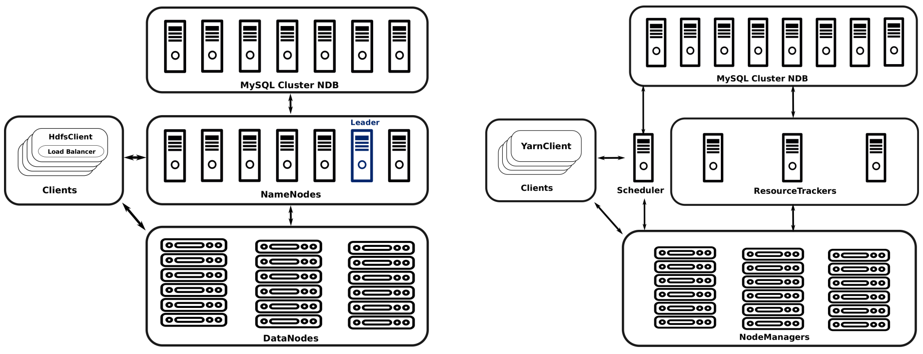
Finally, the auditing service enables the platform administrator or an external auditor to discover the history of accessing the platform to detect any violation to a policy. It includes several contexts such as role, account, study, and login audits.

[10] The concrete roles should be seen as implementations of the European Data Protection Directive.

The secure login service assures that actions that are taken by the users are registered for tracing and auditing purposes. Each log event contains information such as initiator, target, IP/MAC addresses, timestamp, action, and outcome.

5 Hadoop Open Platform-as-a-Service (Hops)

A full installation of our platform builds on an adapted distribution of the Hadoop File System (HDFS), called HopsFS, which builds on a new metadata management architecture based on a shared-nothing, in-memory distributed database (see Figure 4). Provided enough main memory in the nodes, metadata can grow to TBs in size with our approach (compared to 100GB in Apache HDFS [16]), which allows HopsFS to store 100s of millions of files. The HopsFS architecture includes multiple stateless NameNodes that manage the namespace metadata stored in the database (see Figure 4a). HopsFS' clients and DataNodes are aware of all NameNodes in the system. HopsFS is highly available: whenever a NameNode fails the failed operations are automatically retried by clients and the DataNodes by forwarding the failed requests to a different live NameNode. We use MySQL Cluster [17] as the database, as it has high throughput and is also highly available, although any distributed in-memory database that supports transactions and row level locking could be used. On database node failures, failed transactions are re-scheduled by NameNodes on surviving database nodes.



(a) HopsFS (b) HopsYARN

Fig. 4: HopsFS and HopsYARN architectures.

We ensure the consistency of the file system metadata by implementing serialized transactions on well-ordered operations on metadata [18]. A leader NameNode is responsible for file system maintenance tasks, and leader failure triggers our own leader-election service based on the database [19]. HopsFS can reduce the amount of storage space required to store genomic data, while maintaining high availability by storing files using Reed-Solomon erasure coding, instead of the traditional three-way replication used in HDFS. Erasure-coding can reduce disk space consumption by 44% compared to three-way replication. In HopsFS, an ErasureCodingManager runs on the leader NameNode, managing file encoding and file repair operations, as well as implementing a policy that places file blocks on DataNodes in such a way that ensures that, in the event of a DataNode failure, affected files can still be repaired.

Designing, Indexing, and Searching Extended Metadata

We store genomes in HopsFS. However, biobanks require much more extensive metadata for genomes than is available for HDFS files. The limited metadata available in HDFS files includes file size, time last modified, and owner. We also need information such as the sample and sample collection the genome belongs to, the type of sample, and donor information. Our LIMS provides a UI tool for biobankers who are not programmers to design their own extended metadata that is linked to genomes, sample collections, DataSets, or Studies. This extended metadata is stored in the same database as the file system metadata and the integrity of the extended metadata is guaranteed using foreign keys to the file or directory the metadata refers to. To make this extended metadata searchable, we asynchronously and transparently replicate it to Elasticsearch. This indexing of extended metadata enables free-text searching for samples.

HopsYARN

HopsYARN is our implementation of Apache YARN, in which we have (again) migrated the metadata to MySQL Cluster. We partitioned YARN's ResourceManager into (1) ResourceTracker nodes that process heartbeats from and send commands to NodeManagers, and (2) a single scheduler node that implements all other ResourceManager services, see Figure 4. If the scheduler node fails, our leader election service will elect a ResourceTracker node as the new scheduler that then loads the scheduler state from the database. HopsYARN scales to handle larger clusters than Apache YARN as resource tracking has been offloaded from the scheduler node to other nodes, and resource tracking traffic grows linearly with cluster size. This will, in time, enable larger numbers of genomes to be analyzed in a single system.

6 SAASFEE

To process the vast amounts of genomic data stored in today's biobanks, researchers have a diverse ecosystem of tools at their disposal [20]. Depending on the research question at hand, these tools are often used in conjunction with one another, resulting in complex and intertwined analysis pipelines. Scientific workflow management systems (SWfMSs) facilitate the design, refinement, execution, monitoring, sharing, and maintenance of such analysis pipelines. SAASFEE [21] is a SWfMS that supports the scalable execution of arbitrarily complex workflows. It encompasses the functional workflow language Cuneiform as well as Hi-WAY, a higher-level scheduler for both Hadoop YARN and HopsYARN. See Figure 5 for the complete software stack of SAASFEE.

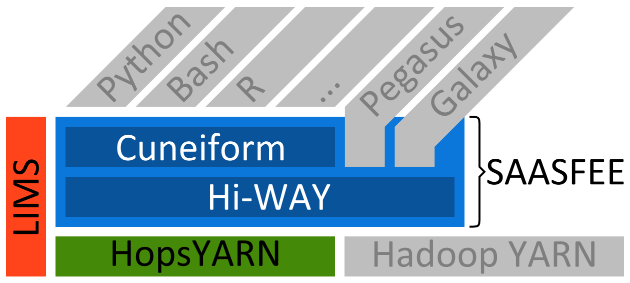


Fig. 5: The software stack of the scientific workflow management system SAASFEE, which comprises the functional workflow language Cuneiform as well as the Hi-WAY workflow scheduler for Hadoop. Cuneiform can execute foreign code written in languages like Python, Bash, and R. Besides Cuneiform, Hi-WAY can also interpret the workflow languages of the SWfMSs Pegasus and Galaxy. SAASFEE can be run both on Hops as well as Apache Hadoop. SAASFEE and HopsYARN can be interfaced and configured via the web interface provided by the LIMS.

Analysis pipelines for large-scale genomic data employ many different software tools and libraries with diverse Application Programming Interfaces (APIs). At the same time the growing amounts of data to be analyzed necessitate parallel and distributed execution of these analysis pipelines. Thus, the methods for specifying such analysis pipelines need to meet both concerns – integration and parallelism equally. The functional workflow Language Cuneiform has been designed to meet these requirements [22]. Cuneiform allows the integration of software tools and libraries with APIs in many different programming languages. This way, command-line tools (e.g., Bowtie [23]) can be integrated with similar ease as, for instance, R libraries (e.g., CummeRbund [24]). By partitioning large data sets and processing these partitions in parallel, data parallelism can be exploited in addition to task parallelism to speed up computation. Cuneiform automatically detects and exploits data and task parallelism in a workflow specification. Editing and debugging workflows is supported by the tools and visualization features provided with the Cuneiform interpreter.

Hi-WAY is a higher-level scheduler that enables the execution of scientific workflows on top of YARN. Hi-WAY executes each of the tasks comprising the workflow in a separate container, which is the basic unit of computation in YARN. Input, output, and intermediate files created during a workflow execution are stored in Hadoop's distributed file system HDFS. Consequently, Hi-WAY benefits form the fault-tolerance and scalability of the Hadoop ecosystem. It has been evaluated to scale to more than 600 concurrent tasks.

Hi-WAY provides a selection of established scheduling policies conducting task placement based on (a) the locality of a task's input data to diminish network load and (b) task runtime estimation based on past measurements to utilize resources efficiently. To enable repeatability of experiments, Hi-WAY generates exhaustive provenance traces during workflow execution, which can be shared and re-executed or archived in a database. One of the major distinctive features of SAASFEE is its strong emphasis on integration of external software. This is true for both Cuneiform, which is able to integrate foreign code and command-line tools, and Hi-WAY, which is capable of running not only Cuneiform workflows, but also workflows designed in the SWfMSs Pegasus [25] and Galaxy [26].

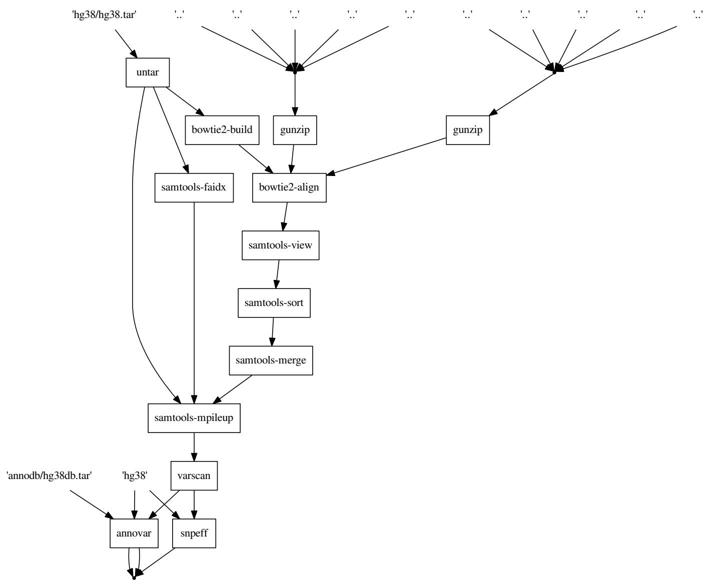


Fig. 6: Static call graph of variant calling workflow. Box-shaped nodes represent distinct tasks, edges represent data dependencies.

We demonstrate the applicability of SAASFEE for large-scale biobank use cases by discussing an example workflow for variant calling. In this use case we try to discover differences in the genome of an organism in comparison to a reference genome. Furthermore, the discovered differences are annotated with database information to ease their interpretation by a specialist. Figure 6 shows the static call graph for this workflow which was automatically derived from the workflow script written in Cuneiform. It enumerates the different data processing steps and gives a coarse overview of task dependencies. Each of these tasks integrates a different command-line tool which is wrapped in Cuneiform and called like a function. The workflow is extensible in the way that each intermediate result can serve as the input to a new sub-workflow and tools can be differently parametrized and, if file formats are standardized, exchanged with different tools.

As a first step to test the scalability of SAASFEE for large-scale use cases which are relevant for biobanks, we ran this variant calling pipeline on 10 GB of compressed whole genome sequencing reads from the 1000 Genomes Project. These reads were aligned against a reference genome, variants were called, and the resulting sets of variants were annotated using publicly available databases. Figure 7 shows the scalability of this workflow. Within the limits of the setup chosen, linear scaling behavior could be achieved for the variant calling workflow.

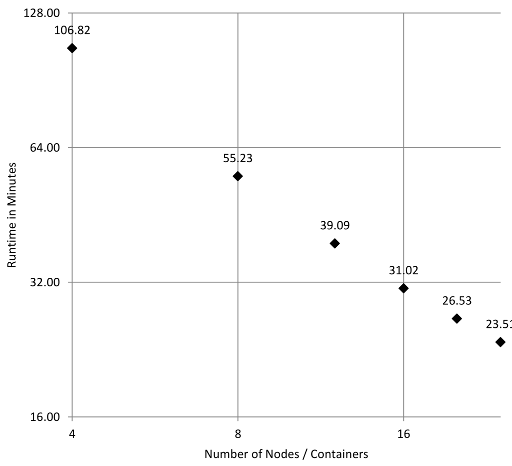


Fig. 7: Scalability experiment for the SAASFEE software stack. A variant calling workflow has been scaled out on up to 24 nodes. Both axes, the runtime in minutes and the number of nodes are on a logarithmic scale (published in Brandt et al. 2015 [22]).

7 Workflows Implemented in BiobankCloud

Next generation sequencing, a technology first introduced to the market in 2005, has revolutionized biological research during the last ten years [27]. This technology allows scientist to study not only the genome of any species, but also the transcriptome and the epigenome. We have implemented several pipelines for the analysis of the most common types of next generation sequencing data into the BiobankCloud.

Variant Calling pipeline: Genetic variations represent changes in the order of the bases in a DNA sequence and variations in the human genome play a decisive role in human disease. The most common variations in the genome are the single nucleotide variations and short insertions and deletions. For this pipeline, whole genome sequencing and/or whole exome sequencing data can be chosen as input data and the workflow was derived from Thalheim [28]. Figure 8 shows a schematic overview on this Variant Calling pipeline built in BiobankCloud.

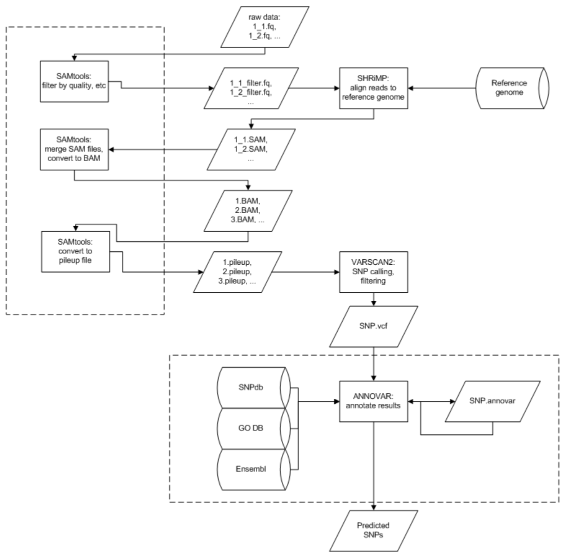

raw data: 1_1.fq, 1_2.fq, ...
SAMtools: filter by quality, etc
1_1_filter.fq, 1_2_filter.fq, ...
SHRiMP: align reads to reference genome
Reference genome
1_1.SAM, 1_2.SAM, ...
SAMtools: merge SAM files, convert to BAM
1.BAM, 2.BAM, 3.BAM, ...
SAMtools: convert to pileup file
1.pileup, 2.pileup, 3.pileup, ...
VARSCAN2: SNP calling, filtering
SNP.vcf
SNPdb
GO DB
Ensembl
ANNOVAR: annotate results
SNP.annovar
Predicted SNPs

Fig. 8: Subsequent processing steps in BiobankCloud's Variant Calling pipeline.

Gene Expression pipeline: This pipeline uses RNA-Seq data as input and enables the user to study differential expression on different levels such as genes and transcripts. Furthermore, it detects differential splicing and promoter use between two or more groups of interest. The pipeline was implemented according to Trapnell et al. [29, 30].

ChIP-Seq pipeline: ChIP-Seq (Chromatin immunoprecipitation coupled with high-throughput sequencing) is the standard technique for studying the genome-wide binding profiles of DNA-binding proteins, e.g. transcription factors, as well as the distribution of histone modifications. ChIP-Seq NGS data are the input data for this pipeline and the workflow is described by Dimitrova et al. [31] was used.

microRNA pipeline: microRNAs are short expressed RNAs that play a key role in many biological processes and in different diseases. Our pipeline for analysis of the expression profiles of microRNAs is based on the publication of Kozubek et al. [32] and is using small RNA-Seq data as input.

8 Sharing Data Between Clusters with Charon

An innovative aspect of the BiobankCloud PaaS is the capability to interconnect several PaaS deployments in a self-managed federation [33], which enables data sharing between biobanks and other BiobankCloud users. These federations can also include public storage clouds, allowing biobanks to extrapolate the capacity of their private resources to the virtually infinite capacity of clouds without endangering the individuals' privacy. These capabilities are implemented through a novel distributed storage system called Charon.

Charon is a cloud-backed file system capable of storing and sharing big data in a secure and reliable way using multiple cloud providers and storage repositories. It is secure and reliable because it does not require trust on any single provider, and it supports the storage of different data sets in distinct locations to comply with required privacy premises and regulations. Three types of data locations are supported in Charon: cloud-of-clouds, single (public) storage cloud, and private repository (e.g., a private cloud). Furthermore, the private repository can be a disk in the client's machine running the Charon file system client, a disk in a remote machine (e.g., a private data center), or any other distributed storage system (e.g., HopsFS).

Private repositories have variable dependability and are subject to local infrastructure restrictions. Charon transfers data from one private repository to another securely through encrypted channels. The single-cloud scenario allows the controlled data sharing between two biobanks, but its dependability directly depends on a single entity – the chosen public cloud provider. The multi-cloud (or cloud-of-clouds) data replication is a possible location to store data, and is where Charon stores all its metadata securely. It avoids having any cloud service provider as a single point of failure, operating correctly even if a fraction of the providers are unavailable or misbehave (due to bugs, intrusions, or even malicious insiders) [34]. Independently from the chosen data location, the data stored in Charon can be shared among mutually-trusted system users.

Figure 9 illustrates a deployment scenario where two biobanks store their data in diverse storage locations—private repositories, single public cloud providers, and a resilient cloud-of-clouds. In this scenario, the namespace tree has six nodes: directories `d1` and `d2`, and files `A`, `B`, `C`, and `D`. The namespace is maintained in the cloud-of-clouds, together with file `B`. File `D`, less critical, is kept in a single public cloud. File `A` is stored locally because it cannot leave biobank 2 (e.g., due to legal constraints). File `C` is shared between the two sites (e.g., in the same legal framework), thus being stored in both of them.

Two distinguishing features of Charon are its *serverless design* and its *efficient management of large files*. The former means that no client-managed server is required in the cloud for Charon, which incurs in reduced maintenance costs. The latter is achieved by employing state-of-the-art techniques for data management such as: working with data blocks instead of whole files, prefetching data blocks, multi-level cache, data compression, and background writes.

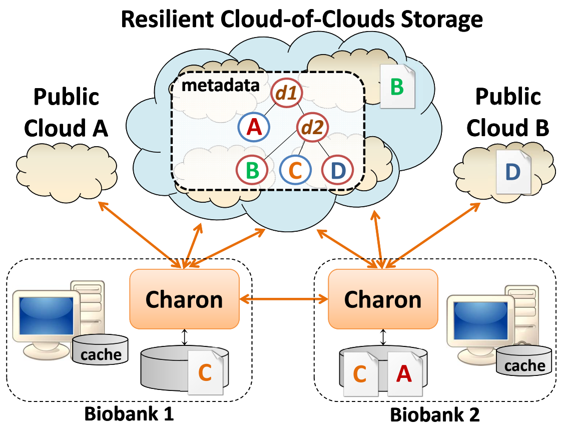


Fig. 9: Charon overview.

9 Automated Installation

BiobankCloud supports automated installation. It can easily be installed by non-technical users who can click-through an installation using only a file that defines a BiobankCloud cluster and account credentials for a cloud computing platform. Our solution is based on the configuration management platform Chef [35]. The main reason we adopted Chef is that it provides support for both upgrading long-lived stateful software and parametrized installations. This contrasts with container-based approaches, such as Docker, that are not yet suitable for online upgrading of stateful systems and also have limited support for parametrization and orchestration. Chef has, however, no support for orchestrating installations. For distributed systems with many services, such as BiobankCloud, there is often a need to start and initialize services in a well-defined order, that is, to orchestrate the installation and starting of services. To this end, we developed an orchestration engine for Chef called Karamel.

In Chef, the basic unit of installation is a recipe, a program written in a domain-specific language for Ruby. In keeping with best practice, in BiobankCloud, each recipe corresponds to a single distributed system service. Recipes are grouped together into software artifacts called cookbooks. We have written Chef cookbooks and services for all our software components. Our Hadoop platform stores its single cookbook in a repository on GitHub called *hops-hadoop-chef*. For each Hadoop service, our hops cookbook has a recipe that both installs and starts it. The recipes include the NameNode (hops::nn), the DataNode (hops::dn), ResourceManager (hops::rm), and NodeManager (hops::nm). Hops also uses a database called NDB, with its cookbook stored in ndb-chef. Similarly, our LIMS (hopsworks) and SAASFEE (hiway) have their own repositories. Karamel augments cookbooks with orchestration rules for recipes defined in a file called *Karamelfile*. The Karamelfile defines what services need to be running within the cluster (and locally) before a given recipe is executed.

Provided the set of all recipes that need to be installed on all nodes, as well as the orchestration rules for the cookbooks, Karamel can take a declarative cluster definition and execute it to install the BiobankCloud platform. In Listing 1.1, we can see a definition of a large BiobankCloud cluster, consisting of 110 nodes. The cluster is defined for Amazon Web Services (*ec2*), but that section of the file can be modified to deploy the same cluster in a different cloud platform, such as OpenStack. The *cookbooks* section defines where the cookbook software artifacts are located. Karamel uses this information to download orchestration rules for the cookbooks and metadata, thus enabling smaller cluster definition files, since they do not need their own orchestration rules. The *attrs* section defines a single parameter for the user that will run the LIMS. In production installations, the *attrs* section typically contains more extensive configuration parameters. Finally the *groups* section defines groups of nodes that install the same software stack. The software stack for each group is defined as a list of recipes. Here, we have four different groups: one for the front-end LIMS (ui), one for the Hadoop and database management services (mgmtnodes), one for the database (datanodes), and one for the storage and processing nodes (workers).

10 Conclusions

In this paper, we introduced the BiobankCloud platform, which provides many features necessary for biobanks to adopt Hadoop-based solutions for managing NGS data. Critical security features that we have introduced for managing sensitive data include multi-tenancy to isolate Studies and 2-factor authentication. Next-generation data management systems for NGS data must be massively scalable. We introduced our scalable storage service, HopsFS, our processing framework, HopsYARN, and our framework for scalable bioinformatics workflows, SAASFEE. We also provide metadata design and search services, while ensuring the integrity of metadata. Finally, in Charon, we showed how we can leverage public clouds to share data securely between clusters. BiobankCloud's secure and scalable open platform software is helping to remove the biobank bottleneck and enabling population-level genomics.

11 Acknowledgements

This work is funded by the EU FP7 project "Scalable, Secure Storage and Analysis of Biobank Data" under Grant Agreement no. 317871.

References

1. M. Janitz, ed., *Next-generation genome sequencing: towards personalized medicine.* John Wiley & Sons, 2011.
2. R. Weissleder and M. Y. Pittet, "Imaging in the era of molecular oncology," *Nature*, vol. 452, no. 7187, 2008.

3. F. F. Costa, “Big data in biomedicine,” *Drug discovery today*, vol. 19, no. 4, 2014.
4. M. Swan, “The quantified self: fundamental disruption in big data science and biological discovery,” *Big Data*, vol. 1, no. 2, 2013.
5. M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, “Spark: Cluster computing with working sets,” *HotCloud*, June 2010.
6. S. Dudoladov, C. Xu, S. Schelter, A. Katsifodimos, S. Ewen, K. Tzoumas, and V. Markl, “Optimistic recovery for iterative dataflows in action,” in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pp. 1439–1443, 2015.
7. M. Bux and U. Leser, “Parallelization in scientific workflow management systems,” *Computing Research Repository*, Mar. 2013.
8. B. Langmead, M. C. Schatz, J. Lin, M. Pop, and S. L. Salzberg, “Searching for SNPs with cloud computing,” *Genome Biol*, vol. 10, no. 11, p. R134, 2009.
9. A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly, *et al.*, “The genome analysis toolkit: a mapreduce framework for analyzing next-generation DNA sequencing data,” *Genome research*, vol. 20, no. 9, pp. 1297–1303, 2010.
10. F. A. Nothaft, M. Massie, T. Danford, Z. Zhang, U. Laserson, C. Yeksigian, J. Kottalam, A. Ahuja, J. Hammerbacher, M. Linderman, M. J. Franklin, A. D. Joseph, and D. A. Patterson, “Rethinking data-intensive science using scalable analytics systems,” in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pp. 631–646, 2015.
11. D. Decap, J. Reumers, C. Herzeel, P. Costanza, and J. Fostier, “Halvade: scalable sequence analysis with MapReduce,” *Bioinformatics*, pp. btv179+, Mar. 2015.
12. L. Pireddu, S. Leo, and G. Zanetti, “Seal,” *Bioinformatics*, vol. 27, pp. 2159–2160, Aug. 2011.
13. A. Schumacher, L. Pireddu, M. Niemenmaa, A. Kallio, E. Korpelainen, G. Zanetti, and K. Heljanko, “SeqPig: simple and scalable scripting for large sequencing data sets in Hadoop,” *Bioinformatics*, vol. 30, no. 1, pp. 119–120, 2014.
14. A. Gholami, J. Dowling, and E. Laure, “A security framework for population-scale genomics analysis,” 2015. The International Conference on High Performance Computing and Simulation.
15. A. Gholami, A.-S. Lind, J. Reichel, J.-E. Litton, A. Edlund, and E. Laure, “Privacy threat modeling for emerging BiobankClouds,” *Procedia Computer Science*, vol. 37, no. 0, pp. 489 – 496, 2014. EUSPN-2014/ICTH 2014.
16. K. Shvachko, H. Kuang, S. Radia, and R. Chansler, “The Hadoop distributed file system,” in *Mass Storage Systems and Technologies, 2010*, pp. 1–10, May 2010.
17. M. Ronström and J. Oreland, “Recovery Principles of MySQL Cluster 5.1,” in *Proc. of VLDB’05*, pp. 1108–1115, VLDB Endowment, 2005.
18. K. Hakimzadeh, H. Peiro Sajjad, and J. Dowling, “Scaling HDFS with a strongly consistent relational model for metadata,” in *Distributed Applications and Interoperable Systems*, pp. 38–51, 2014.
19. S. Niazi, M. Ismail, G. Berthou, and J. Dowling, “Leader election using NewSQL database systems,” in *DAIS*, pp. 158–172, 2015.
20. S. Pabinger, A. Dander, M. Fischer, R. Snajder, M. Sperk, M. Efremova, B. Krabichler, M. R. Speicher, J. Zschocke, and Z. Trajanoski, “A survey of tools for variant analysis of next-generation genome sequencing data,” *Briefings in Bioinformatics*, vol. 15, pp. 256–278, Mar. 2014.

21. M. Bux, J. Brandt, C. Lipka, K. Hakimzadeh, J. Dowling, and U. Leser, "SAAS-FEE: Scalable Scientific Workflow Execution Engine," in *VLDB Demonstrations Track, forthcoming*, (Hawaii, USA), 2015.
22. J. Brandt, M. Bux, and U. Leser, "Cuneiform: A functional language for large scale scientific data analysis," in *Proceedings of the Workshops of the EDBT/ICDT*, vol. 1330, (Brussels, Belgium), pp. 17–26, March 2015.
23. B. Langmead, C. Trapnell, M. Pop, S. L. Salzberg, *et al.*, "Ultrafast and memory-efficient alignment of short dna sequences to the human genome," *Genome biol*, vol. 10, no. 3, p. R25, 2009.
24. L. A. Goff, C. Trapnell, and D. Kelley, "Cummerbund: visualization and exploration of cufflinks high-throughput sequencing data," *R Package Version 2.2*, 2012.
25. E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. F. da Silva, M. Livny, and K. Wenger, "Pegasus: A Workflow Management System for Science Automation," *Future Generation Computer Systems*, vol. 46, pp. 17–35, 2015.
26. J. Goecks, A. Nekrutenko, and J. Taylor, "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences," *Genome Biology*, vol. 11, p. R86, Jan. 2010.
27. J. Shendure and H. Ji, "Next-generation dna sequencing," *Nature biotechnology*, vol. 26, no. 10, pp. 1135–1145, 2008.
28. L. Thalheim, "Point mutation analysis of four human colorectal cancer exomes.," master's thesis, Humboldt Universität zu Berlin, Germany, 2013.
29. C. Trapnell, A. Roberts, L. Goff, G. Pertea, D. Kim, D. R. Kelley, H. Pimentel, S. L. Salzberg, J. L. Rinn, and L. Pachter, "Differential gene and transcript expression analysis of rna-seq experiments with tophat and cufflinks," *Nature protocols*, vol. 7, no. 3, pp. 562–578, 2012.
30. C. Trapnell, D. G. Hendrickson, M. Sauvageau, L. Goff, J. L. Rinn, and L. Pachter, "Differential analysis of gene regulation at transcript resolution with rna-seq," *Nature biotechnology*, vol. 31, no. 1, pp. 46–53, 2013.
31. L. Dimitrova, V. Seitz, J. Hecht, D. Lenze, P. Hansen, M. Szczepanowski, L. Ma, E. Oker, A. Sommerfeld, F. Jundt, *et al.*, "Pax5 overexpression is not enough to reestablish the mature b-cell phenotype in classical hodgkin lymphoma.," *Leukemia*, vol. 28, no. 1, p. 213, 2014.
32. J. Kozubek, Z. Ma, E. Fleming, T. Duggan, R. Wu, D.-G. Shin, and S. S. Dadras, "In-depth characterization of microrna transcriptome in melanoma," *PloS one*, vol. 8, no. 9, 2013.
33. P. E. Verissimo and A. Bessani, "E-biobanking: What have you done to my cell samples?," *IEEE Security and Privacy*, vol. 11, no. 6, 2013.
34. A. Bessani, M. Correia, B. Quaresma, F. Andre, and P. Sousa, "DepSky: Dependable and secure storage in cloud-of-clouds," *ACM Transactions on Storage*, vol. 9, no. 4, 2013.
35. S. Nelson-Smith, *Test-Driven Infrastructure with Chef: Bring Behavior-Driven Development to Infrastructure as Code.* O'Reilly Media, Inc., 2013.

Listing 1.1: Karamel Cluster Definition for BiobankCloud

```
name: BiobankCloud
ec2:
    type: m3.medium
    region: eu-west-1

cookbooks:
  ndb:
    github: "hopshadoop/ndb-chef"
  hops:
    github: "hopshadoop/hops-hadoop-chef"
  hopsworks:
    github: "hopshadoop/hopsworks-chef"
  hiway:
    github: "biobankcloud/hiway-chef"

attrs:
  hopsworks:
    user: bbc

groups:
  ui:
    size: 4
    recipes:
        - hopsworks, hiway::client
  mgmtnodes:
    size: 4
    recipes:
        - hops::nn, hops::rm, ndb::mgmd, ndb::mysqld
  datanodes:
    size: 2
    recipes:
        - ndb::ndbd
  workers:
    size: 100
    recipes:
        - hops::dn, hops::nm, hiway::worker
```