BMOL

BioDec's BioMedical Object Librarian

August 2026

BIODEC

Evolving ICT Infrastructures

Executive Summary

The problem

The management of omics data has some very peculiar characteristics, which make the problem not trivially solvable with traditional IT systems (*file systems*, relational databases, etcetera). The key features is that multiple complex aspects must be managed together, but they have conflicting requirements.

Source heterogeneity Data may originate from external sequencing services: this implies a variability in the data are accessed (it does not necessarily occur via the Internet; physical media such as portable *hard disks* might be used). Other kind of data comes from public datasets as, for example, gnomAD, CADD, ClinVAr, etcetera. All of these kind of data are needed to execute the analysis pipelines.

Data volume Data of *omics* type are often very large (*i.e.* several GB, even tens or hundreds): it is extremely easy to have dedicated file systems that quickly exceed tens of TB of storage.

Computational requirements To analyze omics data often requires substantial computational resources and, in particular, systems capable of handling a highly efficient *I/O* workload, otherwise the analysis execution will slow down: unfortunately, the requirement for high *performance* does not pair well with that of having ample *storage* — indeed, they are conflicting requirements.

Access granularity and *privacy* Last but not least, in case of *human* omics data, there are privacy requirements to comply with, which complicates the access rules, as these must ensure the principle of *privacy-first* and the audit of the data management system.

When other standard aspects of managing a production IT system are added to the above issues (such as the need for *backups* and solutions for *business continuity*), it becomes immediately clear that the problem cannot be solved with a simple architecture (*i.e.* using a standard file system with its access mechanisms).

BMOL

The main idea of BioDec solution is to separate the data management for **consolidation and storage purposes** from the data management **for the purpose of**

computation. The principle may be defined as *archive first* and entails that the omics data life cycle begins with a consolidation phase, which involves classifying and concurrently storing the data, deleting them from the file systems.

First, classify ...

For storage to be effective, it requires metadata that are not compatible with the functionalities of storage systems, whether they are file system based or object storage; furthermore, it is unreasonable to expect omics data to reside as objects within a database (which is the tool of choice for metadata management).

Therefore, the approach involves using a classification system — based on a transactional database — alongside a storage component based on object storage: the traditional file system is removed from the equation as it is sub-optimal both for classification (using *folders* to denote file purpose quickly reveals its limitations) and for efficiency in managing large volumes of data (a fact known to anyone who has had to handle, particularly in virtualized environments, large quantities of files).

Broadly speaking, the process involves invoking a command with the following parameters:

1. an omics data folder, in which the data is organized by *sample*[1],
2. the metadata,
3. other system parameters, such as the addresses of the servers to be used, authentication credentials, etcetera.

...then archive ...

Once the necessary checks have been performed (for example, on user credentials or available space on the object storage), the command communicates with an API server responsible both for interacting with the transactional database where the metadata will be persisted and for copying the omics files to the object storage.

If the action completes successfully[2], the transaction will be considered successful: from that point onward, the metadata will be associated with the URI returned by the object storage, which will become the **unique identifier** for retrieving the file's content.

It will then be in a subsequent phase, when the omics data serve as input for the pipelines, that they will be physically moved to the file system used by the computing environment: note that this operation does not necessarily need to take place during pipeline execution, but could instead be part of a *warm-up* phase wherein the working file system is pre-populated with files required for future calculations. These files may remain available on the computing file system for an extended period if they are needed for that long.

...then search ...

A key benefit of this architecture is that all contents on the computing file systems are effectively only **working copies**. These files will no longer be considered critical data (and thus requiring backup, for example), since if they were lost, they could be reconstructed by retrieving the originals archived in the object storage. This way, several advantages can be achieved:

[1] This is an assumption empirically verified by BioDec's experience in this regard: after all, a sequencing service, while it may not necessarily be aware of the metadata relevant to its client (patient, disease, project data, etc.) certainly and necessarily has its own method for distinguishing different samples.

[2] At this stage, any desired level of redundancy can be added by creating multiple copies across different storage systems.

in the organization of omics data since working *datasets* will be generated by appropriate tools that, by querying the classification system, will allow files to be retrieved according to the logic chosen by the client — and no longer according to the logic permitted by a hierarchy of files and folders on a file system,

in the management of file systems since:

1. one can focus on maintaining smaller and faster working file systems optimized for computing tasks;
2. file permission management is simplified, as complicated permission mechanisms that allow different users to view certain content but not others are unnecessary: working copies can be (taking this to the extreme) strictly personal[3] and duplicated if necessary;
3. files can be subject to periodic cleanup operations, such as removing those belonging to users who are no longer present, with the assurance that the original data remains physically stored elsewhere;

in the management of access since the administrative burden for file security is no longer concentrated on the ACLs of the file systems — apart from for the working file systems, as mentioned above — but on the *capability* of object storage, which were designed with these use cases in mind.

Last but not least, decoupling file management — via the URIs of their storage — from their position (random, at this point) in a file system also simplifies integration with *post-processing* systems that make computation results accessible within enterprise systems.

. . . finally calculate

The results of the pipelines can indeed fruitfully be interpreted as **additional metadata of the initial omics dataset**. BioDec's experience suggests that the quality of managing such information is extremely heterogeneous: with the proposed architecture, it is quite straightforward to consider implementing — similarly to the classification system — a corresponding[4] system that associates metadata of interest with the URIs for the omics data under analysis.

Advantages

Separating the tasks of **classification** and **consolidation** from the tasks of **search** and **collection** allows for managing the data itself much more efficiently. Users may be classified into two classes, based on their roles and responsibilities:

- the archivists,
- the bioinformatics users.

The former are tasked with classifying the omics data in the best possible way (*i.e.* attaching as much available metadata as possible) and verifying its consolidation after receiving or retrieving it from external sources: namely,

[3] *umask 077* as an example on a Linux file system.

[4] Or more than one system, obviously, depending on the different nature of the data generated by the pipelines.

ensuring that the omics data has been persisted on object storage — in the form of a hash of the original data — and that its metadata (including the URI to the object storage) have been recorded in the database.

The latter are tasked with performing searches on the database and subsequently requesting from the archivists that a specific URI collection be made available to them in the form of concrete data (*i.e.* files on a file system or buckets on an object storage).

At that point, it is the archivists' task, based on the data access policy, to verify that the user has the right to access them and, if approved, to transform the URI collection into a set of concrete data (also known as an *access collection*).

The cycle closes with the bioinformatics users, who, if equipped with the appropriate permissions (*i.e.* an access token generated in the previous stage), will be able to access their access collection, operating on the data in the most appropriate way for the computational needs: for example by persisting the entire collection, or only part of it, as omics files or as a set of objects in a bucket.

Responsibilities of the archivists

The archivists are not concerned with what is done with the data; they only care that it is managed in a "secure" manner (i.e., persisted, redundantly stored, and not accessible by default to anyone).

Confidentiality, Integrity, Accessibility

It is their responsibility, once verified that a user has the right to use a specific set of omics data — a verification that depends on organizational processes and whose concrete execution falls outside the scope of the project and the proposed IT solution — to create the collection for them and provide access credentials (via the *features* offered by the particular object storage implementation: from generating a simple access token to simply placing the data in a bucket owned by that user).

From that moment on, their work is complete: determining when the files will be deleted and how to ensure they are not used for inappropriate purposes will become part of the data life cycle on the computational file systems.

Responsibilities of the bioinformatics users

Bioinformatics users are not concerned with how the omics data is managed, nor where it is physically persisted.

It is their responsibility to request the creation of data collections compatible with their computational needs and, once they are, copy them to the correct locations based on the type of analysis to be performed.

It will be their responsibility to ensure that the omics data is accessed only by certain users or groups, or, more generally for human omics data, that only the processing steps they are responsible for are carried out (including, at a certain point, removal, for example).

Confidentiality

In the event of unforeseen circumstances, the computational environment — understood as the input data portion for the pipelines — can always be recreated because the original data resides elsewhere: this freedom allows working on the data with greater peace of mind, as the burden of responsibility for data integrity and accessibility is absent, leaving only that of confidentiality.

FAIR

BMOL follows the *FAIR* principles.

Findability The central component of the proposed architecture is a metadata management engine, which allows organizing omics files in accordance with the rules dictated by the client's needs.

Accessibility The metadata associated with the omics data reside within a PostgreSQL database based on open-source software; this guarantees that there will never be a risk of being unable to retrieve your metadata due to, for example, reliance on third-party proprietary code.

Interoperability Omics files are persisted on systems using the S3 protocol, for which numerous open-source implementations exist, in addition to those offered as a service by numerous *cloud providers*. This ensures a standardized data access interface that is now used in millions of different software projects.

Reusability Finally, the proposed architecture allows for horizontal scaling — by adding new object storage instances and additional classification databases — for specific new use cases, thereby ensuring the system's operational continuity through the reuse of the same components.

Architecture

BioDec's BMOL high-level logical schema is described by figure 1, which shows the main systems (rectangular boxes) and processes (elliptical boxes). The yellow areas represent components already present in the infrastructure, while those inside the box are BMOL main components.

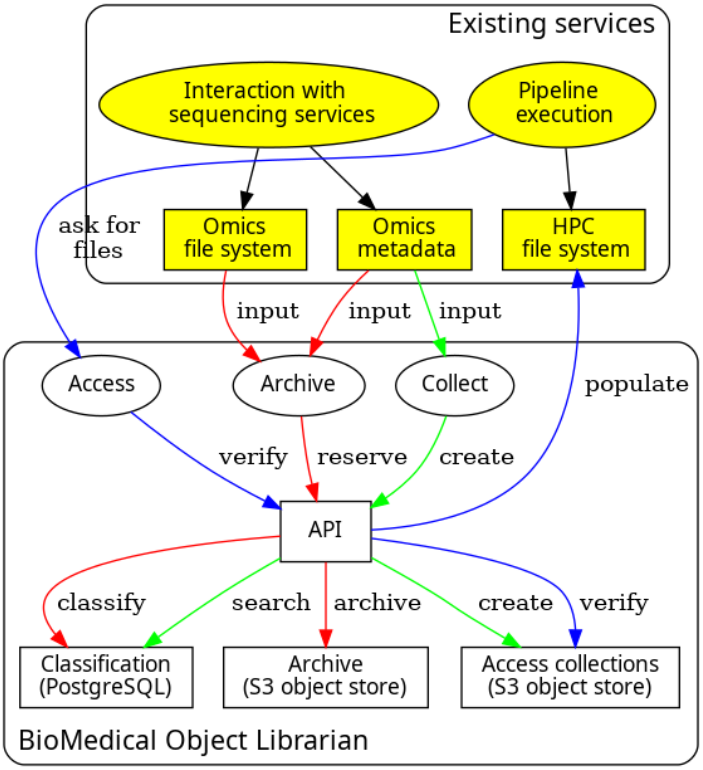

Figure 1: High-level schema of the BMOL system: the ellipses denote the *processes* while the rectangles represent *systems*. Yellow is the color of existing components.

The colored arrows (red, green, and blue) in the figure indicate the workflows of the three main systems:

1. the consolidation system,

2. the collection system,
3. the access system,

which will be detailed in the following sections.

Consolidation system

It constitutes the first pillar of the architecture and comprises the tools and components traversed by the **red** workflow. It is also the first from an operational standpoint, *i.e.*, the one responsible for **the consolidation of omics data**. The tools:

1. must be executed on a dedicated virtual machine, which will host the file system where the omics data from sequencing services will be accessible;
2. will also take as input the metadata necessary for data classification;
3. will interact with the classification and storage components.

Once the appropriate checks have been performed (access rights, storage availability, etcetera), omics data will be inserted into both the storage component and the classification component.

Properties

The consolidation system ensures that:

- the omics data has been persisted (and replicated if necessary),
- the metadata of the omics data has been persisted and it is searchable,
- the link between the metadata and the omics data is provided by the URI of the omics data, as returned by the storage component.

Constraints

Note that omics files in the storage component are not structured for direct use by end users; rather, they are organized within a *cache* structure designed to accommodate data volume growth and manage redundancy mechanisms (replication, *tiering*, etcetera).

For this reason, it is essential that the consolidation process optimizes the use of metadata, as meaningful data collections can only be constructed through searches based on them.

Collection System

It is the second pillar of the architecture and consists of the tools and components traversed by the **green** workflow. From an operational standpoint, it is the workflow that prepares stored data into collections that are easily usable by users who need to access them. The collection tools:

1. can, in principle, be executed on any Linux server,

2. will allow the execution of *queries* for the classification system,

3. will allow the creation of *buckets* on the access component (which will reference the stored objects) corresponding to the query results (for example, omics data belonging to a specific project, or a certain set of samples, or stored within a specific period, etcetera).

Properties

The collection system ensures that new entities are created, which from the user's perspective are objects in the object storage, based on the result of the query that created the collection.

Constraints

Note that the access component, although it is an S3 object storage, must not be used directly but only via the API service, as it will contain only the structure (*scaffolding*) of the omics data, but not their content, which always remains exclusively on the storage component.

The reason why, during query materialization, the contents of omics files are not also replicated in the access component is that it would entail further duplication of omics data copies (with the associated overhead related to storage consumption).

Furthermore, the collection system is a powerful tool that enables access to all data present on the system. Therefore, access to this system must be restricted to users who, due to their role, are tasked with accessing all omics data: by analogy, they are equivalent to system administrators who have *root* access to the file systems of traditional omics data.

Access system

It is the third pillar of the architecture and consists of the tools and components traversed by the **blue** workflow. Operationally, it is the workflow used to obtain a copy on the HPC file system of the omics data corresponding to one or more collections created by the previous workflows. The access tools:

1. can be executed, in principle, on any Linux server,

2. will request the API service to access a specific bucket in order to copy its contents to a specific location on a compute file system.

The API server will verify that the credentials are correct and if so, it will take care of creating a folder on the HPC file system containing the contents of the requested bucket.

Nothing prevents users from subsequently reorganizing the data as they like, according to a different structure suited to their specific needs.

Properties

The access system simplifies the management of read access control for omics data, as it delegates to the object storage the verification of authorization to access a specific collection.

Constraints

As indicated in the previous section, the access component's data should not be used directly, but only as a means to resolve references to the actual data.

Other systems

There will also be other systems, such as the API service for managing access to the infrastructure components referenced in the previous sections, or search tools, which may serve to perform queries but without the ability to materialize collections — for example, to verify the existence of certain data, or to provide lists of URIs that a user with the necessary permissions can then materialize into a collection.

The following is a concise list of the tools and components that make up the systems described previously:

Exchange component It is the space where omics data from external sources or internal sequencing services is consolidated.

Classification component It is based on the PostgreSQL relational database and the APIs to access it.

Storage component It is based on an implementation of an S3 object storage, such as Minio or GarageHQ, with its replication or redundancy mechanisms.

Access collections component It is based on an implementation of an S3 object storage, like the previous one, but also includes the APIs to access it, since data access is mediated.

API server It is the software that will be used to implement API services.

Classification and storage tools These are the commands to be used during the omics data consolidation phase.

Collection tools These are the commands to be used during the materialization phase of omics data collections.

Access tools These are the commands to be used, either interactively or by incorporating them into compute pipelines, to copy the contents of one or more collections onto compute file systems.

Integration tools These are the commands to be used for integration with existing systems.

Metadata

A very important issue is how metadata are classified during the consolidation phase, as this determines how they will be used to formulate queries and, ultimately, to generate access collections. For this reason, it is appropriate to describe the structure of the metadata in detail.

They are distinguished into *mandatory* data and *optional* data. A second subdivision is between *omics metadata*, which depend on the nature of the omics data — thus determined by the sample type, the specific omics data (FASTQ, VCF, etcetera) and other related information (patient data, disease data, etcetera) — *privacy metadata* and *system metadata*.

The only mandatory omics data are the *SampleID* — because it is likely the only field that is always present, as even the sequencing service requires a way to identify the omics data, and the only information it always has is indeed the sample identifier — and the *DataType*, which is partly implicit in the name or file format.

All other data serve to later identify a specific set of samples in a structured manner[5] and consequently generate the corresponding collection.

Omics metadata

Omics metadata can be logically grouped as follows:

- sample data,
- data pertaining to the client to whom the omics data belongs (*customer data*): they could also be identifiers for internal customers, *i.e.* other units of the organization, for whom data are analyzed as a service,
- data pertaining to the projects to which the omics data belong (*project data*) — clearly, as with the previous data, this information may or may not be relevant, and may also need to be described in a more structured manner than proposed here,
- data pertaining to the patient from whom the sample was collected (*patient data*),
- data pertaining to the data life cycle, in case it is necessary to specify an expiration date after which the data must be removed, or sequencing-related data, etcetera (*life cycle data*).

[5] Through a query related to a project, client, disease, etcetera

Sample data

Sample data are the only ones that have a mandatory component.

SampleID Note that the SampleID also identifies the location of the omics datum in the *bucket* of the collections, otherwise all data would be flattened.

For example, if you want to retrieve the *fastq1* and *fastq2* files from sample *alpha* and the *fastq3* file from sample *beta*, it is more practical for the collection to have the following structure

```
bucket
        +-- alpha
        |   +-- fastq1
        |   \-- fastq2
        \ beta
            \-- fastq3
```

Rather than

```
bucket
        +-- fastq1
        +-- fastq2
        \-- fastq3
```

but reconstructing the first alternative is possible only if each of the values *alpha/fastq1*, *alpha/fastq2*, and *beta/fastq3* is present in the metadata of each respective omics datum.

This becomes even more critical since each omics datum will be stored within the component with a non-*human-readable* URI, similar to the following

`https://a2b3.s3.example.com/archive/a2/b3/a2b3569a0ebf`

and therefore, in the absence of metadata, reconstructing what a given omics datum corresponds to would be impossible.

SampleType Genetic type, RNA, DNA, Methylation, etcetera. Or: mRNA, mirna, ncRNA, WES, WGS, etcetera.

DataType FASTQ, BAM, VCF, count, csv, tsv, img, generic, etcetera.

AgeAtSampling Age at sampling: this is an important data point for certain types of analysis where knowing the patient's age at the time of tissue collection is required.

Origin Blood, TissueName, etcetera.

Customer data

List of CustomerID Optional list of customer identifiers.

Project data

List of ProjectID Optional list of project, sub project identifiers, etcetera.

Patient data

If tracked, a list of patient identifiers can be added to the omics data, for example to track familial genetic relationships.

PatientID Patient identifier.

MotherID Mother identifier.

FatherID Father identifier.

DateOfBirth Date of birth.

List of (DiseaseID, DiseaseDataBaseID, AgeAtOnset) Optional list of diseases and their onset dates. Diseases are classified using a pair of identifiers: one for the reference database and one for the identifier within that database.

Life cycle data

These are the data that enable managing the data life cycle with respect to interactions with external entities.

DeliveryDate Delivery date of the omics data.

Delivered If the data must be transferred to an external entity, this field indicates whether the event has occurred or not.

ExpirationDate Expiration date of the omics data — the system does not track what should happen, as this is too dependent on specific aspects of the data, project, or timing; it only marks that the data should no longer be accessible after a given date.

RunID Reference sequencing run for the sample: a run can contain samples from multiple projects or clients and may be removed once the last contained sample has been delivered.

Therefore, a search that returns all samples from a specific run and indicates whether they have all been delivered allows determining if the data from that run can be deleted.

Privacy metadata

These are the data that enable access management to the data in compliance with privacy regulations.

ProcessingID For human omics data, bioinformatics analyses constitute a form of processing (*data processing*), so there must be a way to record consent for such processing and the assigned personnel.

System metadata

System metadata comprises all the data that can be inferred at the time of insertion, such as:

- the *filenames* that are inserted,
- the *URIs* used to store the data,
- the *timestamp* of the event,
- the *username* of the person running the command,
- the *hostname* on which the command is executed.

The URIs of the storage component will have a specific structure to enable solution scalability. If we assume an URI such as the following

```
https://hostname/bucket/path/filename
```

the definition of the terms is as follows.

hostname The S3 service hostname includes a variable name part, corresponding to the first four characters of the hash of the omics file: in this way it will be possible, as the content of the object storage grows, to perform *sharding* transparently (*i.e.* for example by moving data from *00/00* to *07/FF* on a new server, to which correspond the names `[0000-07FF].s3.example.com`).

bucket It will be a bucket with a standard name, for example *archive*.

path It is a pair of folders, corresponding to the first two and the next two characters of the SHA256 hash of the file[6].

filename It is the SHA256 hash of the file.

Therefore, in the following example, if we assume that there is a file named *fastq1* with an SHA256 hash value equal to `a2b3569a0ebf`[7], then the URI of the file *fastq1* is:

```
https://a2b3.s3.example.com/archive/a2/b3/a2b3569a0ebf
```

where the hostname corresponds to `a2b3.s3.example.com`, the bucket to `archive`, the path to `/a2/b3` and finally the filename to `a2b3569a0ebf`.

[6] As Git does.

[7] In fact, an SHA-256 hash contains 64 characters; for readability, a shorter, readable value is shown in the example.

Glossary

Annotation is the operation of adding information to the variant.

ACID *Atomicity, Consistency, Integrity, Duration* are the properties of a transactional database.

API *Application Programming Interface*, the programming interface of an application.

BCL is the file format that contains the *Binary Base Call* and is generated by sequencing systems.

CRUD *Create, Read, Update, Delete*, are the fundamental operations on data.

DPIA *Data Processing Impact Assessment* is the document required by Article 35 of the GDPR.

FASTQ is a text format, now the *de facto* standard, for managing biosequences and their associated quality metrics (technically called *quality scores*)[8].

FDW *Foreign Data Wrapper* is an integration technology specific to the PostgreSQL database server.

Filtering is the selection of a group of variants based on certain criteria, also relying on annotations.

GDPR *General Data Protection Regulation* Also known as the European General Regulation 2016/679.

HPO *Human Phenotype Ontology* is an ontology of human phenotypes.

Object storage is a data management approach that persists them as objects (also called *blobs*) rather than as a file hierarchy within a file system.

Pipeline is the term used to designate a series of programs that, often chained together, perform a bioinformatics analysis. The output of a pipeline may be of interest in itself, or serve as the *input* for a subsequent pipeline (in this latter case, the term *sub-pipeline* is also used to refer to the components of a broader pipeline that aggregate them). Pipelines are the computational components of a computational analysis platform.

[8]Cock, P. J. A.; Fields, C. J.; Goto, N.; Heuer, M. L.; Rice, P. M. (2009). . Nucleic Acids Research. 38 (6): 1767–1771. doi:10.1093/nar/gkp1137. PMC 2847217. PMID 20015970.

S3 API is the HTTP(S) access protocol for the corresponding object storage service of *Amazon Web Services*[9]. Numerous alternative implementations of the protocol exist to avoid dependency on a single vendor; for example, it is recommended to use the Minio software[10].

VCF *Variant Call Format* denotes the format for representing variations (variants) in a gene sequence. It is also a text-based format, its specification being public[11], and it has long been the reference for the majority of bioinformatics programs.

VEP *Variant Effect Predictor* is a tool for analyzing the effect that variants (*SNP*, *CNV*, *indel* or structural variations) have on a specific gene, sequence, protein, transcript, or transcription factor. One of the most well-known implementations is that of the ENSEMBL project[12].

[9] https://docs.aws.amazon.com/AmazonS3/latest/API/Type_API_Reference.html

[10] https://min.io/

[11] https://samtools.github.io/hts-specs/VCFv4.3.pdf

[12] https://github.com/Ensembl/ensembl-vep

Contacts

Company BioDec s.r.l.

Address Via Calzavecchio 20/2, Casalecchio di Reno (BO)

VAT number IT02327271207

Phone +39 051 0548263

Contact person Michele Finelli (m@biodec.com)