Spfy: applying a graph-based sequence database for real-time prediction of *E. coli* phenotypes

Kevin K Le*[1], Matthew D Whiteside†[1], James Hopkins[1], Victor PJ Gannon[1] and Chad R Laing[1]

[1]National Microbiology Laboratory at Lethbridge, Public Health Agency of Canada, Twp Rd 9-1, Lethbridge, AB, T1J 3Z4, Canada

January 16, 2018

Abstract

Current comparative computational workflows chain different analysis software, but lack storage and retrieval methods for generated results. Spfy uses graph data structures to store and retrieve results for computational workflows, facilitating the management and querying of tens of thousands of whole-genome *E. coli* sequences, and efficient downstream processing. By making the storage and retrieval of results part of the platform, with data effectively linked to the organisms of interest through a standardized ontology, we can mitigate the recomputing of analyses. Within Spfy, the output from all analyses is stored, and linked together in the context of a genome graph. This graph also stores metadata for each genome, facilitating inquiries ranging from population genomics to epidemiological investigations. Integrated data storage will be necessary as publicly available whole genome sequencing data for bacterial pathogens currently numbers in the tens of thousands, with hundreds of thousands set to be available within the next few years.

The database is available at `https://lfz.corefacility.ca/superphy/spfy/`.

1 Introduction

Whole genome sequencing (WGS) can in theory provide the entire genetic content of an organism. This unparalleled resolution and sensitivity has recently transformed public-health surveillance and outbreak response [**?**, **?**]. Additionally, the identification of novel disease mechanisms [**?**, **?**], and rapid clinical diagnoses and reference lab tests based on the specific mechanism of disease are now possible. [**?**, **?**].

The rapid characterization and comparison of bacterial pathogens relies principally on the combination of outputs from multiple software programs that are targeted for specific applications. Examples include the Resistance Gene Identifier (RGI) [**?**] for antimicrobial resistance (AMR) gene prediction, VirulenceFinder [**?**] for virulence factor (VF) annotation, and Prokka for bacterial genome annotation with external tools [**?**]. In particular, RGI and VirulenceFinder represent a series of *in-silico* methods which have been developed to replicate the results of traditional wet-lab tests; this allows new WGS results to be viewed in the context of historical tests.

Comprehensive platforms that combine individual programs into a cohesive whole also exist. These include free platforms such as the Bacterium Analysis Pipeline (BAP) [**?**], the Integrated Rapid Infectious Disease Analysis (IRIDA) project `http://www.irida.ca/`, and PATRIC [**?**]. Commercial applications, such as Bionumerics, which is used by PulseNet international for the analyses of WGS data in outbreak situations also exist, and offer support as well as accredited, standardized tests [**?**].

*kevin.le@canada.ca
†matthew.whiteside@canada.ca

WGS data for bacterial pathogens of public health importance have recently accumulated in public databases in the tens of thousands, with hundreds of thousands set to be available within the next few years. For *Escherichia coli*, there are over sixty thousand publicly available genomes in EnteroBase `https://enterobase.warwick.ac.uk/` and three million whole genomes in GenBank [**?**].

To begin to solve this "big-data" problem, platforms such as BAP and IRIDA provide hosted solutions that compute results in real-time, and distribute analyses across computing resources. While effective for self-contained workflows, many of the comparative analyses that are run are broadly useful, and therefore computed multiple times. An effective method to mitigate the recomputing of analyses is to make the storage and retrieval of results part of the platform, and effectively linked to the organisms of interest with a standardized ontology. Such measures can help ensure the rapid response times required for public health applications, and allow results to be integrated and progressively updated as new data becomes available.

We have previously developed Superphy [**?**], an online predictive genomics platform targeting *E. coli*. Superphy integrates pre-computed results with domain-specific knowledge to provide real-time exploration of publicly available genomes. While this tool has been useful for the thousands of pre-computed genomes in its database, the current pace of genome sequencing requires real-time predictive genomic analyses of tens-, and soon hundreds-of-thousands of genomes, and the long term storage and referencing of these results, something that the original SuperPhy platform was incapable of.

In this study, we present the Spfy platform, which was built with a graph database at its core. Spfy aims to solve the problem of real-time analysis and merging of results from different analysis methods. By merging results and resolving connections between genome data, Spfy is able to identify relationships between all genomes sequenced in the past, present, and future. This graph-based result storage allows retrospective comparisons as more genomes are sequenced or populations change, and is flexible to accommodate new analysis methods as they are developed.

By supporting multiple *in-silico* subtyping options, the platform functions similar to a reference laboratory, while adding support for big-data analyses. Subtyping options for E.coli are O-antigen, H-antigen, Shiga-toxin 1 (Stx1), Shiga-toxin 2 (Stx2), and Intimin typing, VF and AMR annotation, pan-genome generation, group comparisons via Fisher's, and machine-learning (ML) based predictions for OmniLog AMR assays. These tasks are divided into subtasks, and distributed across a built-in task queue. Results are also decomposed into graph structures and stored within a larger graph database. By integrating task distribution with graph storage, Spfy enables large-scale analyses, such as epidemiological associations between specific genotypes, biomarkers, host, source, and other metadata, and statistical significance testing of genome markers for user-defined groups, in real-time. Currently, the platform has been tested with XXX genome files and result storage for XX analyses modules.

Future work will focus on adding additional analyses modules, using machine learning and artificial neural networks to aid genotype to phenotype predictions, and supporting different species. While the integrated approach of storing and retrieving results provides enormous benefits, the developed analyses modules are self-contained and can easily be integrated into existing platforms such as Galaxy [**?**], and IRIDA. The website is available at `https://lfz.corefacility.ca/superphy/spfy/`. Spfy's codebase is provided at `https://github.com/superphy/backend` and a developer guide is provided at `https://superphy.readthedocs.io/en/latest/`.

2 FUNCTIONALITY

Prebuilt tools are available for a variety of reference laboratory tasks. Spfy performs: O-antigen typing, H-antigen typing, and VF gene determination using ECtyper `https://github.com/phac-nml/ecoli_serotyping`, STX typing using Phylotyper [**?**], and AMR gene determination using the RGI program [**?**]. Spfy also performs bioinformatics analyses: pangenome generation using Panseq [**?**], statistical significance testing using SciPy [**?**], and support vector machine (SVM)-backed AMR predictions using Scikit-learn [**?**]. For the larger population comparisons, we store and aggregate the results from individual genome analyses.

Knowledge graphs semantically link data points using relations defined in an ontology. Spfy allows users to compare the results associated with particular genome types or metadata by storing all results in a graph structure. Comparisons can be made in real-time, as new genomes are submitted for analysis and the results stored in the existing graph. Any data type or relation in the graph is valid for analysis, such as the presence or absence of pan-genome regions against determined serotypes or other subtyping options. In addition, Spfy supports the submission of user-specified metadata, such as location or source information. Results are displayed on-screen and available for download.

3 IMPLEMENTATION

Spfy's server-side code is developed in Python and the front-end website is developed using the React JavaScript library. For the addition of new data to the database, the following steps are taken:

i) The upload begins through the ReactJS-based website, where user-defined analyses options are selected. The results of these chosen analyses are immediately reported to the user following their completion, while the remaining analyses are subsequently completed and stored in the database without interaction from the user. The public web service accepts up to 200 MB of genome files (50 genomes uncompressed, or 120 genomes compressed) at a time, though an unlimited amount of data can be submitted locally.

ii) User-selected analyses are enqueued into the Redis Queue `http://python-rq.org/` task queue. Redis Queue consists of a Redis Database `https://redis.io/` and task queue workers which run as Python processes.

iii) The workers dequeue the analyses, run them in parallel, and temporarily store results in the Redis database.

iv) Python functions parse the results and permanently store them in Blazegraph `https://www.blazegraph.com/`, the graph database used for Superphy.

3.1 Data Storage

Semantic web technologies describe the relations between different data [**?**]. For biological data, individual data points such as genome, contiguous DNA sequence, or gene, can be linked together in a query-able graph structure. This allows novel data to be seamlessly incorporated into the existing graph, and has the use of graph databases for relational information has been proposed as a common standard for the open sharing of data [**?**].

In Spfy, the results from all the analysis modules are converted into graph objects, which are used to update the main graph database. These graph objects, which consist of nodes and edges, must be annotated with common terminology. To avoid proliferating ontologies, and to allow Spfy to integrate with existing ones, annotations from the GenEpiO [**?**], FALDO [**?**], and TypOn [**?**] ontologies are used to describe biological data. The use of semantic web technologies allows results to be linked to the genomes they were computed from, and for all data points to share a common data structure that can be queried together.

The permanent storage of results is as a one-time cost, and allows population-wide analyses of all stored genomes; result storage also enables Spfy to avoid recomputation when the same analysis is re-run. To identify analysis tasks, Spfy fingerprints requests by computing its SHA1 hash and tags the results in the graph database with this hash. Subsequent queries are then compared through fingerprinting, and results can be rebuilt from the graph by searching for matching fingerprints.

For population analyses, Spfy searches the graph for all nodes annotated with the queried ontology term. Depending on the given analysis, this data is then vectorized into NumPy numerical arrays for use with the SciPy scientific computing library, or the Scikit-learn machine learning library.

3.2 Web design

The front-end website is written using the React JavaScript library `https://facebook.github.io/react/` as a single-page application to allow efficient data-flow without reloading the website. To ensure a familiar user interface, we followed the Material Design specification `https://material.io/`, published by Google, surrounding a card-based design. (see Figure 2) Both the task selection and result displays follow this card-based design: while data storage is graph-based, the results of various analysis modules are presented to users in a familiar tabular structure and available for download as .csv spreadsheet files. (see Figure 3)

3.3 Service Virtualization

Docker `https://www.docker.com/` is a virtualization technology to simulate self-contained operating systems on the same host computer, without the overhead of full hardware virtualization [**?**]. The Spfy platform depends on a series of webservers, databases, and task workers and uses Docker to compartmentalize these services which are then networked together using Docker-Compose `https://docs.docker.com/compose/`. (see Figure 4) Docker integration ensures that software dependencies, which are typically manually installed [**?**, **?**, **?**, **?**], are instead handled automatically. With compartmentalization of service runtimes, code failures do not propagate to other services.

3.4 Continuous integration and testing

TravisCI `travisci.io`, a continuous integration (CI) platform, is used to ensure that Spfy `https://github.com/superphy/backend` does not break with any changes to the codebase, and runs tests for functionality and backwards compatibility. The individual tests use PyTest `https://doc.pytest.org/`, and are run within TravisCI's virtual environment, and the current build status can be checked either on our Gtihub repository or at `https://travis-ci.org/superphy/backend`. CI is also used to automatically build Spfy's core Docker images, and upload them to Docker Hub `https://hub.docker.com/u/superphy/`.

4 RESULTS

Spfy was tested with 4,622 public *E. coli* assembled genomes from Enterobase, storing every sequence and results for all included analysis modules. Spfy provides real-time subtyping, and the results are immediately displayed to the user following their completion. Subtyping options include O-antigen, H-antigen, Shiga-toxin 1, Shiga-toxin 2, and Intimin typing. Reference-lab tests include virulence factor and anti-microbial resistance annotation. All genomes are analyzed within the pan-genome framework of *E. coli*, and results from all analyses are automatically associated with the source genome. The resulting database had 1,333 nodes and 683,666 leaves, with 374,836,872 object properties.

Spfy has been up since May 2017. The server accepts assembled *E. coli* genomes with the *.fasta* or *.fna* extensions. Submissions are checked against a reference set of *E. coli* gene sequences before running analyses. Outputs are displayed on the website in tables and can be downloaded as *.csv* files.

5 DISCUSSION

Many previous bioinformatics software programs have been developed *ad hoc*, with individual researchers and laboratories developing software specific to their environment [**?**]. Such tools were often script-based, with custom data formats, and only suitable for small collections of data [**?**]. While this was acceptable for smaller analyses, bioinformatic pipelines utilizing WGS data are larger and involve linked dependencies, which require the application of systems engineering principles [**?**]. Additionally, many subsets of biology now require the analyses of big-data, where the ability to perform computations in real-time, store data in flexible databases, and utilize a common application programming interface (API) linking resources are required [**?**].

One of the key goals in developing Spfy is to maintain instantaneity, as modern websites have accustomed users to immediate results. We attempt to use innovations in web development to bring a similar experience to Spfy as a predictive genomics platform for *E. coli*. The use of Docker containerization, task queues, and a graph database, were all chosen to support instantaneity; modern analytics platforms must respond to user-requested analyses of big-data in the same efficiency as old analyses of single files.

5.1 Impact on Public Health Efforts

The isolation and characterization of bacterial pathogens are critical for Public Health laboratories to rapidly respond to outbreaks, and to effectively monitor known and emerging pathogens through surveillance programs. Until recently, public-health agencies relied on laboratory tests such as serotyping, pulsed-field gel electrophoresis (PFGE) banding patterns, or Polymerase Chain Reaction (PCR)-based amplification to identify known VFs or AMR genes, along with other tests [**?**], to characterize bacterial isolates in outbreak and surveillance settings.

AMR testing, VF testing, and Shiga-toxin testing are of particular importance to surveillance efforts due to their role in a pathogen's lethality. Current efforts are focused on predictive genomics, where the relevant phenotypic information can be determined through examination of the whole-genome sequence. WGS methods allow a single laboratory method - the sequencing step - to provide a variety of analysis options, such as AMR, VF, and Shiga-toxin results, and as such can be used to evaluate the spread of outbreaks with better resolution and context than traditional methods [**?**].

Spfy uses WGS results. After initial sequencing of new isolates, Spfy can be used in place of a traditional reference laboratory, to determine the O-type and H-type, Stx type, and all known VFs and AMR genes in real-time. These results can be shared with other agencies and researchers over the internet. Furthermore, using Spfy's database of pre-processed genomes, Spfy can determine all strains a sample may be related to which is useful for monitoring the evolution of pathogens over time. A computational approach saves the time and cost associated with performing multiple tests per sample, and allows population comparisons to keep pace the current generation WGS data.

5.2 Comparison with other bioinformatic pipeline technologies

Existing scientific workflow technologies such as Galaxy [**?**], and pipelines such as the Bacterium Analysis Pipeline (BAP) [**?**] and the Integrated Rapid Infectious Disease Analysis (IRIDA) platform `http://www.irida.ca/` help automate the use of WGS data for public-health surveillance. Galaxy aims to provide a reproducible, computation-based research environment which is accessible to individuals without programming knowledge. Galaxy tackles the problem of linking different analysis software together, by defining interdependencies using a custom schema. A visual workflow editor is also provided for ease of use. IRIDA is build on top of the Galaxy framework, and adds prebuilt pipelines specific to bioinformatics uses, as well as sequence and result storage. IRIDA takes a project-based approach, with sequences stored per project, and results stored linearly per sequence. IRIDA adds Controls for collaborating on projects, and uses common terms found in the GenEpiO ontology to describe results.

The Bacterial Analysis Platform (BAP), developed out of the Technical University of Denmark, provides an integrated analysis pipeline for bacterial WGS data as a web service. BAP takes a combined approach to genome analysis including different programs, such as for VF and AMR determination, by default into the pipeline. A record of the different files generated by an analysis are stored in a relational database, and the record is queried to determine which files are returned to the user [**?**].

Like IRIDA and BAP, Spfy automates workflows for users, and like Galaxy, Spfy uses task queues to distribute selected analysis. File uploads begin through the ReactJS-based website, where user-defined analyses options are selected. To these concepts we add Docker containerization for task queue workers, allowing analysis software to safely run in parallel. For result storage, existing workflow technologies use relational tables [**?**], or store resulting files to disk [**?**]. Because output from these programs is user-specific or transitory, results from identical comparisons are often recomputed. Additionally, output from different analyses are structured using distinct terminology and formats, which must be converted before they can be compared. Without a unified structure, these conversions quickly become impractical for broad usage. Graph-based storage of all results solves these problems. The entire platform is packaged using Docker-Compose, and can be recreated with a simple command.

On a per file basis, Spfy performs at a similar speed to BAP on predictive genomics tasks, though Spfy does not provide genome assembly services. Spfy also processes XXXX files over XX tasks in XXXX time by using a novel approach of distributing computations over a task queue and multiple Docker compartmentalized containers. In place of a relational database for storing the location of result files, Spfy parses result data into graph objects and integrates results into a persistent graph database. This enables Spfy to create large datasets where analysis results are linked to genomes, and to perform population-wide analyses. In all, Spfy provides similar functionality with an expanded focus on integrated result storage and big-data analyses.

To approach the challenge of integrating different bioinformatics programs, Spfy instead uses technologies prevalent in common web services not necessarily related to scientific workflows. While we prefer to package individual bioinformatics programs using language-agnostic package managers such as Conda `https://conda.io/`, we do agree that suites of bioinformatics software benefit from having all dependencies installed and deployable in Docker containers [**?**]. These programs are connected to form a workflow using task queues; a design methodology used to offload long-running or non-instantaneous tasks over available computing infrastructure. Unlike platforms such as Galaxy which favor a drag-and-drop design for building workflows, Spfy favors a tightly coupled approach explicitly linking together different bioinformatics programs within the task queue implementation. In particular, Spfy uses asynchronous tasks queues which are tripped in response to user requests, thereby allowing immediate responsivity to users on the website, and processes generating human-readable results can be separated from processes handling long-term result storage.

One of the key benefits of using more common-place technologies is the compatibility with other infrastructure resources. Docker containerization is widely supported by cloud computing services: Amazon Web Services (AWS) `https://aws.amazon.com/docker/`, Google Cloud Platform (GCloud) `https://cloud.google.com/container-engine/`, and Microsoft Azure `https://azure.microsoft.com/en-us/services/container-service/`, and self-hosted cloud computing technologies such as OpenStack `https://wiki.openstack.org/wiki/Docker` all support Docker. Spfy packages compute nodes, which manage a collection of task queue workers, as reproducible Docker containers which can be networked together to form a compute cluster. Docker containerization has a negligible impact on performance [**?**], and allows the platform to easily scale to demand. In contrast to Galaxy, Spfy is a targeted approach building a predictive analytics platform for *E. coli* which leverages generic web technologies to favor a responsive, big-data design strategy tackling the challenge [**?**] of integrating results from multiple genomes and examining shared connections. By packaging the individual bioinformatics software as Conda packages, they can be easily ported to Galaxy.

6 CONCLUSIONS

7 ACKNOWLEDGEMENTS

7.0.1 Conflict of interest statement.

None declared.

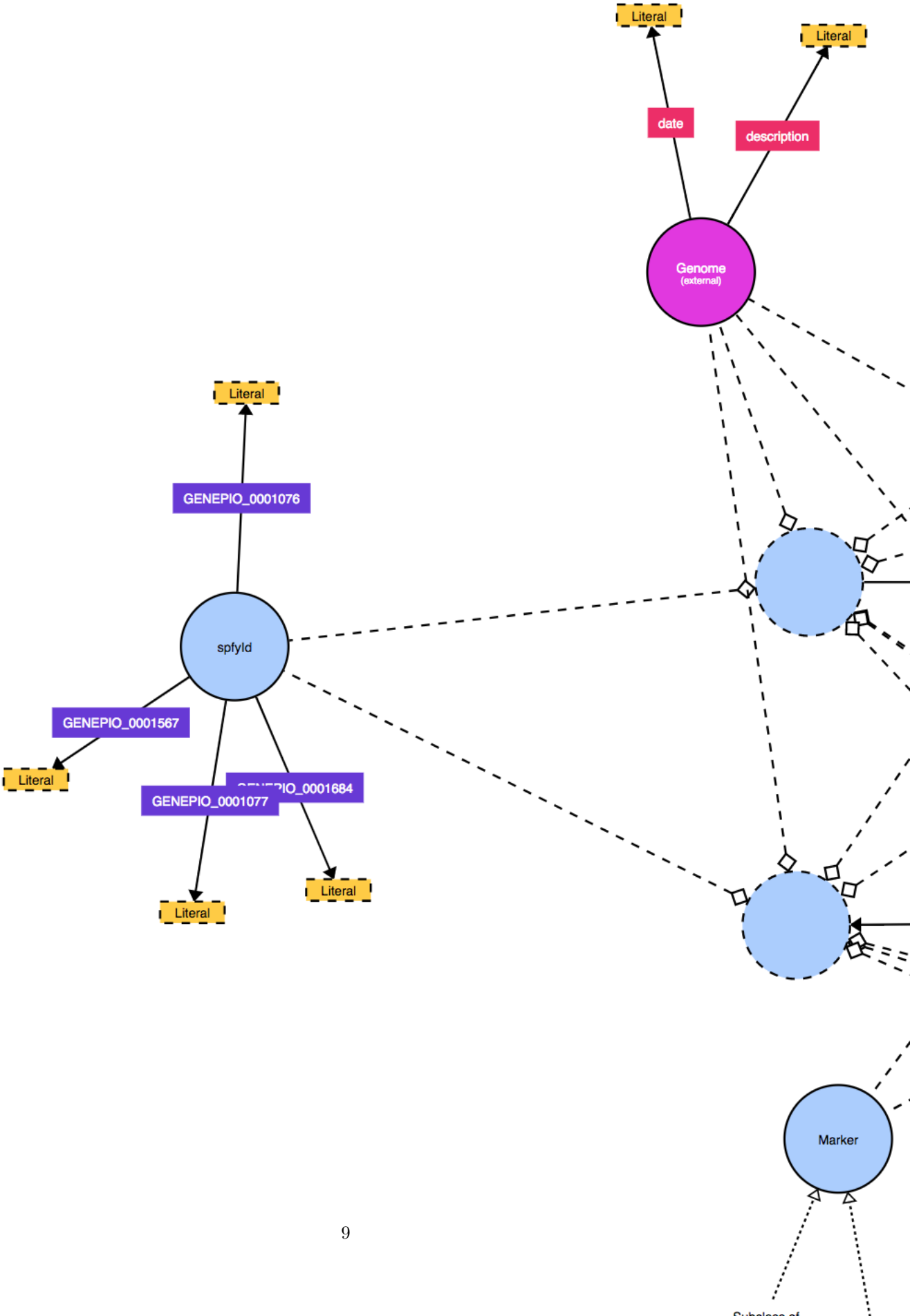
References

[1] Author,A.B. and Author,C. (1992) Article title. *Abbreviated Journal Name*, **5**, 300–330.

[2] Author,D., Author,E.F. and Author,G. (1995) *Book Title*. Publisher Name, Publisher Address.

[3] Author,H. and Author,I. (2005) Chapter title. In Editor,A. and Editor,B. (eds), *Book Title*, Publisher Name, Publisher Address, pp. 60–80.

[4] Author,Y. and Author,Z. (2002) Article title. *Abbreviated Journal Name*, **53**, 500–520.


Literal
Literal
date
description
Genome
(external)
Literal
GENEPIO_0001076
spfyId
GENEPIO_0001567
Literal
GENEPIO_0001077
GENEPIO_0001684
Literal
Literal
Marker

spfy

Tasks

Results

da

Database status as of: 11:43:34 A
Submitted: 11:43:34 AM, Status

JobId: fdef2756-ef67-4366-975b-16bafdc9a

SEE RESULT

Su

GCA_001911775.1_ASM191177v1
Submitted: 11:43:42 AM, Status

JobId: blob7151249537150029571

SEE RESULT

fi

O157 vs O169 for
https://www.github.com/superphy
Submitted: 11:44:22 AM, Status

JobId: 79382759-b2b5-4a2f-b3e6-5adede6e

SEE RESULT

Su

10 Files with pi: 90 for Serotype VI
Submitted: 11:44:45 AM, Status

JobId: blob4433578772837076418

spfy

Tasks

Results

Search

Export to CSV

Filename

Please enter a value

GCA_001911825.1_ASM191182v1_genomic

GCA_001911825.1_ASM191182v1_genomic

GCA_001911825.1_ASM191182v1_genomic

GCA_001911825.1_ASM191182v1_genomic

GCA_001911825.1_ASM191182v1_genomic

GCA_001911825.1_ASM191182v1_genomic

GCA_001911825.1_ASM191182v1_genomic

GCA_001911825.1_ASM191182v1_genomic

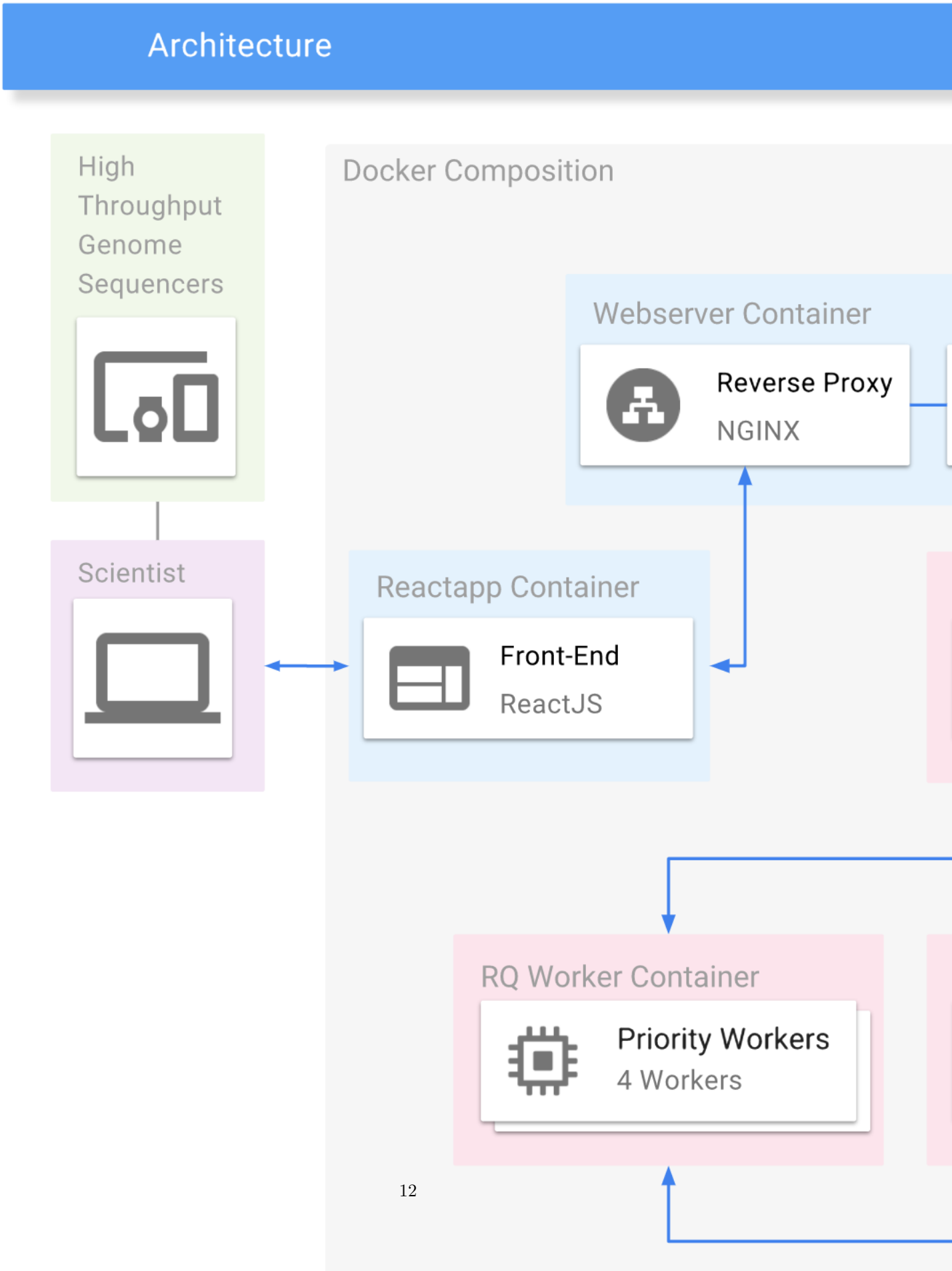
GCA_001911825.1_ASM191182v1_genomic

GCA_001911825.1_ASM191182v1_genomic

GCA_001911825.1_ASM191182v1_genomic

GCA_001911825.1_ASM191182v1_genomic

GCA_001911825.1_ASM191182v1_genomic


Architecture
High Throughput Genome Sequencers
Docker Composition
Webserver Container
Reverse Proxy
NGINX
Scientist
Reactapp Container
Front-End
ReactJS
RQ Worker Container
Priority Workers
4 Workers