Design of a Generic Architecture for executing Bioinformatics Workflows on Distributed Infrastructures

Abel Carrión, Ignacio Blanquer, Miguel Caballer, Cristina Y. González, and Ignacio Medina

Instituto de Instrumentación para Imagen Molecular (I3M).
Centro mixto CSIC - Universitat Politècnica de València - CIEMAT
Camino de Vera s/n, 46022 Valencia, Spain
{abcarcol,micafer1,iblanquer}@upv.es
Computational Genomics Institute of Centro de Investigación Príncipe Felipe
Autopista del Saler 16, 46012, Valencia, Spain
{cgonzalez,imedina}@cipf.es

Abstract. The exponential increase of biological data as a result of improvements on Next Generation Sequencers has revealed the need of powerful hardware that can cope with it. Despite the development of several tools for dealing with this kind of experiments, the main problem of almost all of them is the lack of scalability. In order to address it, this article exposes the design of Bio-WINGS (Bioinformatics Workflows in Next Generation Sequencing), a powerful system that will support a lot of bioinformatics applications over different flavours of distributed computing infrastructures.

Keywords: Bioinformatics, Workflows, Distributed Computing, Cloud Computing

1 Introduction

Many biological processes have a computational nature and computational and statistics models can be useful. Thus, computation has become an essential tool in life science research, such as Genomics (microarrays and massively DNA sequencing) where complex tools are required. This fact generated an informatics crisis for life science researchers due to the inherent difficulty found when using computational resources. For that purpose, software libraries such as Bioconductor [1] and Bioperl [2] were developed, improving greatly the *accessibility* to computation. Despite that, in order to use these libraries, life science researchers still required programming knowledge, being a clear bottleneck for scientists that have a life science background and no programming experience. For instance, Bioconductor demands knowledge of the R programming language. Moreover, in the case of massively DNA sequencing, the volume of data related to the study of biologic sequences has increased exponentially due not only to

better sequencing techniques but also to the reduction of costs. So, these so called NGS (Next Generation Sequencing) experiments have exceedingly large dataset sizes that pose new challenges. In the last years several tools have been developed for executing these kind of experiments on distributed infrastructures. However, one of the main problems of many of these tools is the lack of scalability.
To address this, Bio-WINGS (Bioinformatics Workflows in Next Generation Sequencing) has been designed. Bio-WINGS is a generic architecture for executing Bioinformatics WorkFlows (BWFs) on Distributed infrastructures. BWFs are the process that a bioinfomatician must perform to transform raw data to publishable results. Among other features, Bio-WINGS will simplify the access to a wide spectrum of powerful computer resources by providing a familiar graphical-based environment or GUI (Graphical User Interface) for inexperienced users and it will be also very extensible. The architecture proposed has been inspired by three typical bioinformatics scenarios: the detection of mutations, the assembly of sequences and phylogenetic studies.
The paper's main contributions are to describe:

1. The platform architecture with 4-Tier Architectural Style, where the four layers are: GUI (Graphical User Interface) Layer, Core Engine Layer, Infrastructure Layer and Data Layer. The Infrastructure and Data layers have been designed to be as independent as possible of any specific infrastructure, allowing Bio-WINGS to run on any computing back-end.
2. The procedure for solving the three use cases mentioned before with the architecture proposed will be exposed.

This paper is structured as follows. First, Section 2 offers an exhaustive revision of the state-of-the-art. Next, Section 3 describes all the details regarding the generic architecture, including an overview and a description of each layer. Section 4 shows how three scenarios that inspired this architecture are addressed using the architecture. Finally, Section 5 ends with the most remarkable conclusions extracted during the design of this work and the future steps.

2 Related work

Workflow modelling for data-intensive scientific applications is a topic already covered in the literature. In the beginning, several initiatives for creating workflows to be applied to Grid deployments appeared. These contributions can be broken into two groups: those that are standalone and those that have web-based interfaces. The most relevant standalone projects are described in the next paragraphs.
The Kepler project [3] provides a workflow system derived from the Ptolomey II. In this project a workflow system is modelled as a composition of actors (that are independent) through well-defined interfaces. The extensibility of this system is given by the actor-oriented architecture, adding new actors. The system has a set of actors for performing typical Grid operations (authentication, file

copy, job execution, job monitoring, service discovery, etc.). Nevertheless, the user needs to know a lot of details about the Grid resources.
Taverna [4] is the workflow management system of the myGrid project. It is based on a definition language named Simple Conceptual Unified Flow Language (SCUFL). Taverna provides data models, enactor task executions, and graphical user interfaces.
Chipster [5] offers a wide collection of data analysis methods within the reach of bioscientists via an intuitive graphical user interface. The analysis options are complemented with interactive visualizations and the possibility of saving workflows, which can be shared with other users.
With respect to the web-based user interfaces tools, there is one particularly remarkable: Galaxy [6]. Galaxy is an open web-based platform for genomic research, that makes computation *accessible*, ensures that all analysis are *reproducible* and allows the *transparency* via the sharing of experimental results between users. A Galaxy instance can utilize compute clusters for running jobs, and can be easily interfaced with PBS (Portable Batch System) and SGE (Sun Grid Engine).
In the last years, a new distributed computing paradigm, Cloud Computing, has emerged. Although Cloud Computing is of increasing interest in the computing industry, it has the potential to revolutionize e-science by giving scientists the computational resources they need, when they need them. Thus, Cloud Computing is somewhat different from Grid computing, which is more focused on integrating heterogeneous resources from multiple 'virtual organizations'. In that sense, a project named e-Science Central (e-SC) [7] has developed a cloud-based Science Platform that allows scientists to store, analyse and share data in the cloud. e-SC can be deployed on both private (e.g. Eucalyptus [8]) and public Clouds (Amazon AWS [9] and Microsoft Windows Azure [10]]).
The difference between Bio-WINGS and the solutions cited above is the completeness. Bio-WINGS aims to offer in a single tool all the features highlighted before, mainly: an intuitive GUI for inexperienced users and support to any kind of application or computational back-end (Cluster, Grid and Cloud).

3 Architecture proposal

The Bio-WINGS architecture has been designed taking into account three principles: *generality*, *extensibility* and *modularity*. Generality refers to the possibility of executing a wide range of bioinformatics applications and supported by any kind of distributed computing infrastructure. Extensibility, enables a user to add a new application or a new computing back-end. Last but not least, the modularity of the architecture allows that new elements can be added without the need to change other parts of the system.

3.1 Overview

As shown in Figure 1, the architecture follows a 4-layer architectural style, where the four layers are: UI Layer, Core Engine Layer, Infrastructure Layer and Data

Layer. Next subsections describe all the details regarding each layer and their interaction with external elements and between them.

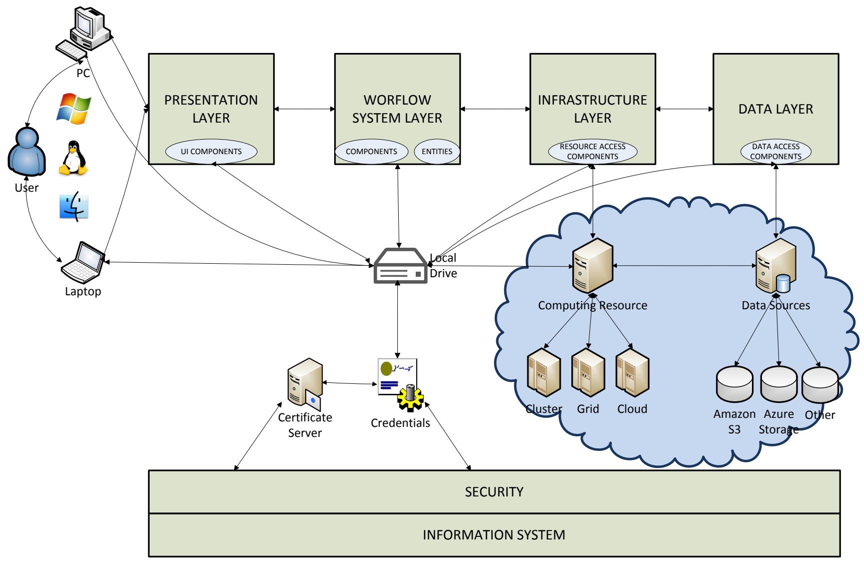

Fig. 1. Bio-WINGS architecture

3.2 User Interface Layer

The presentation to the Bio-WINGS system is offered via the UI(User Interface) Layer. This layer can be implemented using two approaches: a web-interface that can be accessed using any kind of web browser or a stand-alone application. In this second case, because bioinformaticians use different Operative Systems, a key aspect is to develop the application with a multi-platform programming language, such as Java. As it can be seen in Figure 1, this layer is composed by a set of UI components. Although, at the moment of the writing of this article the UI look has not been yet defined, it should contain the following: a section for designing the workflow, a tool bar with the bioinformatics tools available, a section with information about the experiments submitted by the user, etc.

3.3 Workflow Engine (WINGS)

The core engine of the architecture is a workflow system called WINGS (Workflow In New-generation GRIDs). In previous works [11], WINGS has been described . However, briefly, it is a workflow engine focused on multi-grid capabilities and easy extensibility. Every element is implemented as a plug-in, so new elements (execution environments, operations, information systems, file transferrers) can be added without the need of modifying the other parts of the system. The system has a data flow orientation: when the data is ready the execution starts, enabling implicit parallelism. In that sense, it is aligned with BWFs (Bioinformatics Worflows) which are mostly data flow oriented. Although ,initially, WINGS was designed as an alternative workflow system for grid infrastructures, thanks to its extensibility this work will adapt it to other computing back-ends, such as Cloud Computing. As Figure 1 depicts, this layer interacts with other elements of the architecture. The first interaction takes place with the local drive of the user computer, getting the necessary credentials for using the infrastructure selected by the user. Once the credentials are verified, the workflow is analysed and if a task can be executed (the input data is available according to the data-flow mechanism) it must be passed down to the underlying layer as well as a reference to the input data.

3.4 Infrastructure Layer

As its name suggest, the infrastructure layer is the part of the architecture in charge of contacting the requested computing resource by the user. The *extensibility* property of the architecture allows adding support to any kind of distributed computing back-ends: Cluster, Grid or Cloud. The aim of this work is to support the most relevant exponents of each of these three categories. In fact, there is already support to various systems due to the use of WINGS as core engine system. In that sense, the architecture heritages support to clusters that use PBS/Torque and grids with the middlewares Fura and gLite [12].
This layer receives from the upper layer three elements: the user credentials, the tasks to be run and a reference to the input data. The procedure of this layer is as follows: the reference to the input data is passed to the innermost layer of the architecture, the Data Layer. When the Data Layer confirms that the input data is ready, the Infrastructure Layer submits the tasks to the computing resource along with the user credentials to authenticate the user. All the information regarding the status of the jobs (instances of the tasks) are passed up to the Workflow System layer to keep the user informed about the evolution of the execution. This procedure is repeated every time that a step of the workflow is ready to be processed. Moreover, the user will be capable defining breakpoints in the execution of the workflow to examine the data obtained at a certain point.

3.5 Data Layer

The innermost layer is the one responsible for staging the files needed by the tasks to be executed in the computer resources. Analogous to the case of the Infrastructure Layer, the Data Layer also benefits from the *extensibility* property of the architecture, making feasible the use of any data storage, given that the necessary plug-in has been developed.

Before the execution of the tasks in the computing resources, the Infrastructure Layer communicates the reference to the input data of the task to the Data Layer. Then, the actions performed by the Data Layer depends on the size of the input data. If the input data can be considered "Big Data", the computing resource will be allocated in the nearest location to the data resource. In other case, the data will be transferred to the near most storage resource to the computing resources that are going to be used. In both cases the idea is to improve the efficiency on the data transference. Finally, when the result of a workflow step is available, the Data Layer stages-out the data, moving the result from the data source to the local drive.

3.6 Cross-cutting

The are two global features in the architecture: the security and the information system between layers. Because, in a bioinformatics environment the privacy of the data managed is a crucial issue, security is of most importance. In order to deal with this, the communication between all the layers will be done via SSL (Secure Sockets Layer) and the data will be encrypted and decrypted accordingly. The authentication of the user in a computing resource or data source will be done in most cases by means of certificates. Another aspect is the information system, which is in charge of providing the status of the resources and the jobs to the layers that request it.

4 Use cases

In order to illustrate the possibilities of the generic architecture described in the previous point, this section presents three uses cases of special relevance in the bioinformatics field (the detection of mutations, the assembly of sequences and phylogenetic studies) that have inspired this work. After introducing each use case, the corresponding workflow, ready to be input in the GUI Layer of Bio-WINGS, will be explained.

4.1 Mutation analysis

Mutation Analysis is a hot topic in bioinformatics research. Sequencing individuals' genome as a routine test is becoming economically feasible due to the reduction of sequencing costs. Complete genome sequencing is especially interesting in rare diseases and oncology. However, the process requires multiple processing steps to reduce errors, identify variability and search if the mutation has been already associated to a disease in existing databases. However, despite of the popularity of the topic, there are few complete workbenches for mutation analysis. An example is Variant [13] from the Centre of Research Prince Philippe.

Once the sequencing has been completed, the mutation analysis goes through mainly three steps (see workflow of Figure 2). First errors that may have been

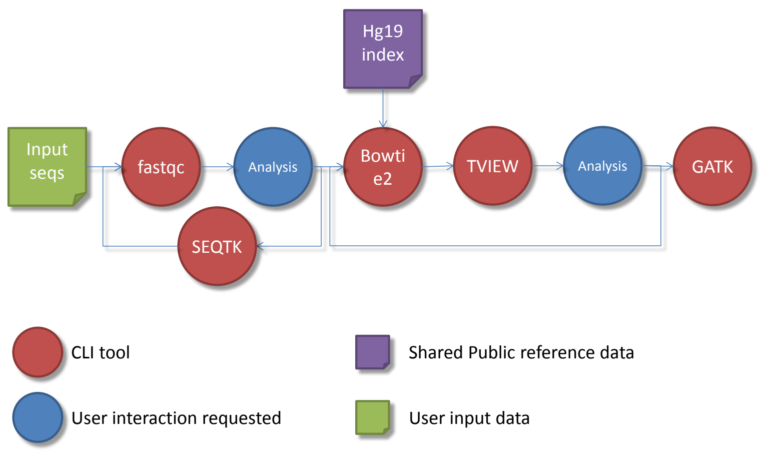

Fig. 2. Mutation analysis workflow

producing during the sequencing process need to be reduced. It is very important to analyse the quality of the input data, remove suspicious sequences, trim low-quality parts and have a global assessment of the quality of the sequences based on global statistical analysis. For this purpose, FastQC [14] is used.
Second stage deals with the alignment. Alignment consists on searching if the sequences obtained map to a reference consensus genome and where they do. It is important to understand that each individual has a different genome, so the reference used is a consensus of what could be a representative genome for a human. Therefore, alignment is a complex and computing intensive process that depends on the size of the reference database, the size of the input set of sequences, and the tolerance in the matchings, among other fators. There are many tools used in alignment, but in the case of NGS (Next Generation Sequencing), hash- or suffix array-based approaches are preferred. Bowtie2 [15] manage to align millions of sequences to a reference genome in hours or minutes, even assuming a few sequencing errors per sequence. This stage also requires information from databases such as GenBank, UNIPROT, genomes1000K. This stage ends up with exact and partial matches, providing the location where similar sequences can be found in the reference. Variations may indicate that a mutation is produced.
Last step is to characterise the differences found. Many of these differences may be caused due to human variability, and could be produced in areas of the genome with low or no activation, therefore not having a known effect on hu-

man?s health. However, other differences may have been already identified as particularly frequent in ill patients. Moreover, differences may be caused due to undetected sequencing errors, polymophisms or other causes. Therefore, a statistical analysis is required before searching on the literature and databases. A procedure called Variant Call is typically applied followed by variant annotation, by using tools such as GAKT [16]. Databases with annotated mutations are publicly available, such as dbSNP, ENSEMBL, HGMD, OpenGWAS, COSMIC and OMIM.

4.2 *De novo* sequence assembly

Over the last decade, DNA sequencing machines have evolved from slow, but accurate machines to massively parallel sequencing platforms capable of producing shorter fragments, termed "reads", of much greater redundancy. This change in paradigm has prompted computational biologists to develop new algorithms and new software to handle the assembly of these short reads into full genomic data. However, this problem of piecing together sequenced reads into a full genome, known as *de novo* assembly, is similar to putting together a jigsaw puzzle with millions to billions of very small pieces.

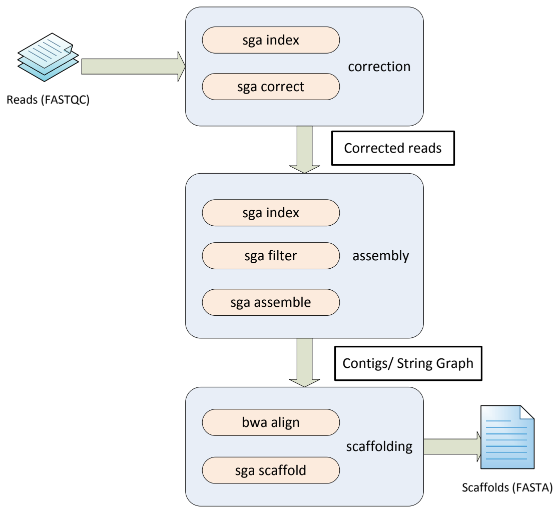

Fig. 3. Assembly workflow

As the Figure 3 represents, the output of an assembly is typically a set of *contigs*, which are contiguous sequence fragments, ordered and oriented into *scaffold* sequences, with gaps between contigs within scaffolds representing regions of uncertainty. The workflow showed uses an open source assembler called SGA(String Graph Assembler). The problem lies in the fact that most software today cannot cope with the vast amounts of data provided by DNA sequencers without the use of powerful hardware, such as the distributed computing platforms supported by Bio-WINGS.

4.3 Phylogenetics

Phylogenetics is the science of classification of organisms. It studies the evolution of a genetically related collection of things (genes, proteins, organisms) that are derived from a common ancestor. Thanks to the phylogenetics it is possible to find evolutionary ties between organisms, analyse changes occurring in different organisms during evolution, understand relationships between an ancestral sequence and its descendants and estimate the time of divergence between a group of organisms that share a common ancestor.

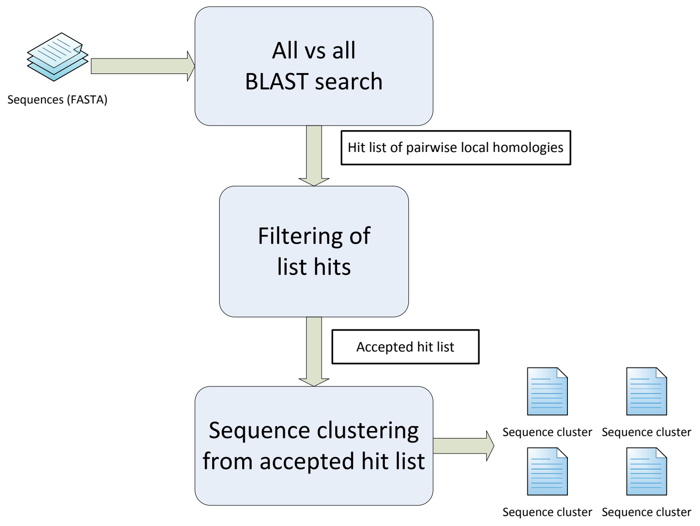

Fig. 4. Phylogenetics workflow

Next, an explanation of the pipeline depicted in Figure 4 is offered. The purpose of the clustering pipeline is to assemble sets of sequences together that

have at least local homologies (i.e., matching or nearly matching sub-sequences). These can form the basis for the construction of individual phylogenetic data sets.
The input of the pipeline is a FASTA file with a bunch of sequences and the first step consists on performing an all-against-all search between those sequences. Every sequence is queried against every other using BLAST [17] to identify all local homologies ("hits") between every pair of sequences. The result is a BLAST report that contains a hit list of pairwise local homologies where each entry is query and target sequence with a resulting hit of certain length, etc. Next, the hits can be filtered in a variety of ways. For example, for phylogenetic purposes it is ideal to have sequences of nearly the same length so that alignment programs work well and there is little missing data. Thus, the list of hits can be filtered to exclude small regions of local homology. Once a final list of hits is obtained, a set of clusters of sequences is built. Presently, the filtering keeps only hits that have greater than 51% coverage in both directions (at a stringent BLAST e-value cutoff of -10). Then the filtered hit list is turned into a set of clusters via "single-linkage clustering". To be a member of such a cluster, a sequence merely has to have a hit with any other member of the cluster. Even at this stage, stricter, smaller, clusters can be obtained by other clustering methods, such as "complete linkage clustering". The output of the pipeline is thus a set of clusters, each of which contains one or more sequences.

5 Conclusions and Future Work

Currently, Bioinformatics is a field that suffers an informatics crisis due to the vast amount of data to be analyzed. In order to handle this issue, several tools can be found in the literature that address this problem. However, all of them lack any of the following crucial features: an intuitive interface for people (bioinformaticians) who normally don't have programming skills, the availability of a wide set of bioinformatics tools and the support of several kinds of computing distributed infrastructures that offer scalability. For that reason, this article has detailed the design of a generic architecture for addressing the new bioinformatics challenges, Bio-WINGS. This architecture follows a 4-layer style and its design principles are: generality, extensibility and modularity. A key aspect is the plugin-based design that allows adding new functionality to any component without changing the rest of the parts of the system. The architecture is intended to support scenarios such as mutation analysis, phylogenetics studies, sequence assembly, etc. as well as different distributed computing infrastructures (Grids, Clouds). The current working line is completing a working prototype developed with existing technologies.

Acknowledgments. The authors wish to thank the financial support received from The Spanish Ministry of Science and Innovation to develop the project Servicios avanzados para el despliegue y contextualizacion de aplicaciones virtualizadas para dar soporte a modelos de programacin en entornos cloud, with reference TIN2010-17804.

References

1. Robert C. Gentleman, Vincent J. Carey, Douglas M. Bates, Ben Bolstad, Marcel Dettling, Sandrine Dudoit, Byron Ellis, Laurent Gautier, Yongchao Ge, Jeff Gentry, Kurt Hornik, Torsten Hothorn, Wolfgang Huber, Stefano Iacus, Rafael Irizarry, Friedrich Leisch, Cheng Li, Martin Maechler, Anthony J. Rossini, Gunther Sawitzki, Colin Smith, Gordon Smyth, Luke Tierney, Jean Y. Yang, and Jianhua Zhang. Bioconductor: open software development for computational biology and bioinformatics. *Genome biology*, 5:1–16, 2004.
2. Jason E. Stajich, David Block, Kris Boulez, Steven E. Brenner, Stephen A. Chervitz, Chris Dagdigian, Georg Fuellen, James G. Gilbert, Ian Korf, Hilmar Lapp, Heikki Lehväslaiho, Chad Matsalla, Chris J. Mungall, Brian I. Osborne, Matthew R. Pocock, Peter Schattner, Martin Senger, Lincoln D. Stein, Elia Stupka, Mark D. Wilkinson, and Ewan Birney. The Bioperl toolkit: Perl modules for the life sciences. *Genome research*, 12:1611–1618, 2002.
3. Scientific workflow management and the Kepler system. *Concurrency Computat.: Pract. Exper.*, 18:1039–1065, 2006.
4. Tom Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver, Kevin Glover, Matthew R. Pocock, Anil Wipat, and Peter Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20:3045–3054, 2004.
5. M. Aleksi Kallio, Jarno Tuimala, Taavi Hupponen, Petri Klemela, Massimiliano Gentile, Ilari Scheinin, Mikko Koski, Janne Kaki, and Eija Korpelainen. Chipster: user-friendly analysis software for microarray and other high-throughput data. *BMC Genomics*, 12:507+, 2011.
6. Jeremy Goecks, Anton Nekrutenko, James Taylor, and The Galaxy Team. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, 11:R86+, 2010.
7. S. Woodman H. Hiden, P. Watson and D. Leahy. e-Science Central: Cloud-based e-Science and its application to chemical property modelling. 2011.
8. Eucalyptus: Open source aws compatible private clouds.
9. F. Miller, A. Vandome, and J. McBrewster. Amazon Web Services. 2010.
10. Windows azure, http://www.windowsazure.com/.
11. C. de Alfonso, M. Caballer, and V. Hernandez. WINGS: Versatile Workflow for the Grid. *Advanced Engineering Computing and Applications in Sciences, 2008. ADVCOMP '08. The Second International Conference on*, pages 51–56, 2008.
12. P. Andreetto, S. Andreozzi, G. Avellino, S. Beco, A. Cavallini, M. Cecchi, V. Ciaschini, A. Dorise, F. Giacomini, A. Gianelle, U. Grandinetti, A. Guarise, A. Krop, R. Lops, A. Maraschini, V. Martelli, M. Marzolla, M. Mezzadri, E. Molinari, S. Monforte, F. Pacini, M. Pappalardo, A. Parrini, G. Patania, L. Petronzio, R. Piro, M. Porciani, F. Prelz, D. Rebatto, E. Ronchieri, M. Sgaravatto, V. Venturi, and L. Zangrando. The gLite workload management system. *J. Phys.: Conf. Ser.*, 119:062007+, 2008.
13. Ignacio Medina, Alejandro De Maria, Marta Bleda, Francisco Salavert, Roberto Alonso, Cristina Y. Gonzalez, and Joaquin Dopazo. VARIANT: Command Line, Web service and Web interface for fast and accurate functional characterization of

variants found by Next-Generation Sequencing. *Nucleic acids research*, 40:W54–W58, 2012.

14. Fastqc, http://www.bioinformatics.babraham.ac.uk/projects/fastqc.
15. Ben Langmead and Steven L. Salzberg. Fast gapped-read alignment with Bowtie 2. *Nat Meth*, 9:357–359, 2012.
16. Aaron McKenna, Matthew Hanna, Eric Banks, Andrey Sivachenko, Kristian Cibulskis, Andrew Kernytsky, Kiran Garimella, David Altshuler, Stacey Gabriel, Mark Daly, and Mark A. DePristo. The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Research*, 20:1297–1303, 2010.
17. S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215:403–410, 1990.