A Cloud-based GWAS Analysis Pipeline for Clinical Researchers

Paul Heinzlreiter[1,2], James Richard Perkins[3], Oscar Torreño[5,1], Johan Karlsson[5,6], Juan Antonio Ranea[5], Andreas Mitterecker[7,1], Miguel Blanca[4] and Oswaldo Trelles[5,1]

[1]*RISC Software GmbH, Softwarepark 35, 4232 Hagenberg, Austria*

[2]*Leibniz Supercomputing Centre (LRZ), Boltzmannstr. 1, 85748 Garching, Germany*

[3]*Research Laboratory, University Hospital-IBIMA, Málaga, Spain*

[4]*Allergy Unit, University Hospital-IBIMA, Málaga, Spain*

[5]*Department of Computer Architecture, University of Málaga, Málaga, Spain*

[6]*Integromics S.L., Avenida de la Innovación 1, 18100 Armilla, Granada, Spain*

[7]*Institute of Bioinformatics, Joh. Kepler University Linz, Linz, Austria*

Keywords: Cloud Computing, Bioinformatics, Biomedicine.

Abstract: The cost of obtaining genome-scale biomedical data continues to drop rapidly, with many hospitals and universities being able to produce large amounts of data. Managing and analysing such ever-growing datasets is becoming a crucial issue. Cloud computing presents a good solution to this problem due to its flexibility in obtaining computational resources. However, it is essential to allow end-users with no experience to take advantage of the cloud computing model of elastic resource provisioning. This paper presents a workflow that allows the end-user to perform the core steps of a genome wide association analysis where raw gene-expression data is quality assessed. A number of steps in this process are computationally intensive and vary greatly depending on the size of the study, from a few samples to a few thousand. Therefore cloud computing provides an ideal solution to this problem by enabling scalability due to elastic resource provisioning. The key contributions of this paper are a real world application of cloud computing addressing a critical problem in biomedicine through parallelization of the appropriate parts of the workflow as well as enabling the end-user to concentrate on data analysis and biological interpretation of results by taking care of the computational aspects.

1 INTRODUCTION

This paper presents *genCloud*, a workflow allowing the end-user to employ various state of the art tools for the analysis of genome wide association studies (GWAS). A workflow describes a series of activities as part of a process. These activities are typically linked and information is transferred from one activity to another. Such links represent dependencies (for example, one activity must occur before another). Currently, running these GWAS applications requires computational infrastructure and knowledge of specific tools, including the command line, scripting, and the R computational language. These tools often require an amount of computational resources greater than what is typically available in a moderate size hospital, which is a typical environment where such an analysis takes place. The workflow described here presents a contribution to the areas of cloud computing and biomedicine allowing end-users with basic computational knowledge, such as clinical scientists, to analyse their own experiments. This is possible because the workflow is made available to end-users through the jORCA software client (Martin-Requena et al., 2010). By presenting a case study of such an analysis, using real life data we apply cloud computing to solve an important biomedical problem in a real-world situation. Our biomedical use case is particularly well suited to cloud computing because it can require very different amounts of computational resources depending on the size of the analysis.

In terms of the underlying cloud infrastructure the work presented in this paper offers several distinguishing characteristics as compared to standard cloud installations:

- User authentication based on grid technology.
- Specific support for transferring big datasets

Heinzlreiter P., Perkins J., Torreño Ó., Karlsson J., Ranea J., Mitterecker A., Blanca M. and Trelles O..
A Cloud-based GWAS Analysis Pipeline for Clinical Researchers.
DOI: 10.5220/0004802103870394
In *Proceedings of the 4th International Conference on Cloud Computing and Services Science* (CLOSER-2014), pages 387-394
ISBN: 978-989-758-019-2

through grid computing protocols.

- Enactment of specific biomedical and bioinformatics workflows through an user-friendly software client, which significantly lowers the level of computational experience required by the end-users.

1.1 Applying Cloud Computing for Workflows

Cloud computing (Mell and Grance, 2011) specifically addresses requirements of the biomedical workflow solution described here:

- *Dynamic Instantiation of Additional Resources through Elasticity:* The capacity needed for running user applications can easily be increased or decreased either automatically through a scheduling component or following a user request without requiring interaction with the service provider.
- *Flexible Configuration of Workflow Modules without Administrator Intervention:* This approach is specifically viable in a scientific environment, where the users often rely on specific libraries and software packages to perform their research. The domains of biomedicine and bioinformatics – which are addressed within the scope of the work described here – are no exception to this rule: For example the statistical software environment R (R Development Core Team, 2008) is heavily used in these research domains.
- *Storing readily Configured Instances for reuse through Snapshotting Mechanisms:* An infrastructure-as-a-service (IaaS) approach is used giving the user the utmost flexibility by providing access to fully configurable virtual instances. Similarly the instances involved in an analysis workflow can be stored as snapshots and later reused, allowing the user to recreate the analysis setup when data sources are updated (for example for new genome releases), or to allow other users to recreate their original analysis. Such reproducibility of published results is a recurrent problem in biomedical research (Button et al., 2013) (Mobley et al., 2013).

1.2 Related Work

Considerable work has been done in the field of bioinformatics in connecting independent web-services (WS) into higher-level workflows solving more complex problems.

Taverna (Oinn et al., 2004) is a widely used tool in bioinformatics for composing web-services into workflows and enacting them. The stand-alone tool allows users to access various web-service repositories and connecting the services graphically. It is important to note that the focus of Taverna is not specific for cloud computing environments but instead it focuses on web-services which could in turn be deployed on cloud computing environments.

Galaxy (Afgan et al., 2010) is an open, web-based platform for biomedical and bioinformatics research. The tool allows users to create and execute workflows either from the web browser or through the API provided. Users can register and use the system on a public galaxy server. Additionally it is possible to deploy the tool easily on top of commodity hardware or in cloud environments such as Amazon EC2 and Openstack (Le Bras and Chilton, 2013). CloudBioLinux (Krampis et al., 2012) and CloudMan (Afgan et al., 2012) allow the researchers to easily and quickly get access to a functional compute infrastructure. This infrastructure can be configured in a matter of minutes and finalized when is no longer required, using the CloudMan tool. The infrastructure provides access to a set of bioinformatics tools, as specified in the CloudBioLinux virtual machines. After the configuration of the compute infrastructure, the Galaxy workflow engine is deployed. With the galaxy tool deployed and the compute infrastructure configured, the end-users could focus on creating analysis workflows and running them on the infrastructure being configured, rather than focussing on specific details of configuration and deployment.

2 WORKFLOW

This section describes the biomedical workflow and its underlying technologies in detail. Starting from the workflow itself, the focus is being put on the underlying cloud technology being applied during its execution, such as data handling and user authentication. While the specific workflow as described in section 2.1 serves as proof of concept, the underlying cloud, data transfer, and authentication solutions as well as the client software presented in section 2.5 can also be used for other computational workflows and are not limited to the bioinformatics and biomedical domains.

2.1 GWAS Workflow

An overview of the bioinformatics workflow is given in Figure 1. The user begins by uploading CEL files to the cloud data storage. Each CEL file contains the raw data generated from a single nucleotide poly-

morphism (SNP) – an area of variation in the human genome – microarray analysis for a single patient containing details of the genotype. The amount of data involved can range from a few megabytes to potentially terabytes in size, depending on the number of patients included in the study. Once the CEL files have been uploaded, the birdseed algorithm (Korn et al., 2008), which is implemented in the Affymetrix Power Tools package, is executed. It produces a table of genotype calls for each of the different probes on the microarray. Each probe represents a different SNP. Next this table is filtered to remove SNPs that do not vary across different patients, and that could not be called accurately by the birdseed algorithm. This is achieved through the use of a Python script that parses the output of the birdseed algorithm. Once this table has been filtered, an additional python script must be called to convert the birdseed output, which is specific for Affymetrix data, into a variant call format (VCF) file, the standard representation of a genotype for biomedical data. This script must be called separately for each CEL file, which is specific for each patient. Therefore it is an embarrassingly parallel problem and as such very suitable for a data-parallel execution on cloud infrastructures. Finally the user can download the set of VCF files and can then apply standard web-based tools to view the data, or further packages for downstream analysis. Given that this is a standard file format for several types of genomic biomedical data, it is the starting point for many well established further workflows. Several of these workflows are currently under development.

The workflow is using already available software for all of its stages. The specific novelty is given by its execution on a community cloud environment, which has been extended in functionality specifically regarding the data transfer between the stages of the workflow.

It should also be noted that the workflow as presented here is designed to analyse data produced using the Affymetrix SNP chip microarray platform. However it is also possible to adapt the workflow to other types of input data by modifying instances of the *genCloud* cloud images or creating new ones as necessary to implement related workflows processing data from different platforms, as well as sequencing data.

Taken together, these raw data analysis workflows and different downstream analysis workflows will provide a flexible, easy to use service enabling a wide variety of potential analysis workflows, for various genomics technologies.

Although the workflow as shown in Figure 1 only contains a small number of steps, it is computationally quite involved and contains the key steps of an initial GWAS data analysis. Also the different steps have varying computational requirements. Therefore we have chosen to implement this workflow as a first step for testing the different tools available and ensuring its validity for more complex analysis.

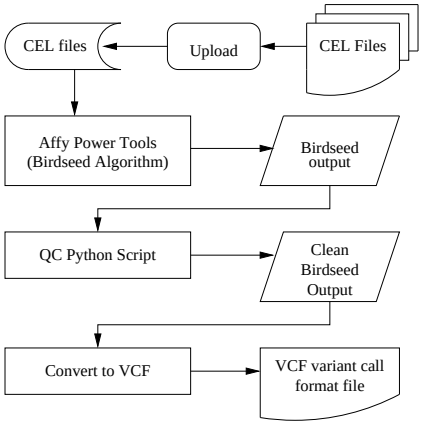

Figure 1: The GWAS workflow.

2.2 Cloud Computing Infrastructure

The cloud computing infrastructure is provided on a small community cloud installation running OpenStack (Pepple, 2011) as cloud middleware. Besides offering IaaS cloud services, as an improvement beyond the state of the art the cloud installation has been extended to support efficient transfer of large data sets through the GridFTP protocol (Allcock et al., 2003). In addition, the data storage is built on top of the distributed file system Ceph (Weil et al., 2006a), which is accessed by OpenStack through the RADOS Gateway (Ceph Team, 2012) implementing the API being used by the OpenStack services. Ceph is made accessible for OpenStack services like object containers and persistent volumes. The object containers can be used as storage for files similar to a directory in a file system. However OpenStack containers cannot be nested. Persistent volumes on the other hand are equivalent to ordinary hard disks and can be dynamically attached to running OpenStack instances.

To glue all of these components together, standard web-service protocols have been used. To enable the transfer of mass data into OpenStack object containers they have been made accessible for the GridFTP data transfer protocol by mounting them into the local file system of a GridFTP server. The same mounting approach is applied for providing access to the containers from inside the running virtual instances.

This underlying infrastructure enables the configuration and execution of arbitrary workflows such as the *genCloud* workflow discussed in this paper.

2.3 Data Storage and Transfer

The Ceph distributed file system (Weil et al., 2006a) is used as the basic data storage system in our cloud setup. Its main advantage is given by the fact that it does not have a single point of failure due to not requiring a specific master node. The storage server is determined by the CRUSH data distribution mechanism (Weil et al., 2006b).

To make the Ceph storage accessible to the OpenStack services, a web-service enabled access mechanism is exposed through the RADOS Gateway, which implements the web-service interfaces defined by the Amazon S3 (Amazon Web Services, 2013) and OpenStack Swift storage services on top of Ceph.

In addition to the web-service based data access mechanisms, we have configured the OpenStack containers to be accessible for the mass data transfer protocol GridFTP. Since a GridFTP server operates on a local file system, the object containers - which are normally accessed through web-service calls - need to be mounted into the local file system of the GridFTP server.

This is achieved by the Cloudfuse daemon, which makes the files within the object container accessible through POSIX file system calls.

The user authentication services represent another crucial part of the setup of the GridFTP server. They are performed using Grid credentials, namely X.509 certificates. These can be either long-lived personal user credentials or short-lived credentials being generated by a service such as MyProxy (Novotny et al., 2011) upon user request.

Our setup of GridFTP works with short-lived credentials, which are generated by MyProxy upon a user request and can be used by the GridFTP service to authenticate itself towards remote servers whilst performing the data transfer requested by the user.

To ensure a reliable transfer of the data between different GridFTP servers, we are relying on the GlobusOnline service (Allen et al., 2011), which manages the data transfers on the user's behalf, as soon as a GridFTP server has been registered as an endpoint of the service.

2.4 User Authentication

The user authentication for the whole OpenStack installation is performed through the Lightweight Directory Access Protocol (LDAP) (Howes and Smith, 1995) thus allowing for an easy and central management of the local credentials for all services. The management of the OpenStack users follows the central grid computing (Foster and Kesselman, 2003) concept of a Virtual Organization (VO) (Foster et al., 2001) which is grouping users from different organizations, who are for example working on a shared project. Within OpenStack the VOs are mapped onto user groups called tenants, thus enabling the participants of one project to easily share resources among themselves.

The VO concept is used to assign the users to specific OpenStack tenants and also to ease the management of access rights, which are assigned at the level of VOs. A good example is given by the access to the local GridFTP server, enabling users to access the service and to use GlobusOnline to transfer data to the local OpenStack containers. All the access rights based on the VOs are represented within the central LDAP tree of the installation, which is subsequently queried by the different services during user authentication.

2.5 Client Software

jORCA (Martin-Requena et al., 2010) is a user-friendly software client for using web-services. The application was specifically designed to help users take advantage of computational resources made available as web-services, i.e. discover web-services, display available parameters, request information and finally execute the web-service. To enable all these activities, jORCA uses metadata repositories (i.e. containers of meta-information), with information about available web-services and data types. The MAPI library (Karlsson and Trelles, 2013) provides the unification of metadata information. By using this library, it is possible to extend the execution functionality of jORCA with components called workers.

2.5.1 Web-service Invocation and Data Transfer

Support for invoking a RESTful WS protocol / frontend from jORCA is also implemented, providing an interface between jORCA and the underlying cloud infrastructure. This interface provides operations to submit new jobs, to cancel previous ones, to poll for status and to retrieve intermediate and final results. To submit a new job the user has to fill the required WS parameters as references to data already uploaded on cloud data storage. Once the frontend receives the job submission it contacts the job scheduler to dispatch the new task. The scheduler will decide depending on the previously mentioned parameters if it needs to create a new instance and also if some idle instances could be deleted. The scheduler will use euca2ools (Debian Wiki, 2012) to perform the described operations.

At the end of the job submission the front-end will return a new unique URL resource to the user which can be cancelled, polled for status, intermediate results, and, when ready, the final result. The intermediate and final results will be data references to the results stored on the data storage. The user can now choose to download the data from the data storage (using the jORCA plugin developed) and/or submit the result data references as input to another WS. This greatly facilitates the invocation of a series of WS as a workflow because the intermediate data is already available on the infrastructure for a subsequent WS.

The potentially large input and output data sizes when invoking web-services deployed on the cloud infrastructure made it essential to use a well-established protocol to transfer large data sets reliably and securely. The Globus Online (GO) initiative (Allen et al., 2011) uses GridFTP to transfer large amounts of data. Additionally, GO provides easy-to-install software for client-side transfer of data and also supports long-running data transfers where GO mediates the data transfer. The actual data is transferred directly between two GridFTP servers being registered as GO endpoints; GO only monitors and controls the transfer.

jORCA can initiate a GO-mediated data transfer either upon user request or automatically if the metadata of the web-service to be invoked indicates that it requires the input data to be transferred via GO prior to invocation. To transfer data from the local computer running jORCA, it is necessary to previously install and configure the Globus Connect software, which is acting as a local GO endpoint. The transfer status – running, finished, or error – is displayed to the end-user through jORCA. Once finished, the necessary information about the uploaded file is transferred to the service which, in turn, will use GO to move the data if necessary.

2.6 Mapping the Workflow onto the Cloud Infrastructure

The biomedical workflow described in 2.1 consists of a sequence of computational steps, which are interconnected by data stored in POSIX-compliant files. The output of one step is stored on the file system, from where it is read by the next module within the workflow.

To enable a flexible execution of the workflow on top of the above-described cloud infrastructure and across multiple cloud instances, the intermediate data files are being stored in OpenStack containers.

The current cloud setup enables the containers to be mounted on the local file system of the running instances. Therefore the existing programs and scripts accessing local files do not need to be adapted to run on the cloud infrastructure. When a user wishes to execute the workflow, the first step is to upload the input data and store it in a container.

Since the initial input data will typically come from an external source, the GO service can be invoked through jORCA to transfer the data to the local cloud installation.

Sequential parts of the workflow consisting of multiple consecutive stages can either be collocated on one instance or distributed across multiple instances depending on their computational load.

The parts of the workflow, which are executed in parallel, are replicated across multiple instances running concurrently.

The instances monitor the availability of their input data: as soon as the corresponding input dataset is available in the specified container, the instance starts to copy the input data from the container to its local hard disk and to subsequently execute its computational algorithm. As soon as a group of collocated stages on one instance is finished, the output is written to a container, enabling the next stage of the workflow running on a different instance to proceed by consuming the newly produced intermediate data in the same way.

3 RESULTS

The workflow described in section 2.1 has been executed on a small community cloud installation using the technologies described in section 2.6.

The user starts by uploading the input data for the workflow, such as the SNP and CEL data, to an OpenStack container using GO through the jORCA client.

To perform the data processing workflow, a simple bash-shell script has been set up which calls the executables performing the computational as well as data management steps within the workflow.

The infrastructure enables the containers to be mounted into the local file system of a running virtual instance, and thus the data access methodology of the original non-cloud-aware workflow modules remains valid.

Initial runtime measurements have shown that the last step of the workflow – converting the Birdseed output into a VCF file – is by far the most time-consuming (see tables 1, 2, and 3). A typical use-case involves processing a large number of CEL-files and the VCF conversion of a single CEL-file takes significantly longer than the rest of workflow combined. Given these facts, it is clear that the parallelization of

the conversion to VCF is key to improving the process.

Therefore the following parallelization approach has been realized:

- All steps before the VCF conversion are executed sequentially on one instance.
- The VCF conversion is parallelized at the file level by assigning the conversion tasks for different files to different machine instances.

The workflow starts with the sequential component retrieving all the input data required for the birdseed algorithm from the OpenStack container and storing it on a local disk within the instance. It is important to note that while the data is directly accessible through the mounted container, the copy operation speeds up the computational step, which only has to access a local disk as compared to web-service calls for accessing the OpenStack container.

After the data has been made available locally the birdseed algorithm and the filtering of its results are performed in sequence before the output of the filtering step is again stored in the container.

Making the filtered data accessible within the container enables the data-parallel execution of the final VCF generation step across different cloud instances.

For the initial prototype all instances are executing the same shell script and accessing the same container. The existence of a specific file marks whether the sequential part of the workflow is still active.

While this is the case the other instances taking part in the parallel VCF generation load the required input data for the last workflow step from the container onto their local hard disks. Thus this data transfer overlaps in time with the computation of the first steps of the workflow.

Since the input data for the VCF generation includes human genome data of significant size, this overlapping reduces the overall processing time of the workflow.

As soon as the previous steps of the workflow have been completed, the responsible instance also loads the input data for the last step. Meanwhile the other instances can already start the computation within the last step of the workflow. As mentioned before, the completion of the previous workflow steps is indicated by the existence of a specific file in the shared container, which can be tested by the script being executed by all instances.

To properly distribute the files, which need to be converted to the VCF format between the involved instances the same lock-by-file approach is performed:

Within a loop, the script iterates over the set of files to be converted, and adds a corresponding lock file to the container marking each file that is currently being or has already been processed. If an input file is already locked by another instance, it is skipped and the local instance continues with the next file available.

While this approach does not guarantee the avoidance of concurrent access to the same file, is has been chosen as the synchronization mechanism for this first workflow prototype due to its simplicity.

3.1 Test Runs

Test runs of this setup have been performed based on the following input data set:

- 8 CEL files with 66 MB each
- Human genome data slightly bigger than 3 GB

The following cloud resources have been used to perform the computations:

- 5 instances with 4 GB of RAM and two Intel Core2 Duo CPUs with 2.4 GHz each. However for the execution only one CPU per instance was used.

One of the instances was used to perform the sequential part and afterwards all five instances processed the eight CEL files in parallel.

Table 1: Runtimes measured for the sequential part.

Workflow Step	**Runtime**
Input data upload	62 s
Birdseed algorithm	263 s
Filtering SNPs	26 s
Storing Result in Container	3 s
Sum	**354 s**

Within the parallel execution the data upload has only been performed once for each instance. The uploading times for the input data for the sequential and parallel parts of the workflow are different due to the different types of input data of the different workflow stages.

The result data upload in tables 2 and 3 refers to the result data of the sequential part of the workflow.

Table 2: Maximum runtimes measured for the parallel part.

Workflow Step	**Max Runtime**
Genome data upload	558 s
Result data upload	8 s
VCF conversion	5207 s
Sum	**5773 s**

Table 3: Mean runtimes measured for the parallel part.

Workflow Step	Mean Runtime
Genome data upload	547.80 s
Result data upload	2.80 s
VCF conversion	4970.33 s
Sum	**5430.93 s**

3.2 Application of the Workflow to Real Data

We have shown an example usage scenario for this workflow, where it was used to analyse the genotypes of patients with cross intolerance to non-steroidal anti inflammatory drugs, one of the most important drug allergies, which can lead to potentially fatal reactions (Ayuso et al., 2013). A group of over 100 patients were genotyped, along with over 100 control subjects. More details on the data and its analysis are given in (Cornejo-García et al., 2013). This approach led to the discovery of a number of SNPs that show a strong association with the pathology. These SNPs are currently being further investigated in a new cohort of patients and through the use of in vitro experimental assays.

4 DISCUSSION AND CONCLUSIONS

We have presented a novel cloud-based workflow, *genCloud*, which allows an end-user to perform a number of important and computationally intensive steps in the analysis of genome wide association data. We believe that close collaboration between target end-users and developers is essential for the development of such a service and to ensure that user demands are met. We also believe that cloud based systems are the most suitable and flexible solutions for the computational needs of clinical research groups in hospitals and medical research groups for two main reasons:

- They require minimal computational know-how from the end-user in terms of installation, administration and scripting. The user can therefore focus on understanding the details of the tools themselves and the biological interpretation of the results, rather than their implementation.
- They can also be scaled according to the user's needs: the analysis presented here focussed on a small, proof of principle analysis, with 8 samples. However, some GWAS experiments produce 100s, even 1000s of samples, requiring more computational resources. This isn't an issue for a cloud based system: the cloud instances started for the analysis can be as large and as numerous as required. The upload of the genome data - representing the biggest input dataset by far - is only required once. Afterwards multiple sets of SNP files can be processed by the workflow.

It should be made clear that the workflow described here is only able to handle data from the Affymetrix SNP chip microarray platform. Given the ever growing popularity of whole genome sequencing data, as well as its ever reducing cost, the next step will be to extend the workflow for this kind of data. A typical whole genome sequencing file with a high coverage can have more than 100 GB per individual. The task to extract the genotypes out of these files can take around a week and will take around 1 TB temporary hard disk size per individual. As this step is completely independent for each individual it will benefit greatly from a cloud-based implementation, since this will enable flexible scaling and parallelisation. Moreover, given that the output of the analysis of this data is also a VCF file, it could be combined with the same downstream analysis workflows that are used for the current data.

The runtime measurements presented in Section 3.1 have been collected from a workflow of instances being started by the user before the actual workflow was invoked by starting the workflow script on each instance through an interactive shell. While the automatic invocation of instances stored as snapshots has already been performed for other applications on the same infrastructure it has not been applied yet for the workflow described here. The automatic deployment of new instances will be realised as part of the scheduler component of the infrastructure, which is still in the design phase.

Future work will focus on the integration of the above-mentioned cloud technologies and our community cloud with the Galaxy workflow engine subsequently enabling us to provide a much better and simplified user experience to medical and research staff, who are not experts in cloud computing and workflows. As additional steps we will investigate possibilities of automatic exploitation of cloud elasticity to enable scalable workflows being able to handle different input sizes.

ACKNOWLEDGEMENTS

This publication is supported by the European Community through the FP7 IAPP project Mr. SymBioMath, grant agreement number 324554.

REFERENCES

Afgan, E., Baker, D., Coraor, N., Chapman, B., Nekrutenko, A., and Taylor, J. (2010). Galaxy cloudman: delivering cloud compute clusters. *BMC Bioinformatics*, 11(Suppl 12)(S4).

Afgan, E., Chapman, B., Jadan, M., Franke, V., and Taylor, J. (2012). Using cloud computing infrastructure with cloudbiolinux, cloudman, and galaxy. *Current Protocols in Bioinformatics*.

Allcock, W., Bester, J., Bresnahan, S., Plaszczak, P., and Tuecke, S. (2003). Gridftp: protocol extensions to ftp for the grid. Technical Report GFD-R-P.020, Open Grid Forum. Proposed Recommendation.

Allen, B., Bresnahan, J., Childers, L., Foster, I., Kandaswamy, G., and Kettimuthu, R. (2011). Globus online: Radical simplification of data movement via saas. Technical Report Preprint CI-PP-5-0611, Computation Institute, The University of Chicago.

Amazon Web Services (2013). Amazon simple storage service (amazon s3). http://aws.amazon.com/s3/.

Ayuso, P., Blanca-López, N., Doña, I., Torres, M., Guéant-Rodriguez, R., Canto, G., Sanak, M., Mayorga, C., Guéant, J., Blanca, M., and Cornejo-García, J. (2013). Advanced phenotyping in hypersensitivity drug reactions to nsaids. *Clinical and Experimental Allergy*, 43(10):1097–1109.

Button, K., Ioannidis, J., Mokrysz, C., Nosek, B., Flint, J., Robinson, E., and Munaf, M. (2013). Power failure: why small sample size undermines the reliability of neuroscience. *Nature Reviews Neuroscience*, 14(5):365–376.

Ceph Team (2012). Rados gateway - ceph documentation. http://eu.ceph.com/docs/wip-3060/radosgw/.

Cornejo-García, J., Liu, B., Blanca-López, N., na, I. D., Chen, C., Chou, Y., Chuang, H., Wu, J., Chen, Y., Plaza-Serón, M., Mayorga, C., Guéant-Rodríguez, R., Lin, S., Torres, M., Campo, P., Rondón, C., Laguna, J., Fernández, J., Guéant, J., Canto, G., Blanca, M., and Lee, M. (2013). Genome-wide association study in nsaids-induced acute urticaria/angioedema in spanish and han-chinese populations. *Pharmacogenomics*. in press.

Debian Wiki (2012). euca2ools - debian wiki. https://wiki.debian.org/euca2ools.

Foster, I. and Kesselman, C., editors (2003). *The Grid 2: Blueprint for a New Computing Infrastructure*. Elsevier.

Foster, I., Kesselman, C., and Tuecke, S. (2001). The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 3(15).

Howes, T. and Smith, M. (1995). A scalable, deployable directory service framework for the internet. Technical Report UM-CITI 95-7, University of Michigan.

Karlsson, J. and Trelles, O. (2013). Mapi: a software framework for distributed biomedical applications. *Journal of Biomedical Semantics*, 4(4).

Korn, J., Kuruvilla, F., McCarroll, S., Wysoker, A., Nemesh, J., Cawley, S., Hubbell, E., Veitch, J., Collins, P., Darvishi, K., Lee, C., Nizzari, M., Gabriel, S., Purcell, S., Daly, M., and Altshuler, D. (2008). Integrated genotype calling and association analysis of snps, common copy number polymorphisms and rare cnvs. *Nature Genetics*, 10(40):1253–1260.

Krampis, K., Booth, T., Chapman, B., B. Tiwari, M. B., Field, D., and Nelson, K. (2012). Cloud biolinux: preconfigured and on-demand bioinformatics compu ting for the genomics community. *BMC Bioinformatics*, 13(1):42.

Le Bras, Y. and Chilton, J. (2013). Deploying production galaxy instances on openstack with cloudbiolinux and cloudman. https://www.e-biogenouest.org/resources/243.

Martin-Requena, V., Rios, J., Garcia, M., Ramirez, S., and Trelles, O. (2010). jorca: easily integrating bioinformatics web services. *Bioinformatics*, 26(4):553–559.

Mell, P. and Grance, T. (2011). The nist definition of cloud computing. Technical Report 800-145, National Institute of Standards and Technology.

Mobley, A., Linder, S., Braeuer, R., Ellis, L., and Zwelling, L. (2013). A survey on data reproducibility in cancer research provides insights into our limited ability to translate findings from the laboratory to the clinic. *PLoS One*, 8(5). e63221.

Novotny, J., Tuecke, S., and Welch, V. (2011). An online credential repository for the grid: Myproxy. In *Proceedings of the Tenth International Symposium on High Performance Distributed Computing*.

Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M., Wipat, A., and Li, P. (2004). Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054.

Pepple, K. (2011). *Deploying OpenStack*. O'Reilly Media, first edition.

R Development Core Team (2008). *R: A language ad environment for statistical computing*. R Foundation for Statistical Computing.

Weil, S., Brandt, S., Miller, E., Long, D., and Maltzahn, C. (2006a). Ceph: A scalable, high-performance distributed file system. In *Proceedings of the 7th Symposium on Operating System Design and Implementation*, pages 307–320.

Weil, S., Brandt, S., Miller, E., and Maltzahn, C. (2006b). Crush: controlled, scalable, decentralized placement of replicated data. In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*.