The ARES Project: Cloud Services for Medical Genomics

Mauro Femminella, Gianluca Reali, Dario Valocchi
Dipartimento di Ingegneria, University of Perugia,
Via G: Duranti, 93
06125 Perugia, Italy.
[mauro.femminella , gianluca.reali]@unipg.it
dario.valocchi@gmail.com

Emilia Nunzi(*)(**)
Valerio Napolioni(**), Matteo Picciolini(**)
(*)Dip. di Medicina Sperimentale, University of Perugia
(**)Polo d'Innovazione Genetica, Genomica e Biologia
Via Gambuli, 06132 Perugia, Italy
emilia.nunzi@unipg.it, [v.napolioni,
m.picciolini]@poloinnovazioneggb.com

Abstract— **This paper shows the cloud services provided by the project ARES. The network solutions have been illustrated in a companion paper in the same conference. The ARES project aims to deploy CDN services over a broadband network for accessing and exchanging genomic datasets, accessible by medical and research personnel through a Cloud interface. This paper illustrates the procedure defined to access such services, also providing a case-study simulation to show the implementation of the bioinformatics pipeline included. The experimental activity in ARES aims to gain a detailed understanding of the network problems relating to its sustainability given the increasing use of genomics for diagnostic purposes. The main aim is to allow an extensive use of genomic data through the collection of relevant information available from the network in the medical and diagnostic field diseases.**

Keywords— Genomic Big Data, Cloud Services, Genomic Pipelines

I. Introduction

The ARES (**A**dvanced Networking for the EU Genomic **Res**earch) project is proposing a new approach for the use of genomic data in the medical/research field. [31]. It consists of a cloud-based access to genomic processing services for medical and research personnel. The need of this research is given by the massive and increasing production of genomic data.

The completion of the human genome sequencing project represented a milestone in the field of biological and medical sciences. It happened about ten years ago in the framework of the US project Human Genome. It has been the results of years of expensive research activity. Although the clear evidence for the scientific relevance of that result, at that (recent) time, the possibility of handling the human genome as a commodity was far from imagination due to the elevated cost and the complexity of sequencing and analyzing complex genomes. Today the situation is much different. The order of magnitude of the cost necessary for sequencing one human genome is getting close to 1000 € [3].

The cost for sequencing both a unit of DNA and the whole human genome over the time has been decreasing faster than the Moore's law [1]. This cost evolution is referred to as "the big drop". It has begun in 2008, and it is due to the introduction of novel sequencing machines. Under a very practical viewpoint, this means that the cost per unit data produced decreases more rapidly than the cost for storing a unit data and, more importantly, for distributing a unit data. Hence, if the trend shown in [1] will continue for some years, the bottleneck of the process of effectively using the genome information will reside on the ICT side. This is the essential rationale of our research and our proposal. Nowadays, the limiting step is no more the sequencing capability of current technology but rather the ability to process large data file efficiently and allow remote exchange and access for meta-analysis investigations. As proof of principle we have implemented a test bed for transferring and sharing genome sequence file and processing software for diagnostic purposes.

In this paper we illustrate the protocols for accessing the cloud services and a case study relevant to a real usage of a genomic processing service.

II. Background

A. Genome analysis tools.

Different genome sequencing tools have been developed in the last decade. The analysis of such data mainly consists of sequence similarity detection. This detection of similarity, is crucial in both diagnostic and research activities. BLAST and PSI-BLAST [4], [6], are the most popular sequence alignment tools. Nevertheless, the envisioned growth of genome databases poses some challenges to their suitability and reliability. Alternatives to BLAST and PSI-BLAST are, SSEARCH [8], CUSHAW[9], FASTA [10], CASAVA[7] , SAM [11], IMPALA [12], and HMMER [13].

Some alternative approaches have been also proposed for accelerating the execution of BLAST, such as its parallel execution on shared memory HPC, (SGI Altix [17]), distributed-memory HPC (IBM BlueGene/L [20]), and execution in clusters implemented through high-speed interconnections (MPP2 [16]). There are also solutions for executing parallel BLAST methods for general clusters [14], [15], [19]. In general, the objective of these approaches is to speed up the fulfillment of BLAST queries by resorting the partitioned databases. For example, the mpiBLAST package [14] can achieve super-linear speed-up with the size of databases by removing unnecessary paging. Nevertheless, some scalability issues illustrated in [16], along with problems due to result merging and I/O synchronization have not yet been addressed effectively [19].

B. Cloud and Data Services

In recent years, Google and Amazon have struggled to tackle the problem of big data handling by their specific cloud applications [2]. For example, the Amazon S3 cloud computing service provides a web services interface to store and retrieve, namely, any amount of data. This service may be used to store and retrieve genome data. Nevertheless, no specific tools are available for optimizing the perceived network quality in the basis of the specific research needs (e.g. optimized download time and number of accomplished requests through dynamic VM instantiation, and CDN provisioning). In particular, no specific tools for handling genomic data set are made available. For example, a popularity model usable for managing data is not available. For other data types, such as a video clip, the popularity evolution typically increases until a maximum is reached. Then, it unavoidably decreases over time. For genomes, the popularity evolution does not have a typical known pattern, as sketched in Figure 1. A genome is a plentiful source of information, most of which is still unveiled. It may happen that the interest over an old genome of a dead man is more interesting than a genome recently sequenced. The popularity could also decrease and increase again after some time according to the research dynamics in a totally unpredictable manner. Thus, in the short and medium terms the evolution of the available data in the future networks can be modeled as a pure-birth process. Therefore, beyond the clear need to have large amounts of storage capacity, specific solutions for managing data storage are expected. The proposal includes a dynamic cache management [18], which adapts to the request pattern of genome data.

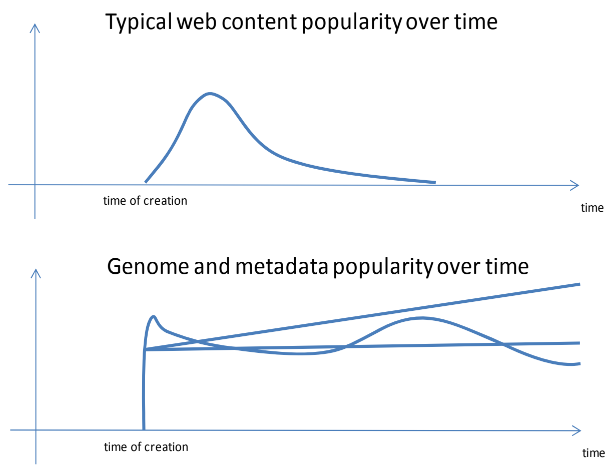

Figure 1 – Possible popularity evolution over time.

III. Network Aspects

A detailed description of the ARES network architecture is reported in a companion paper submitted to the same conference [29]. In order to clarify the working context of this paper, we provide a short and basic description of network architecture for the sake of consistency of this paper.

As mentioned above, the user interface of the system is based on a cloud support. Behind this interface, the content made available in the cloud is managed through a novel content distribution network (CDN). This CDN is implemented by means of a distributed network of servers and file storage devices, typically referred to as caches. Their dynamic management aims at improving the user perception of the cloud service. This network architecture could also be compliant with the Application Delivery Network paradigm [34].

The core protocol of this dynamic management is the NSIS suite of protocols, specified by the IETF RFC 4080 [21]. It consists of two layers:

- NSIS transport layer protocol (NTLP), which is a generic signaling transport layer used for node discovery and message sending.
- NSIS signaling layer protocol (NSLP), which is the upper layer, which implements the specific signaling application.

GIST (General Internet Signaling Transport protocol, [22]) is the NTLP implementation proposed by the IETF. It implements a set of basic capabilities, including node discovery and message transport and routing. It provides end-to-end signaling, that allows sending signaling messages towards a destination, and path-coupled signaling, that allows installing states in the NSIS peers over the path to the destination.

In [24], we have illustrated our implementation of a further routing paradigm, implementing off-path signaling. It allows sending signaling message to an arbitrary set of peers, regardless the user data flow. This mechanism is widely used in ARES to discover the available network resource and to find the optimal location where the genomic processing is implemented, as shown in what follows.

The ARES solution is implemented through a massive use of service modularization and virtualization provided by the NetServ service deployment architecture [28]. The runtime environment executing services is provided by a virtual services framework, which manages access to service building blocks and other network resources. Applications use building blocks to drive all network operations, with the exception of the IP packet transport.

The components of the NetServ architecture are:

- *Service containers*, which are user-space processes. Each container executes a Java Virtual Machine (JVM), running the OSGi framework for hosting service modules, which are Java archive files, also referred to as *bundles*.

- *NetServ controller*, which installs modules in service containers, or remove them, at runtime. In more detail, it coordinates the NSIS signaling daemons, the service containers, and the node transport layer. It makes use of the netfilter library through the iptables tool. Any packet matching with one of the service rules is routed from the network interface to a service container process, which is executed in user space. The NetServ controller is also in charge of setting up and tearing down service containers, authenticating users, fetching and isolating modules, and managing service policies.
- *Signaling module*, based on an extended version of NSIS-ka, an open source NSIS implementation by the Karlsruhe Institute of Technology [23].
- The *NetServ repository*, which stores a pool of modules deployable through NetServ signaling.

The cloud operation is managed through OpenStack, which is a cloud management system that allows using restful APIs to control and manage the execution of virtual machines (VMs) on a server pool [25]. It allows retrieving information about the computational and storage capabilities of both the cloud and the available VMs. Each VM can expose, through OpenStack, the required configuration, which allows mapping the minimum required computational capabilities for a specific processing service into the hardware configuration for the VM that executes the service.

OpenStack allows also using a restful API to inject new VMs into the pool, thus allowing the system to retrieve the relevant files from other locations. Furthermore, OpenStack is not bounded to the use of a single hypervisor, but it can make use of different drivers to support different formats of virtual machines and hypervisors, such as Q-emu, VMWare, KVM, Xen, LXC and Hyper-V.

Our architectural solution thus combines different cloud paradigms, as shown in Figure 2. We have implemented an optimized solution, the core of which is an optimization algorithm that determines the most suitable PoP for executing the software pipeline implementing the desired service. From the perspective of the medical user, the access is totally equivalent to a Software-as-a-Service (SaaS) cloud delivery model. The software components are transferred from the most convenient location, being it either a static repository or a dynamic cache managed by a NetServ service. Finally, the virtual machines executing the desired genomic software pipeline uses the computing resources made available through a Infrastructure-as-a-Service (IaaS) cloud model.

IV. CLOUD SERVICE CLASSES

The -omics (whole-genome, whole-exome, transcriptome) data processing is typically performed through a pipeline of different software packages, [26]. The input files are:

a) *FASTQ* files containing sequencer output raw data;
b) *FASTA* files containing genome/exome/transcriptome sequences;
c) *annotations* files containing the list of gene sequences.

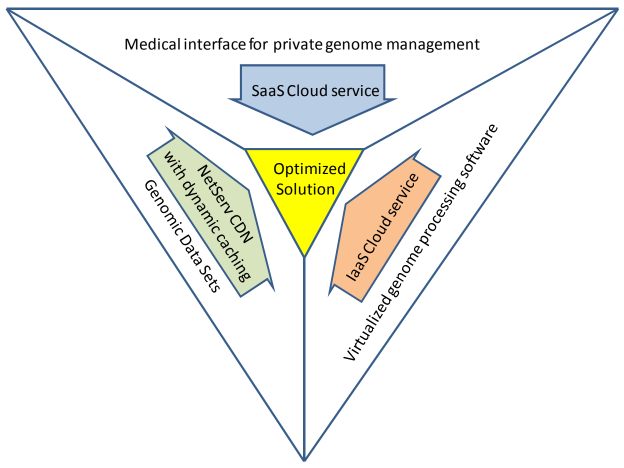
Medical interface for private genome management
SaaS Cloud service
Optimized Solution
NetServ CDN with dynamic caching
Genomic Data Sets
IaaS Cloud service
Virtualized genome processing software

Figure 2 – Networking and service paradigms contributing to the optimized solution.

Even if the patient's genome may be locally stored, all other files must be downloaded from different databases. The overall amount of data ranges from few GB to tens of GB.

When all the files are available in a selected PoP, the relevant processing can start, and it may take different hours to generate the desired result. The time taken for having this result could take some hours to same days. In the view of the increasing diffusion of genomic data, this poses two technical challenges to network designers, the minimization of the service delivery time when a very serious disease must be treated and the minimization of the overall network traffic. In regards to the former objective, some initiatives aim to decrease the processing time. For example, the Translational Genomics Research Institute (Phoenix, AZ, USA) adopts a cluster of several hundreds of CPU cores [27] to significantly decrease the processing time of the tremendous volume of data relevant to the neuroblastoma. While this approach may be expected in a small number of prestigious organizations, it cannot be generally adopted when genomic processing will be largely needed in most of countries. In regard to service time, a lot of research is urgently needed on the networking side. For example, the Beijing Genomics Institute (Bejing, China), which produces 2,000 human genomes a day, instead transmit them through the Internet or other networks, sends computer disks containing the data, via express courier [30].

We believe that a significant contribution to the minimization of the service delivery time can be provided by suitably managing shared computing and storage resources and accessing them through the illustrate cloud management operation.

In addition, we are implementing different cloud service classes according to the severity of the handled clinical situation. For example, peripheral neuroblastic tumours (Neuroblastoma, Ganglioneuroblastoma, Ganglioneuroma) *require a very rapid diagnosis*, while breast cancer and other solid cancers may be handled in some days. Neuropsychiatric conditions can be processed in a longer time-window (e.g. 2-3 weeks)

For this reason, we are implementing different cloud service classes, as follows:

- Minimum delay CDN services, to be used for handling very urgent situations. In this case optimization functions are configured for generating the shortest service delivery time.
- Short delay CDN services for handling less urgent situations, to be used for handling serious situations, but with sufficient tolerance in order to introduce some optimization of the network resources so as to increase the number of simultaneous medical requests processed by using the network resources.
- Balanced network load cloud services for handling all other situations. This class refers to delay tolerant situations, typically relevant to research purposes and not for diagnosis. In this case the optimization functions aim at maximizing the number of simultaneous medical requests processed by using the network resources.

Table I reports some examples of tolerable times for medical personnel requiring support from ARES. These tolerable times include the CDN service time, in addition to other times which depends on other medical requirements, such as the type of the sequencing, the portion of the genome to be analyzed, the processing software used and the reliability of results. Through the expertise of the researchers involved in ARES, we are translating these times in CDN service classes.

TABLE I. EXAMPLE OF TOLERABLE TIMES

Diseases	Time (days)
Neuroblastoma	2
Breast Cancer	7
Colon Cancer	7
Acute Lymphoblastic Leukemia	4
Leukemias	4
Lymphomas	4
Myeloma	7
Cervical Cancer	7
Pancreatic Cancer	4

V. EXPERIMENTAL CDN SCENARIOS

Figure 3 sketches the experiment that will be executed to demonstrate the effectiveness of our approach for supporting a generalized use of genomic information in the future medical system. In what follows we illustrate the involved entities and the steps for the execution of the experiment.

We stress that in this activity we will use publicly available genomic files with no reference to the subject's identity , and no information about current health conditions, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction.

Involved Entities:

Public Genome/Annotation Database: database storing publicly available genomic informations.

Medical Centers: we assume the existence of different medical centers that make use of genomic annotations. Diagnoses are done by processing genomics data of patients through pattern-matching algorithms. Desired patterns are taken from genome annotations.

Private Genome/Annotation Data-base: we assume that each Medical Center stores patients' genome information within a local private data-base. This information is not publicly accessible (Nevertheless, the experiments in ARES will be done by using publicly available genomic information only, that will be used also to implement the private data-bases without compromising the significance of the results).

NetServ CDN nodes: These nodes have a threefold role. As GIST enabled nodes, they discover network resources; as NetServ nodes they instantiate genome processing packages; as NetServ nodes, they implement CDN mirrors that deliver annotation files from the Annotation Data-bases to the processing Computing Node/Computing Cluster. These functions can either be co-located or implemented in separated nodes.

Computing Node/Computing Cluster: one or more GIST computers discovered by a specific NetServ service. They receive the Annotation files through the ActiveCDN nodes and the genome of the patient. After executing the pattern-matching algorithms, results are returned to the requesting Medical Center.

Controller: A single node implementing the NetServ GCM.

Steps of the experiments:

Step 1: A Medical Center asks a genomics core/facility to sequence a patient's specimen (DNA/RNA). The genomic core/facility processes the sample and makes the data available with or without having performed preliminary analysis. This would consist of searching for the already known sequences included in genome annotations for potential changes (i.e. mutations) and similar variations.

Step 2: The service request happens though the provided cloud interface. It triggers the NetServ node and resource discovery procedure, which identifies the set of one or more candidate nodes for executing the genome processing software.

Step 3: A NetServ GCM triggers a service for transferring the genome processing software over the selected Computing Node/Cluster.

Step 4: The controller triggers the software modules. They are now ready to receive input data.

Step 5: The Medical Centre sends the patient's genome data to the Computing Node/Cluster, implemented by means of OpenStack.

Step 6: The NetServ ActiveCDN service downloads the needed genomic information files from the Public Genome/Annotation data-bases into the Computing Node/Cluster.

Step7: The Controller triggers the service execution over the Computing Node/Cluster.

Step 8: The processing results are returned to the Medical Centre.

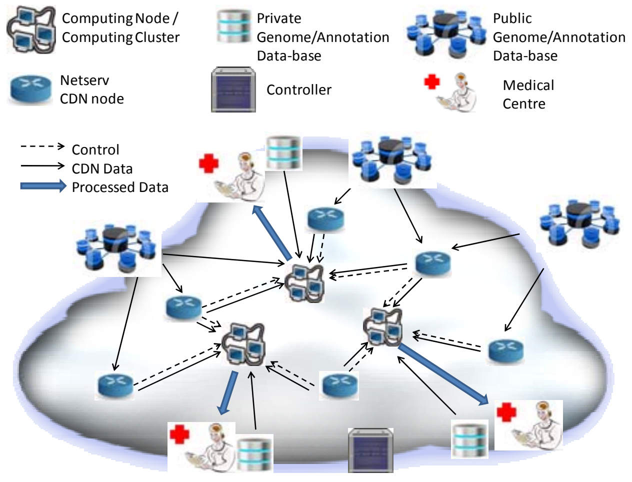
Computing Node / Computing Cluster
Private Genome/Annotation Data-base
Public Genome/Annotation Data-base
Netserv CDN node
Controller
Medical Centre
Control
CDN Data
Processed Data

Figure 3 – CDN for medical applications.

VI. Experimental Outcomes

As mentioned above, the need of resorting to the CDN paradigm for handling the illustrated scenarios derives from (i) the expected increase in the use of genome information and (ii) the greater speed of growth of the quantity of information to be used against the possibility of storing and transmitting it over the network. Thus, under the perspective of the CDN operation, it is essential to find suitable network resource management policies which provide both efficiency and suitable performance.

In the envisioned experiments, we take advantage of the balance of the two following opposing strategies:

- The NetServ CDN service struggles for pushing contents towards applications, so as to help applications in accessing data.
- The NetSev Discovery service struggle for distributing application instances in suitable network positions for minimizing network traffic, improving resource utilization efficiency, and optimizing workload distribution.

In synthesis, through an iterative definition of the experiment planning, we aim at identifying a suitable strategy for:

- Determining the optimal number of the Computing Nodes/Clusters to be used.
- Determining the optimal size of each Computer Cluster.
- Determining the suitable position of each Computer Node/Cluster over the network.

The input quantities affecting the final result are:

- The number of service requests in a given timeframe.
- The size of the used files (genomes and/or annotations)
- The required time for having the service fulfilled.
- The geographical distribution of the requesting centres.
- The geographical location of the involved databases.
- The amount of genomes and/or annotations used in experiments.
- The degree of similarity of different requests.

For example, if different service requests consists of scanning a common subset of genomes according to a common subset of annotations, the common processing can be aggregated. Clearly, the larger the number of considered simultaneous requests is, the lowest the probability of finding commonalities between them is. Nevertheless, the larger the number of considered simultaneous requests is, the higher the CDN effectiveness is.

A further trade-off affect the CDN provisioning. From the point of view of the service time, it is preferred to instantiate as many mirrors as possible, very close to the Computer Clusters. Nevertheless, overloading the network can cause excessive consumption of its resources, which can, in turn, cause shortage of storage capacity and bottleneck in file propagation through the network itself.

Our experiments will be planned iteratively, through heuristics that will sequentially explore the available degrees of freedom according to the maximum descent direction of the used metrics.

A. Case Study: Differential Expression.

This section illustrates a significant case study, consisting of a genome processing to analyze Differential Expression (DE). DE is an analysis aimed to find statistically relevant changes in genome read counts between two different experimental conditions [40]. The software pipeline is shown in Figure 5.

The following software tools are used:

- **FastQC**, which provides a simple way to do some quality control checks on raw sequence data coming from high throughput sequencing pipelines [35].
- **Trimmomatic** performs a variety of useful trimming tasks for paired-end and single ended data [37]. The selection of trimming steps and their associated parameters are provided via command line.
- **STAR** is an ultrafast tool for aligning sequencing reads to long reference sequences. STAR outperforms other aligners by a factor of >50 in mapping speed, while at the same time improving alignment sensitivity and precision [38].
- **HTSeq** is a Python package that provides infrastructure to process data from high-throughput sequencing assays [36]. It provides API and libraries to perform DE analysis.
- In addition, also the latest human genome reference model 19 (hg19, see [39]) is used.

This pipeline has been implemented over a Linux virtual machine. The final size of the disk image of this virtual machine is 3GB, and it requires to allocate at least 30 GB of RAM for executing a proper processing. The number of cores can be defined dynamically, according to the available resources and tolerable processing time. During the computations, the total disk consumption, managed by OpenStack, was up 300 GB.

In our experiment we emulated a doctor interaction with the ARES system as follows:

- Assume a doctor needs to investigate the expression of a gene related to, e.g., a cancer, and he selects the DE analysis for this purpose.
- An optimization function determines the processing location among the available PoPs.
- The implemented CDN service provides the needed genomic data sets, including the patient's genome, the size of which is a 3.2 GB, together with the VM necessary to perform the relevant processing.
- The DE analysis is done and the results is sent to the requesting medical personnel.

The functional experiments have been executed over the network shown in Figure 4. These experiments are quite simple and aim to verify the correct functional implementation of the ARES components and obtain an evidence that the dynamic caching mechanism works and is effective. The genomic data are initially stored onto a single server, and caches are dynamically populated with them over time. Three clients ask for the same service.

The overall time needed for the execution of the software pipeline is variable according to the number of biological samples analyzed. It approximately ranges in the interval between 1 and 2 days. These outcomes of experimental processing time are used to determine the remaining time available for data and VM transfer and to define the CDN service classes. In this regards, we have defined the approach illustrated in Figure 6, used in ongoing experiments, that follows a metrological approach for validating the proposed procedure [32], [33].

In particular, the outcome of measuring the client-side success of the procedure is the achievement of results within the pre-established timeframe, compliant with the CDN service deployed.

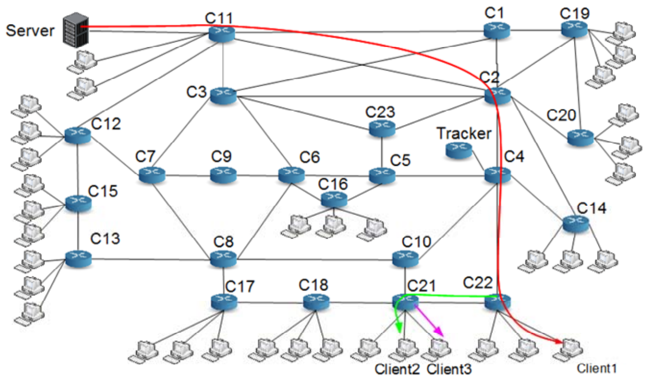

Figure 4 – Topology of the network used for experiments.

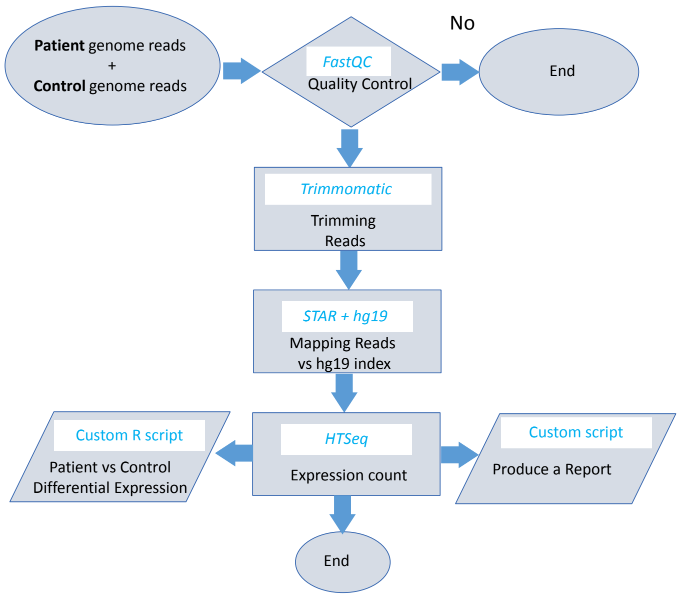

Figure 5 – Differential expression pipeline.

Each measured service time, is estimated under many different conditions. In particular, we evaluate both the worst case approach, when the CDN service is required for the first time and caches are not populated with the needed data, and also the estimate of the expected value of the service time versus the number of requests submitted to the system in a month. In this case, estimates are obtained by averaging outcomes of the experiments and have been characterized by calculating the corresponding uncertainty in terms of type A uncertainty, i.e. standard deviation of each estimate [32]. The validation of the test of the network consists in verifying that obtained estimates respect the given service time with a target reliability at least equal to 99%, i.e. that each service time estimate is lower than the target service time with a probability value at least equal to 0.99. Thus, the possibility of implementing different cloud service classes can therefore be demonstrated.

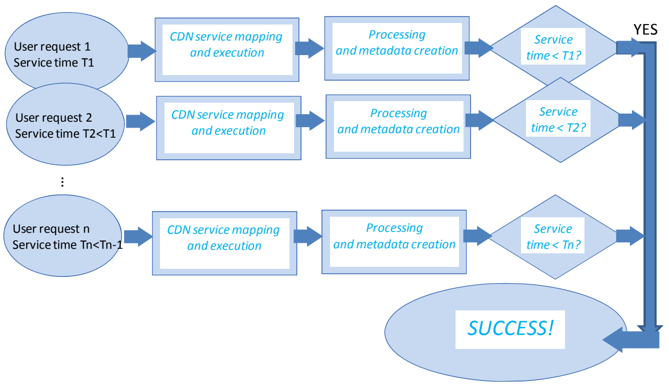

Figure 6 –Methodology used for evaluating service time performance

The project ARES is still in progress. Similar metrological approaches, based on the GUM (Guide to the expression of uncertainty in measurement) specifications, will be implemented through multiple experiments, used to collect also network-side metrics [32].

Access transparency: the set of CDN services are accessible regardless the user locations, to be verified experimentally. Success = successful verification for all locations.

Location transparency: the NSIS signaling provides transparency to any change of the repository locations. Success=transparency verified for all PoPs.

Availability: according to the CAP theorem, a distributed information system cannot guarantee consistency, availability, and partition-tolerance at the same time. The achievable availability for all CDN classes will be investigated in relation to the tolerable service time and the metrics illustrated below.

Failure transparency or *partition tolerance*: CDN service are robust to PoP and router failures. We will show how the system can manage and overcome node failures. In particular, the client programs will operate correctly after a server or repository failure. Repeated failures will be emulated so as to investigate and maximize the actual robustness. This metric is strictly related to access transparency.

Consistency: the cache instantiation and update procedures will guarantee metadata consistency. This metric is strictly related to location transparency. Repeated experiments, also in the presence of node failures, will be executed. Any experiment will be considered successful if all caches are synchronized with the relevant metadata.

Scalability: CDN services will allow increasing the tolerable network load and also scale gracefully to huge ones. Scalability will be analyzed and optimized in relation to the suitable trade-off induced by the CAP theorem.

VII. CONCLUSIONS

In this paper we have illustrated the current cloud services defined and implemented by the project ARES. These services aims to offer medical and research personnel suitable ICT tools in a networked environment for handling genome data set. Services, organized in different classes according to the time requirements of the situation handled, are accessible though a cloud interface. The cloud environment is implemented by using OpenStack. In addition to verifying the correct execution of all the virtualized software components, we have presented a case study relevant to the execution of a genomic pipeline widely used for diagnostic purposes.

Ackowledgements

This work is co-funded by EU under the project ARES, supported by GÉANT/GN3plus in the framework of the first GÉANT open call[1].

References

[1] DNA Sequencing Costs, Data from the National Human Genome Research Institute (NHGRI), Genome Sequencing Program (GSP), http://www.genome.gov/sequencingcosts/. Site visited on January 13, 2014.
[2] Amazon Simple Storage Services (S3), https://aws.amazon.com/s3/. Site visited on January 13, 2014.
[3] E. Strickland, “The gene machine and me”, IEEE Spectrum, Volume: 50 , Issue: 32013 , pp. 30 – 59.
[4] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman, “Basic Local Alignment Search Tool,” J. Molecular Biology, vol. 215, pp. 403-410, 1990.
[5] C. Trapnell and al, “Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks”, Nature Protocols, 7(3), 2012, p.562 2012.
[6] S.F. Altschul et al., “Gapped BLAST and PSI-BLAST: A New Generation of Protein Database Search Programs,” Nucleic Acids Research, vol. 25, pp. 3389-3402, 1997.
[7] Technical note: Illumina systems and software, http://support.illumina.com/sequencing/sequencing_software/casava.ilmn. Site visited on January 13, 2014.
[8] T.F. Smith and M.S. Waterman, “Identification of Common Molecular Subsequences,” J. Molecular Biology, vol. 147, pp. 195-197, 1981.
[9] Y. Liu, B. Schmidt, D. L. Maskell. “CUSHAW: a CUDA compatible short read aligner to large genomes based on the Burrows-Wheeler transform” Bioinformatics Advance Access, published May 9, 2012. http://www.nvidia.com/content/tesla/pdf/CUSHAW-CUDA-compatible-short-read-aligner-to-large-genomes.pdf. Site visited on January 13, 2014.
[10] W.R. Pearson, “Searching Protein Sequence Libraries: Comparison of the Sensitivity and Selectivity of the Smith-Waterman and FASTA Algorithms,” Genomics, vol. 11, pp. 635-650, 1991.
[11] K. Karplus, C. Barrett, and R. Hughey, “Hidden Markov Models for Detecting Remote Protein Homologies,” Bioinformatics, vol. 14, pp. 846-856, 1998.
[12] A.A. Scha¨ffer et al., “IMPALA: Matching a Protein Sequence Against a Collection of PSI-BLAST Constructed Position-Specific Score Matrices,” Bioinformatics, vol. 15, pp. 1000-1011, 1999.

[1]http://www.geant.net/opencall/Applications_and_Tools/Pages/Home.aspx#ares

[13] S.R. Eddy, “Profile Hidden Markov Models,” Bioinformatics, vol. 14, pp. 755-763, 1998.
[14] A.E. Darling, L. Carey, and W. Feng, “The Design, Implementation, and Evaluation of mpiBLAST,” ClusterWorld Conf. and Expo and the Fourth Int’l Conf. Linux Clusters: The HPC Revolution, 2003.
[15] R. Bjornson, A. Sherman, S. Weston, N. Willard, and J. Wing, “TurboBLAST(r): A Parallel Implementation of BLAST Built on the TurboHub,” Proc. 16th IEEE Int’l Parallel and Distributed Processing Symp. (IPDPS), 2002.
[16] C. Oehmen and J. Nieplocha, “ScalaBLAST: A Scalable Implementation of BLAST for High-Performance Data-Intensive Bioinformatics Analysis,” IEEE Trans. Parallel and Distributed Systems, vol. 17, no. 8, Aug. 2006.
[17] N. Camp, H. Cofer, and R. Gomperts, High-Throughput BLAST, SGI whitepaper, 2002.
[18] M. Femminella, G. Reali, D. Valocchi, R. Francescangeli, H. Schulzrinne, “Advanced Caching for Distributing Sensor Data Through Programmable Nodes”, IEEE LANMAN 2013, Brussels, April 10-12, 2013, *invited paper*.
[19] H. Lin, X. Ma, P. Chandramohan, A. Geist, and N. Samatova, “Efficient Data Access for Parallel BLAST,” Proc. 19th IEEE Int’l Parallel and Distributed Processing Symp. (IPDPS), 2005.
[20] O. Thorsen, B. Smith, C.P. Sosa, K. Jiang, H. Lin, A. Peters, and W. Feng, “Parallel Genomic Sequence-Search on a Massively Parallel System,” Proc. Fourth Int’l Conf. Computing Frontiers (CF), 2007. pp. 1607-1623, 2005.
[21] X. Fu et al., “NSIS: a new extensible IP signaling protocol suite”, IEEE Communications Magazine, 43(10), 2005, pp. 133- 141.
[22] H. Schulzrinne, R. Hancock, “GIST: General Internet Signalling Transport”, IETF RFC 5971, October 2010.
[23] NSIS-ka, open source NSIS implementation by Kalsruhe University, available at: https://projekte.tm.uka.de/trac/NSIS/wiki/. Site visited on January 13, 2014.
[24] M. Femminella, R. Francescangeli, G. Reali, H. Schulzrinne, "Gossip-based signaling dissemination extension for next steps in signaling", IEEE/IFIP NOMS 2012, Maui, US, 2012.
[25] OpenStack web site, http://www.openstack.org/. Site visited on January 13, 2014.
[26] M. Yandell and D. Ence, “A beginner’s guide to eukaryoticgenome annotation”, Nature Reviews, Genetics, vol. 13, May 2012.
[27] TGen achieves 12-foldperformance improvementin processing of genomicdata with Dell and Intel-basedHPC cluster, http://i.dell.com/sites/doccontent/corporate/case-studies/en/Documents/2012-tgen-10011143.pdf. Site visited on January 13, 2014.
[28] M Femminella, R Francescangeli, G Reali, JW Lee, H Schulzrinne, “An enabling platform for autonomic management of the future internet”, IEEE Network, 25 (6), pp. 24-32.
[29] M. Femminella, G. Reali, D. Valocchi, E. Nunzi, “The ARES Project: Network Architecture for Deliverying and Processing Genomics Data”, IEEE 3rd Symposium on Network Cloud Computing and Applications (NCCA 2014), Rome, 2014.
[30] Don Preuss, “1,000 Genomes in the Cloud and NCBI Experiences” https://respond.niaid.nih.gov/conferences/bioinformatics2012/Festival%20Proceedings/Preuss_1000_Genomes.pdf. Site visited on January 13, 2014.
[31] The project ARES, http://conan.diei.unipg.it/lab/index.php/research/ares. Site visited on January 13, 2014.
[32] “Evaluation of measurement data – Guide to the expression of uncertainty in measurement” JCGM 100:2008
[33] Nunzi, E., "Uncertainties Analysis in RTT Network Measurements: the GUM and RFC Approaches," Advanced Methods for Uncertainty Estimation in Measurement, 2006. AMUEM 2006. Proceedings of the 2006 IEEE International Workshop on , vol., no., pp.87,91, 20-21 April 2006
[34] P. Romano, F. Quaglia, "Design and Evaluation of a Parallel Invocation Protocol for Transactional Applications over the Web", IEEE Transactions on Computers, 63(2), 2014, pp. 317-334.
[35] FastQC, http://www.bioinformatics.babraham.ac.uk/projects/fastqc/. Site visited on January 13, 2014.
[36] HTSeq: Analysing high-throughput sequencing data with Python, http://www-huber.embl.de/users/anders/HTSeq/doc/overview.html. Site visited on January 13, 2014.
[37] Lohse M, Bolger AM, Nagel A, Fernie AR, Lunn JE, Stitt M, Usadel B, “RobiNA: a user-friendly, integrated software solution for RNA-Seq-based transcriptomics”, Nucleic Acids Res. 2012 Jul; 40 (Web Server issue): W622-7.
[38] A. Dobin et al, "STAR: ultrafast universal RNA-seq aligner"Bioinformatics 2012; doi: 10.1093/bioinformatics/bts635.
[39] The human genome (hg19, GRCh37 Genome Reference Consortium Human Reference 37 (GCA_000001405.1)). http://hgdownload.cse.ucsc.edu/goldenpath/hg19/chromosomes/. Site visited on January 13, 2014.
[40] S. Anders and W. Huber, "Differential expression analysis for sequence count data", Genome Biology 2010 11:R106.