The ARES Project: Network Architecture for Deliverying and Processing Genomics Data

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 (*)(**)
(*)Dip. di Medicina Sperimentale, University of Perugia
(**)Polo d'Innovazione Genetica, Genomica e Biologia
Via Gambuli, 06132 Perugia, Italy
emilia.nunzi@unipg.it

***Abstract*— This paper shows the network solutions proposed and implemented in the framework of the project ARES. The strategic objective of ARES is to create an advanced CDN, accessible through a cloud interface, supporting medical and research systems making a large use of genomic data. The expected achievements consist of identifying suitable management policies of genomic contents in a cloud environment, in terms of efficiency, resiliency, scalability, and QoS in a distributed fashion using a multi-user CDN approach. The experimental architecture envisages the use of the following key elements: a wideband networking infrastructure, a resource virtualization software platform, a distributed service architecture, including suitable computing and storage infrastructure, a distributed software platform able to ingest, validate, and analyze high volumes of data, a software engine executing a sophisticated intelligence for workload distribution and CDN provisioning.**

Keywords—Big Data, Cloud Services, NSIS Signaling, Dynamic Caching.

I. INTRODUCTION

The project ARES (Advanced Networking for the EU Genomic Research) aims to define and implement a novel content distribution network (CDN) architecture that relies on an innovative hosting platform called NetServ [9][13], which allows deploying services at runtime on network nodes. This architecture is accessible by medical and research personnel through a cloud interface. The target usage of this network is dual. First, it concerns both the access and the exchange of "big data" files, typically generated as result of human genome sequencing. Second, it allows to select the best data center in which processing all these data in a cloud fashion, so as to minimize the amount of exchanged traffic and/or the service time, that is defined as the amount of time elapsing from the initial request of processing made by the researcher/doctor up to the time in which processing results are available.

The rationale of using genome files as case study is based on the increasing scientific and societal need to utilize human genome data and the parallel development in sequencing technology that has made human genome sequencing on large scale affordable [21]. A lot fields including academia, business, public health will take advantage, in the next few years, of the massive information stored in the DNA sequence of the genome of any living body. Access to this information will allow redesigning strategic sectors such as biology, medicine, food industry, information and communication technology (ICT), and others [18]. More details about service specification for ARES cloud services can be found in a companion paper submitted to the same conference [32].

The design of ARES leverages on the following key elements:

- A wideband networking infrastructure (i.e. the Géant network [25]).
- An effective resource virtualization software platform (i.e. NetServ [13]).
- A distributed service architecture, including suitable computing and storage infrastructure, accessible in a manner similar to a cloud service (i.e by using the OpenStack cloud management framework [17]).
- A multitenant distributed software platform able to ingest, validate, and analyze high volumes of data
- A software engine executing a sophisticated intelligence for workload distribution and CDN provisioning.

The specific nature of the genome information brings further research challenges addressable through suitable cloud design. The specific features of genomic data are:

- Not expiring: while a generic web content can become obsolete and its request rate decreases over time [22], a genome is a huge source of information to be discovered. Even if the size of the single genome is a bit more than 3 GB (which is already significant) [23], the huge processing of it makes the overall related information continuously increasing. Thus, the same file can be used for decades with the same degree of popularity and importance.
- Requests are highly correlated in time and location. If a specific genome file is requested, it is likely that also genomes of people affected by the same disease and their annotations are requested as well [24]. This can be exploited to design suitable storage systems.
- The needs of research and medical personnel making

use of genomic information are specific and evolving over the time.

In synthesis, the proposed activity will facilitate the experimentation of novel solutions addressing the storage requirements of big data in clouds and their processing.

The paper is organized as follows. In section II, we provide the background about computing tools for processing genomes, content delivery networks, and cloud services. In section III, we describe the main challenges we faced in the design of the ARES solution, which is presented in some details in section IV. Finally, we draw our conclusion in section V.

II. BACKGROUND

A. Distributed computing platforms for biology

Distributed computing has attracted researchers since decades. A reference computing architecture for scientific purposes is the so-called Beowulf cluster. A Beowulf cluster can execute many different programs, in areas ranging from data mining in support of physics and chemistry research, to business areas such as the movie industry. Nevertheless, a massive use of it has shown scalability problems in multi-user environments, essentially due to lock phenomena caused by competition in using files located in a shared Network Attached Storage (NAS) or Storage Area Network (SAN). Notably, the cluster of the National Center for Biotechnology Information (NCBI) has undergone scalability issues due to the increase of the genome sequence database and parallel explosion of the incoming query rate [3][4].

BLAST [7] is one of the most widely used tools in computational biology. A crucial aspect related to the BLAST execution is how databases are distributed. Parallel BLAST typically exploits data location by grouping incoming queries and scheduling them to nodes that use the most recently requested databases [5][6]. As sequencing techniques and the relevant ontology-based databases evolve, the incoming queries exhibit dynamic patterns.

It is also worth to mention Apache Hadoop [2][20], which is an open source implementation the well known MapReduce algorithm. Apache Hadoop is a software library designed for distributed processing of large data sets across clusters of computers. It is designed to scale up from single servers to thousands of nodes, each equipped with computing and storage resources. This library itself is designed also to manage application failures, thus supporting resiliency techniques. In order to adopt Apache Hadoop to process genome files, it would be necessary to use already available applications based on the MapReduce framework, such as [15], or to adapt the desired application, which however seems an overly demanding effort. To have an idea of the required effort, please the reader can find a (non-exaustive) list of genome processing software can be found in Table 1 of [23]. Finally, even if Hadoop is a distributed platform, all entities have to be located in the same data center, otherwise the impact of the inter-data centers communications on the other network traffic becomes unacceptable[19]. This is valid for all distributed computing platforms. An interesting virtualized approach to deploy computing cluster on demand is presented in [29].

B. Content Distribution and Cloud Services

A Content Distribution Network (CDN) is a distributed network of servers and file storage devices, typically referred to as caches, deployed to store content and applications in geographic proximity to users, in order to reduce the network resources used to satisfy a customer request and improve his perception of the network service [26][27][28]. CDNs can address a wide range of needs, such as multimedia content distribution over the Internet, cache large files for fast delivery, support of on-line business applications and so on. The network architecture proposed here could also be compliant with the Application Delivery Network paradigm [35].

In recent years Google and Amazon have struggled to tackle the problem of big data handling by their specific cloud applications. 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 actually used to store and retrieve genome data. Nevertheless, no specific tools are available for optimizing the perceived network quality on the basis of the specific research needs, e.g. optimized download time and number of accomplished requests through dynamic virtual machines (VMs) instantiation, and CDN provisioning.

A first attempt in this direction is described in the work [19], which however considers the online processing of scientific data (astronomic data in the specific example) with characteristics different from genomics data.

III. PROGRESS BEYOND THE STATE OD THE ART

The ARES solution consists of the design and deployment of an advanced CDN concept, able to both suitably store information and to find suitable location for instantiating applications used for processing big data. This solution is customized on genomic processing, even if it can be applied to a number of other scenarios.

Users of our solution will access the ARES cloud interface deployed over CDN services. While a cloud approach could allow a flexible use of the infrastructure, it requires careful, strategic planning of cache deployment and management in order to address potential issues, such as security and bottlenecks. CDNs can bring significant contribution to address these issues, enabling a cloud network to meet challenging large content delivery demands for satisfying customers' expectations [27][28]. The simultaneous use of both approaches, can actually provide the expected benefits of both techniques.

The ultimate goal of our proposal is to outperform the commercially available cloud services, by using a similar level of technical complexity, for handling “big data”. The motivations underling our design choices are the following:

- Our application makes use of advanced signaling techniques able to discover the available network and computing resources through any configurable discovery algorithm.
- We can customize our CDN for any specific need of the application.

- Beyond "data mobility", we make extensive use of "application mobility".
- We can rely on geographical proximity of processing nodes, and are well aware of the node location.

Thus, our approach consists of leveraging on the CDN ability to deliver large files and introducing a suitable dynamic caching that allows the CDN to cache the content in geographically diverse areas (see also our previous work [8]). This dynamicity will be exploited for handling both genomic data caches and for suitably locating computing nodes processing genomic data, in a mutual attraction fashion. The result will be an improved end-user experience.

To achieve these results, we faced the following challenges:

A. Challenge1: Optimal Job Scheduling

Tools used for sequencing genes are computationally intensive. The time needed to fulfill queries is a function of the algorithm approach, the number and type of input sequences, and parameter settings. For this reason is extremely complex to find an optimal schedule of queries, which should also take the coordination of database accesses into account. In addition, annotation data have different data structures organized in heterogeneous metadata files [23]. Hence, it is challenging to make such data easily and efficiently accessible by requesting applications.

B. Challenge2: Decentralized Architecture

It is necessary to implement a decentralized management system, since

- the researchers producing, ingesting and retrieving scientific data work in a decentralized manner,
- it is difficult to generalize stable metadata structures.

The system should therefore be open to sharing data and deploy services on many nodes. The genomic data processing will be done through different alternative software packages. For example, we expect to make use of open source tools such as Cufflink, Tophat, Fastqc, Trimmomatic, Bowtie2, Cnvnator, Blast, Bwa and gatk).

C. Challenge3: Cloud and CDN Management

Parallel sequencing methods often optimize the response time to a set of queries by distributing workload over a given number of software instances or processors. Nevertheless this approach may result substantially sub-optimal since it ignores the query arrival pattern and do not consider to dynamically provisioning the network according to it. The outcome of our research is expected to find the suitable:

- workload distribution among different data centers, co-located with Géant points of presence (PoPs) [25],
- implementation of the VMs running the instances of the genomics processing tools over the available hosting nodes in PoPs,
- dynamic creation and autonomic distribution of CDN caches, without any manual provisioning from human operators.

The system final configuration will result from the requirements of the medical personnel participating in the project, that will be translated into CDN management policies, in terms of workload distribution, VM deployment, and dynamic cache instantiation. The application of these policies will be made possible by computing virtualization technologies, which are being deployed in the Géant PoPs and managed by means of the OpenStack cloud management framework.

IV. ADVANCED CLOUD AND NETWORK VIRTUALIZATION SERVICES

A. The NetServ architecture

In order to realize the service architecture whose main functions have been sketched in the previous sections, we resort to the NetServ architecture [9][13]. NetServ is a service virtualization framework for router and servers. In order to describe the nature and the objectives of the envisaged experiments, it is necessary to recall the main entities of the implemented NetServ architecture in some details. Figure 1 shows the general architectural model of NetServ. It is currently based on the Linux operating system.

The NetServ controller coordinates NSIS signaling daemons, service containers, and the node transport layer. It receives control commands from the NSIS signaling daemons, which may trigger installation or removal of both application modules within service containers and filtering rules in the data plane by using the netfilter library through the iptables tool. Each deployed module has a lifetime associated with it. It needs to be refreshed by a specific signaling exchange by its lifetime expiration, otherwise it is automatically removed. 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.

Service containers are user-space processes. Each container includes a Java Virtual Machine (JVM), executing the OSGi framework for hosting service modules. Each container may handle different service modules, which are OSGi-compliant Java archive files, referred to as bundles. The OSGi framework allows for hot-deployment of bundles. Hence, the NetServ controller may install modules in service containers, or remove them, at runtime, without requiring JVM reboot. Each container holds a number of preinstalled modules, which implement essential services. They include system modules, library modules, and wrappers of native system functions. The current prototype uses Eclipse Equinox OSGi framework. Service modules, represented by circles in Figure 1, are OSGi bundles deployed in a service container. Figure 1 shows two types of modules:

- Server modules, circles located within the upper-right service container. They act as standard network servers, communicating with the external through a TCP or UDP port. We will use these types of bundles in the framework of ARES.
- Packet processing modules, circles located within the

lower-left container. They are deployed in routers along packet path and can both inspect and modify packets in transit. The solid arrow in Figure 1 labeled as "forwarded data packets" shows how an incoming packet is routed from the network interface, through the kernel, to a service container process being executed in user space [13].

The module classification as *server module* or *packet processing module* is only logical, since each NetServ module may act in both ways. This is actually an important NetServ feature since it overcomes the traditional distinction between router and server by sharing each other's capabilities.

The *NetServ repository*, introduced in the NetServ architecture for management purposes [13], includes a pool of modules (either building blocks or applications) deployable through NetServ signaling in the NetServ nodes present in the managed network.

It includes an NSIS-based signaling protocol [10], used for dynamic discovery of nodes hosting the NetServ service environment, and service modules deployment therein. NSIS is an IETF standard consisting of two layers:

- NSIS transport layer protocol (NTLP), a generic lower layer used for node discovery and message sending, which is regarded as independent of any signaling application;
- NSIS signaling layer protocol (NSLP), the upper layer which defines message format and sequences; it contains the specific signaling application logic.

These NSIS signaling layers are part of the NetServ architecture. They are represented in Figure 1 by two boxes, labeled as "GIST" and "NetServ NSLP". GIST (General Internet Signaling Transport protocol, [11]), is the IETF implementation of the NTLP. GIST uses existing transport and security protocols to transfer signaling (i.e. NSLP) messages on behalf of the served upper layer signaling applications. It provides a set of easy-to-use basic capabilities, including node discovery and message transport and routing. Since its original definition, GIST allows transporting signaling message according to two routing paradigms. It provides end-to-end signaling, that allows sending signaling messages towards an explicit destination, and path-coupled signaling, that allows installing states in all the NSIS peers that lie on the path between two signaling peers.

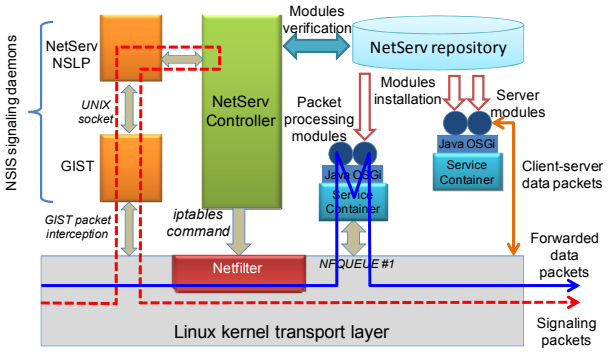

Figure 1 – NetServ node architecture.

In [14] we have shown our implementation of a third routing paradigm, named off-path signaling, that allows sending signaling message to arbitrary sets of peers, totally decoupled from any user data flow. To this aim, we defined topological off-path domains of GIST peers. An off-path domain is defined by the value of a metric, that could be the number of GIST/IP hops or the estimated network latency, and by a topological shape. In [14] we defined and implemented three different off-path topological shapes:

- Bubble, which allows disseminating signaling around the sender, i.e. to all peers such that their distance from the center of the bubble, measured according to the selected metric, is lower that the radius of the bubble;
- Balloon, which allows disseminating signaling around the receiver;
- Hose, which allows disseminating signaling messages to peers close to the network path connecting information source and destination. An example of Hose signaling is shown in Figure 2.

Thanks to the extensibility of GIST, other off-path domains can be implemented as they become necessary for different signaling scenarios

The current implementation of the NetServ signaling daemons is based on an extended version of NSIS-ka, an open source NSIS implementation by the Karlsruhe Institute of Technology [12]. As mentioned above, NSIS wraps the application-specific signaling logic in a separate layer, the NSLP. The NetServ NSLP is the NetServ-specific implementation of NSLP. The NetServ NSLP is able to manage the hot-deployment of bundles on remote nodes. NetServ NSLP offers 3 types of messages with the relevant answers:

- SETUP message, which triggers the installation of a specific bundle on the remote NetServ node. It could carry an URL, pointing to a repository from which the remote node can download the bundle, or it could even carry the bundle itself in the payload. Its header carries several information: the version of NetServ and of the Bundle, the NetServ user which requested the setup, the bundle time-to-live, and an arbitrary series of <key,value> pairs which are bundle-dependant that are passed as initialization parameters to the bundle, once it is installed.
- REMOVE message, which explicitly removes an installed bundle from the remote node, before time-to-live expiration.
- PROBE message, which is used to check if a specified bundle is installed on the remote node. The remote node can answer either that the bundle is actually installed or that NetServ NSLP is active but that the bundle is not installed. The PROBE response can also transport in the payload some specific information about the probed NetServ nodes, according to the (eventual) query specified in the forward PROBE.

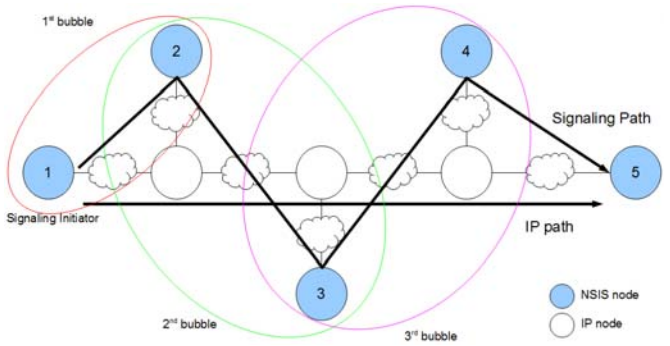

Figure 2 –NSIS Hose off-path signaling, implemented through the instantiation of multiple NSIS off-path bubbles [14] from NSIS nodes close to the IP path. Nodes 2, 3, and 4 belong to the Hose between 1 and 5. In principle, node 5 is not request to be an NSIS node.

Furthermore, it is possible to define a new NSLP for each NetServ bundle, allowing to define specific signaling logic for each functional element of our architecture. All these NSLP would be multiplexed on the common, enhanced GIST layer.

B. Virtualized service architecture in PoPs

It is known that a CDN is a network of mirrors that store replicas of the information stored within a central repository. When a user request a piece of information, the requests is redirected towards the mirror being "closest" to the requesting users. Clearly, any distance definition may be used.

We have implemented an application module, named ActiveCDN that implements CDN functionality on NetServ-enabled nodes, which is an improved version of the cache module presented in [8], which proposes to deploy ActiveCDN bundles on all the NetServ nodes on the path between the information repository and the node requesting the content, by using the NSIS on-path signaling. Also in the ARES service architecture, an ActiveCDN module is instantiated by a NetServ node having the role *of content provider*, having the full control of the placement of the ActiveCDN modules. The ActiveCDN module can be redeployed to different parts of the Géant network as needed, also by other ActiveCDN modules. This is in stark contrast to the largely preconfigured topology of existing CDNs.

The ActiveCDN module is in charge of caching both the genomic data files and the VM image files. It implements the HTTP server function by using the NetServ native support for Jetty, a light-weighted HTTP server implemented in Java, which allows creating restful interfaces and managing different servlets. When the memory space is exhausted, data will be firstly moved to storage units, and then removed according to the classical least recently used (LRU) policy. However, even if LRU strategy is currently implemented in the prototype being developed in the lab of University of Perugia, the strategy to optimally handle genomic contents deserves further investigation. In fact, there are two aspects to take into account. The first is that subsequent requests could be strongly correlated, thus a more sophisticated strategy than LRU could be designed, also taking into account the type of VM used for processing genomic data. In order to have an idea of correlations among different diseases, which could trigger specific patterns of processing requests, the interested reader can take a look at the Diseasome project [24]. The second aspects to take into account is that the files we are considering (genomes, VM images, auxiliary genomic files) can be as large as a few GBs, thus the choice of storing and maintaining them in all nodes of the path between the content repository and the requesting node could be not optimal. In particular, deduplication strategies (see e.g. [33]) have to be investigated in order to find the best performing solution.

The node acting as content provider can be a separate bundle, managing a content repository in one or more Géant PoPs, or another ActiveCDN, acting as a proxy of a content repository which is outside the scope of the Géant network. This second case is necessary to correctly handle the NSIS signaling, which is the real enabler of the overall proposed service architecture.

As for the database organization in content repositories, in order to efficiently store and retrieve large amount of genomic files, we resort to an NoSQL approach [31]. The rationale of this approach is that the NoSQL database management systems is targeted to deal with a huge amount of data, when these data do not require a relational model. Essentially, what matters is the ability to store and retrieve great quantities of data, without relationships between the elements. We propose to use an ad hoc *key-value store* solution [36] to handle stable content repositories, implemented in Java. We also propose a key-value management for in-memory caching for ActiveCDN modules.

In the proposed scenario each PoP has a certain amount of storage space and computational capabilities hosted in one or more servers and managed through a virtualization solution. This infrastructure is managed through OpenStack. OpenStack is a cloud management system that allows using both restful interface and APIs to control and manage the execution of virtual machines (VMs) on the server pool [17]. It allows retrieving information about the computational and storage capabilities of both the cloud and the available VMs. It also allows triggering the boot of a selected VM with a specific hardware configuration. Thus, it is possible to wrap, inside a certain number of VMs, different existing software packages processing genome files. Each VM is characterized by a minimum hardware configuration for executing a specific processing service. Virtual machine image files are stored within the storage units of the PoP managed by OpenStack. OpenStack allows also to inject, in the managed server pool, new VMs by using restful APIs. 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 VMWare, KVM, Xen, etc.

We assume that within the nodes in the data center managed by OpenStack, a specific node executes a VM that hosts a NetServ instance. As shown in Figure 3, NetServ hosts an OpenStack Manager bundle able to interact with OpenStack to retrieve the information on the available VMs and on the hardware capabilities, so as to offer this information to the decision system. In addition, another NetServ node will run an ActiveCDN bundle. Using also the functions of the ActiveCDN bundle, it is able to retrieve the desired VM image file by a remote location by using the relevant URL (i.e. other Géant PoPs), and using the OpenStack APIs to inject into the

OpenStack storage component this image. The ActiveCDN bundle can either be executed on the same NetServ container hosting the OpenStack Manager service module, or on an instance of NetServ deployed within a different VM.

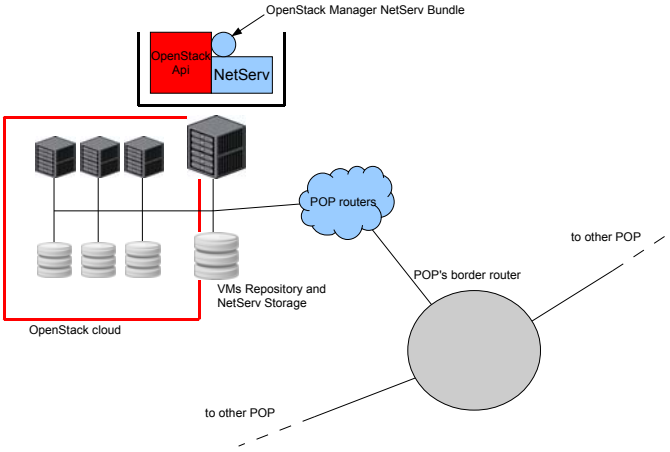

Figure 3 –Architecture of a GEANT PoP.

Finally, at least one of the NetServ nodes will run the decision system, implemented in the Genome CDN Manager (GCM) NetServ bundle, shown in Figure 4. This service module will offer an addressable service to the end users via a web-based interface, and will coordinate all the elements involved in the processing during their execution.

The GCM node will make an intensive use of the NSIS capabilities. In the proposed architecture, the GCM can use a Hose off-path signaling scheme, which extends off-path signaling [14] to the NSIS-enabled nodes adjacent to those along the IP path between the signaling initiator and the receiver. As specified in [11], GIST can use also external applications to retrieve information useful for the signaling transport. In the Hose scheme, GIST can use the traceroute utility to retrieve information on the IP nodes that lie on the path between the initiator and the signaling recipient. The signaling initiator will send this information on a signaling Bubble [14] having a certain IP hop radius. The node that receives the signaling message is able to read the IP hop radius of the Bubble and the path information, represented as a list of IP addresses. Then, it will determine if it must either forward again the signaling message or send it to the signaling application. If it is not a receiver, then it will run again a traceroute towards each of the addresses contained in the path, in order to find the node of the path at the minimum distance from itself. If this minimum distance is lower than a target metric value, it will instantiate another signaling Bubble. The use of this off-path signaling in the proposed architecture allows managing the NetServ nodes in each PoP as on-path nodes connected each other in a virtual signaling path, as illustrated in Figure 2 (see the sequence of blue nodes).

Caching and distribution systems could be even more efficient with the support of NSIS and NetServ directly in routers. There is an ongoing effort to port NetServ in Juniper routers, exploiting the Junos core, which is FreeBSD.

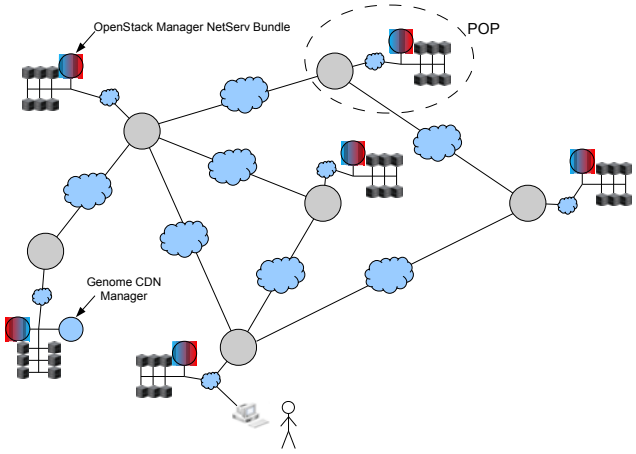

Figure 4 – Overall network scenario.

Another solution not bound to proprietary platforms is to use routers executed on general purpose hardware running linux. It is possible to enhance their performance, making it close to that achievable by a specialized hardware, through the Netmap tool [16], which is a novel framework for fast packet I/O running on general purpose linux/unix operating systems. In this case, it could be possible to execute NetServ directly on routers (implemented in linux servers), thus achieving both in-network service execution and optimized routing performance. In addition, software defined networking paradigm have been already successful tested with NetServ, as shown in [34].

C. Signaling flow for processing genome files

Our available solution handles mirrors, together with the processing tools, in a totally new perspective, which jointly optimizes good performance and scalability issues. In the following, we detail all the steps needed to accomplish a processing request.

The signaling flow starts with a user that requires a certain processing service on some genomic files. The genomic data could be stored either in a remote server or in the local data center of the user. The user interacts through a web interface with the GCM, specifying the data source (local path or URL) and the type of required processing. Once the query is sent by the user, the control passes to the GCM.

The GCM will use its own NSLP in order to retrieve the positions of the auxiliary data (annotations or reference data, such as human genome) and the appropriate VM that can wrap the requested service, by means of PROBE messages. The VM position on the storage units will be obtained by querying the OpenStack Manager NetServ bundle in the PoPs.

Then, the GCM triggers the NetServ node hosting the VM repository to initiate an NSIS session towards the NetServ node hosting auxiliary data repository, and vice versa. The NSIS session will transport a probe message according to the Hose off-path signaling scheme [14]. Each NSLP probing message will follow the "virtual" path (Figure 2), and collects the data on the computing and storage capabilities of the PoP servers hosting NetServ instances close to the path between the data

and the VM with the processing software. When the NetServ nodes have collected this information, they forward it to the GCM. Then, it uses the following information to find the suitable position of the data center where both data and VM image have to be moved: data size, VM image size, minimum computing requirement, available bandwidth and network latency between data and processing VM, and available computing and storage capabilities of both the cloud node hosting the VM and of the cloud node hosting the data. The selected location could be chosen on the basis of the parameter to minimize, such as the overall service time and/or the exchanged network traffic, on the basis also of the requirement set by the requesting person (for detail see [32]). When the processing service has finished, the output data will be returned to the end-user for download and stored, for any future usage, in the NetServ ActiveCDN bundle.

Now, let us focus on the download phase, which is the basic CDN service offered by the proposed platform. Figure 5 shows the main steps of that signaling. Each HTTP interface is managed by the ActiveCDN bundle [8]. As said above, this bundle is designed in order to offer generic data distribution exploiting the NetServ hot-deployment capabilities for caching the data in a dynamic and efficient way. Each time an HTTP GET message reaches the restful interface managed by a NetServ ActiveCDN bundle, it starts an NSIS signaling session for installing a NetServ cache in strategic points of the network. The server uses the NetServ NSLP to setup the caches on the download path. Then, it sends the PROBE message to retrieve an ordered list of the ActiveCDN caches installed along the path and, finally, it sends another SETUP message to change the state of the caches on the path. The latter message specifies the download sequence indicating, for each cache, the address of the cache from which the content can be downloaded, thus forming a sequence from the client backward to the cache initiating the signaling session. Then, the server redirects the GET message to the cache nearest to the client. The cache bundle searches for the content and, if it is not found, it asks the content to the cache specified in the signaling message. The procedure either goes on until one cache actually hosts the content, or the request is returned back to the first cache, that can certainly send the requested content. The caches on path will store this content, passing through them, and finally the last one will deliver it to the client. Future requests will be redirected by the same NSIS signaling session illustrated above, even if the content will be served directly by the nearest cache hosting the data. This procedure is illustrated in Figure 5.(b). In the considered network scenario, the requesting HTTP client is the ActiveCDN bundle running in the PoP selected for hosting the computation, whereas the HTTP server is the ActiveCDN bundle/content repository found during the NSIS sessions triggered by the GCM.

In the network scenario of this proposal, there may not be NetServ nodes on the IP path between an arbitrary pair of NetServ nodes. Hence the Hose signaling scheme illustrated above is required both to install and to manage the ActiveCDN caches adjacent to the path (the virtual path depicted in Figure 2). To this aim, the NetServ SETUP and PROBE messages will be carried by the Hose off-path signaling.

Finally, we have to mention parallel downloading techniques (see, e.g., the related work section of [35]). In fact, the ARES network architecture, being able to cache the same content in different parts of the network, could also exploit parallel downloading to efficiently transfer large files towards a destination from surrounding caches. We will evaluate the applicability of those techniques during the project lifetime.

D. Experimental activity

Our planned experiments aims to investigate (i) which metric is suitable for the successful usage of genomes and annotation for diagnosis and research purposes, for associating requesting users with an available mirrors (such as total number of hops, estimated latency, available bandwidth), (ii) the best policy for dynamically instantiating and removing caches, with reference to the user perception, the signaling load, and the information refresh protocols, (iii) the achievable performance when only a subset of nodes, with a variable percentage, execute the NetServ modules analyzed.

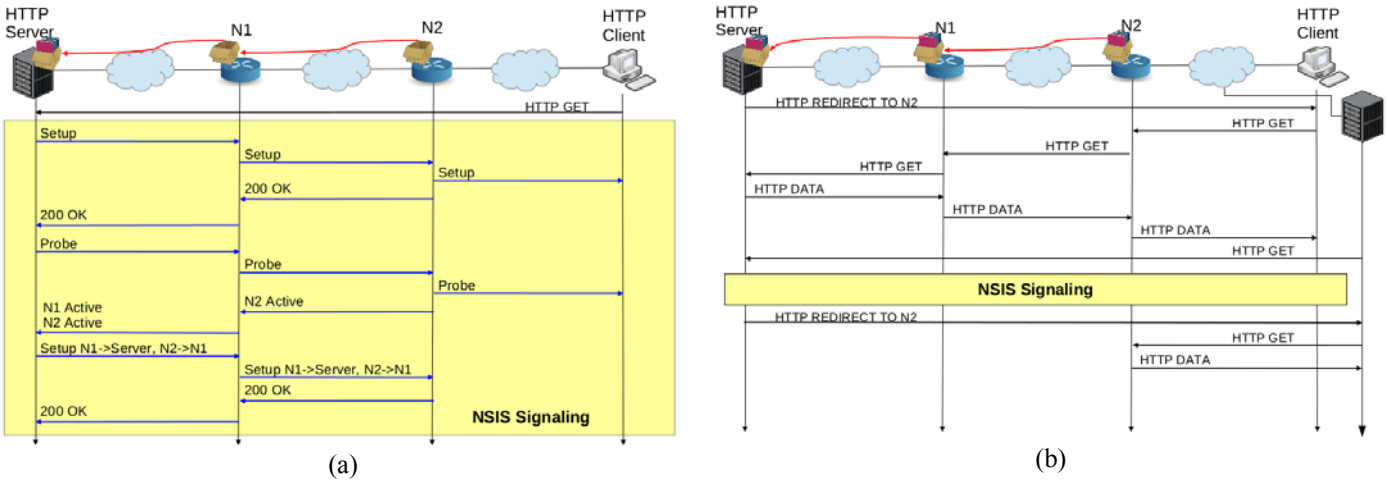

Figure 5 – (a) NetServ-based CDN signaling, highlighted in yellow in subfigures, and (b) data transfer and subsequent signaling.

Experiments will be structured according to the following steps:

- definition of the service requirements for genome and annotation exchange;
- deployment of a suitable architectural framework, able to provide suitable insights in service operation and usage perception in biomedical research environments;
- deployment of mechanisms and protocols for service description, discovery, distribution, and composition;
- performance evaluation, in terms of QoE, scalability, total resource utilization, signaling load, performance improvement and coexistence with legacy solutions.

V. CONCLUSIONS

In this paper we have illustrated the networking solutions defined and implemented in the framework of the project ARES. This project aims to offer medical and research personnel a suitable tool for handling genome data set in a networked environment. This proposal makes an extensive use of virtualization, a dynamic and scalable cache management mechanism based on a modified NSIS signaling, augmented through the introduction of off-path signaling distribution.

ACKOWLEDGEMENTS

This work is co-funded by EU under the project ARES[1], supported by GÉANT/GN3plus in the framework of the first GÉANT open call[2].

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 Dec. 20, 2013.
[2] Apache Hadoop web page: http://hadoop.apache.org/. Site visited on December 20, 2013.
[3] S. McGinnis and T.L. Madden, "BLAST: At the Core of a Powerful and Diverse Set of Sequence Analysis Tools," Nucleic Acids Research, pp. W20-W25, 32(Web Server issue), July 2004.
[4] D.L. Wheeler et al., "Database Resources of the National Center for Biotechnology Information: Update," Nucleic Acids Research, vol. 32, pp. D35-D40, 2004.
[5] 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.
[6] C. Oehmen and J. Nieplocha, "ScalaBLAST: A Scalable Implementation of BLAST for High-Performance Data-Intensive Bioinformatics Analysis," IEEE Trans. Parallel and Distributed Systems, 17(8), Aug. 2006.
[7] Oehmen CS, Baxter DJ., "ScalaBLAST 2.0: rapid and robust BLAST calculations on multiprocessor systems", Bioinformatics, 15 March 2013, 29(6), pp. 797-798.
[8] 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*.
[9] The NetServ project, http://www.cs.columbia.edu/irt/project/netserv/. Site visited on December 20, 2013.
[10] X. Fu et al., "NSIS: a new extensible IP signaling protocol suite", IEEE Communications Magazine, 43(10), 2005, pp. 133- 141.
[11] H. Schulzrinne, R. Hancock, "GIST: General Internet Signalling Transport", IETF RFC 5971, October 2010.
[12] NSIS-ka, open source NSIS implementation by Kalsruhe University, available at: https://projekte.tm.uka.de/trac/NSIS/wiki/.
[13] M Femminella, R Francescangeli, G Reali, JW Lee, H Schulzrinne, "An enabling platform for autonomic management of the future internet", IEEE Network, 2011, 25 (6), pp. 24-32.
[14] 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.
[15] A.McKenna, et al., "The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data", Genome Res. 2010 September, 20(9), pp. 1297–1303, doi: 10.1101/gr.107524.110
[16] L. Rizzo, "netmap: A Novel Framework for Fast Packet", USENIX ATC '12, Boston, USA, June 2012.
[17] OpenStack web site, http://www.openstack.org/. Site visited on December 20, 2013.
[18] E. E. Schadt, M. D. Linderman, J. Sorenson, L. Lee, G.P. Nolan, "Computational solutions to large-scale data management and analysis", Nat Rev Genet. 2010 September; 11(9), pp. 647–657.
[19] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, Francis C.M. Lau, "Moving Big Data to The Cloud: An Online Cost-Minimizing Approach", IEEE Journal on Selected Areas in Communications, 31(12), December 2013, pp. 2710 - 2721.
[20] A. O'Driscoll, J. Daugelaite, R. D. Sleator, "'Big data', Hadoop and cloud computing in genomics", Journal of Biomedical Informatics, vol. 46, 2013, pp. 774–781.
[21] J. Zhang, R. Chiodini, A. Badr, G. Zhang, "The impact of next-generation sequencing on genomics", Journal of Genetics and Genomics, vol. 38, 2011, pp. 95-109.
[22] L. Breslau, Pei Cao, Fan Li, G.Phillips, S. Shenker, " Web caching and Zipf-like distributions: evidence and implications", IEEE Infocom'99, 21-25 March 1999, New York, USA.
[23] M. Yandell and D. Ence, "A beginner's guide to eukaryotic genome annotation", Nature Reviews, Genetics, vol. 13, May 2012.
[24] Diseasome, "Exploring the human disease network", http://diseasome.eu. Site visited on December 20, 2013.
[25] GÉANT: Network, http://www.geant.net/Network/Pages/default.aspx. Site visited on December 20, 2013.
[26] D.C. Verma, "Content Distribution Networks: An Engineering Approach", Wiley, March 2003, ISBN: 978-0-471-46413-6.
[27] W. Jiang, S. Ioannidis, L. Massoulié, F. Picconi, "Orchestrating massively distributed cdns," CoNEXT '12, New York, USA, 2012.
[28] M. Yu, W. Jiang, H. Li, I. Stoica, "Tradeoffs in CDN designs for throughput oriented traffic," CoNEXT '12, New York, USA, 2012.
[29] M.Caballer, C. de Alfonso, F. Alvarruiz, G. Moltó, " EC3: Elastic Cloud Computing Cluster", Journal of Computer and System Sciences, Volume 79, Issue 8, December 2013, Pages 1341–1351.
[30] N. Lynch, S. Gilbert, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services", ACM SIGACT News, 33(2), 2002, pg. 51-59.
[31] Moniruzzaman AB, Hossain SA, "NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison", http://arxiv.org/abs/1307.0191.
[32] M. Femminella, G. Reali, D. Valocchi, E. Nunzi, V. Napolioni, M. Picciolini, "The ARES Project: Cloud Services for Medical Genomics", 3rd IEEE Symposium on Network Cloud Computing and Applications (NCCA 2014), Rome, 2014.
[33] S.Sanadhya, R. Sivakumar, Kyu-Han Kim, P. Congdon, S. Lakshmanan, J.P. Singh, "Asymmetric Caching: Improved Network Deduplication for Mobile Devices", ACM Mobicom 2012, Istanbul, Turkey, August 2012.
[34] E. Maccherani et al., "Extending the NetServ Autonomic Management Capabilities using OpenFlow", IEEE/IFIP NOMS 2012, Maui, US, 2012.
[35] 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.
[36] M. Seeger, "Key-Value Stores: a practical overview". 21 September 2009, http://blog.marc-seeger.de/2009/09/21/key-value-stores-a-practical-overview. Site visited on Dec. 20, 2013.

[1] http://conan.diei.unipg.it/lab/index.php/research-ares
[2] http://www.geant.net/opencall/Applications_and_Tools/Pages/Home.aspx