# Orchestration of Cloud Genomic Services

Gianluca Reali, Mauro Femminella, Luca Felicetti, Matteo Pergolesi
Department of Engineering, University of Perugia, CNIT Research Unit
Perugia, Italy

***Abstract*— The amount of resources necessary for genome processing services in clouds is huge. The introduction of suitable service orchestration procedures in cloud networks is therefore essential. The current achievements on SDN and NFV for cloud management are a valuable starting point for research. In this paper, we consider different classes of genome processing services and, for each of them, formulate an integer linear programming problem. The proposed solutions leverage network replacement caching, which is a key element for improving scalability of service deployment all the optimization strategies. Results show significant reduction of exchanged traffic volume and transfer time.**

***Keywords— Cloud genomics; SDN; NFV; network caching; signaling; resource management***

## I. INTRODUCTION

The Human Genome Project (HGP), whose goal was the complete mapping of all the DNA of human beings, is a milestone in biology and medicine [1]. The binary size of a raw human genome is about 3.2 GB. However, the volume of data exchanged for processing a genomic is much higher, due to the need of using additional auxiliary files and metadata, available in specialized repositories. Furthermore, given the impressive insights received from genome processing, the number of genomic applications increases over time [3] and it is expected that networked genomic services will shortly be more consuming of network resources than the most widely used online applications, such as YouTube and Twitter [4]. The most convincing solution for accessing genomic services is through a cloud interface [2][5][11]. Hence, the need of introducing new technologies and protocols for orchestrating genome processing services in the clouds is pressing.

The status of the art of software defined networking (SDN) [10] and network function virtualization (NFV) [6] represents a valuable reference for this issue. In this paper, we present a novel SDN-based orchestration for cloud genomics. Our proposal integrates some recent results on NFV-based deployment of genomic services [18], orchestrated by means of a distributed management of cached genomic contents. In this paper we change the approach and put forward a proposal based on centralized service management, based on SDN, for improving the dynamic features of service deployment. We named the resulting network model Genomic Centric Networking (GCN), as it poses as an original vertical application of the 5G softwarized network architecture.

Our contribution is as follows. We classify the networked genomic services on the basis of the features that can be optimized. For each of them we formulate an ILP optimization problem and indicate the related solutions. The effectiveness of solutions is improved by means of network caching, which results to be determinant for scalability of service deployment. Although the benefits of network caching have already been demonstrated in different applications and protocols (e.g. Domain Name Service, Content Distribution Services, Information Centric Networking [16]), what emerges is that the distinctive features of genomic data sets and services, such as a long data lifetime, allow making a better use of the caching potentials. Finally, we show some results collected by experiments performed by using a distributed GCN test-bed realized by integrating open-source software platforms as OpenStack [7]. The proposed framework includes a centralized SDN orchestration engine that allows evaluating the most convenient datacenter for deploying the requested services. The selection is done by solving the mentioned optimization problems. Experimental results show that the proposed strategy can significantly decrease the volume of traffic exchanged to execute the genomic services in comparison with other solutions.

The paper is organized as follows. In Section II, we describe the related work in the area. Section III illustrates the Genomic Centric Networking and the service orchestration, including the optimization problems, the signaling architecture, and the network caching analysis. Experimental performance is presented in Section IV. Section V reports our conclusions.

## II. RELATED WORK

### A. Cloud-Based Genome Processing Services

Amazon Web Services (AWS) provides a dataset including the genomes of 1,700 people via the Amazon storage service (S3), taken from the well known 1000 Genomes Project [8] [9]. Other initiatives include the Elixir project, which provided the pan-European network and storage infrastructure for biological data [12], the Open Science Data Cloud, provided by the Bionimbus project [14], and the ARES project, consisting of an overlay architecture for genomic computing implemented over the Géant network [15].

These initiatives either do not consider networking aspects [12][14], or include just a high level description, without including specific networking solutions with a rigorous evaluation of the expected performance [13][15].

### B. Software Dafined Networking and Network Function Virtualization

Software-defined networking (SDN) [10] consists of a centralized network management approach decoupling network management and traffic control functions. It allows network operators to dynamically control the behavior of network

nodes, by the abstraction of protocol functionality. The entity that orchestrates the behavior of networks nodes is the so-called SDN Controller.

NFV refers to a novel deployment solution of network functions which relies on virtualization of resources. Instead of implementing network functions in custom hardware appliances, they reside in virtual machines running in datacenters. The resulting virtualized network functions (VNF) constitutes building blocks that can be chained to create communication services. Different NFV orchestration platforms exist. A noteworthy example is Open Source MANO (OSM) [21]: it is a project delivering an open source Management and Orchestration (MANO) stack compliant with the ETSI NFV Information Models.

## III. Genomic Centric Networking

We consider a network of datacenters managed by means of an SDN controller. For simplicity, we assume a single controller entity, although multiple coordinated SDN controller entities can be deployed in operation, with some issues that are not in the scope of this paper. In what follows we describe the entities specific of our proposal.

GCN interface: the user interface for accessing cloud services in a SaaS fashion.

GCN_OE (GCN orchestration engine): the entity that orchestrates network operation and service deployment. It receives service requests issued through the GCN interface. According to the type of the requested services, it selects the suitable optimization problem and finds the relevant solution. It interacts with all network entities deployed in datacenters and with data repositories. The interface that the CGN_OE uses for enforcing its orchestration decision is specific of the kind of NFV platform that is used to deploy the GCN service. For example, in the ETSI NFV MANO model, the GCN_OE is modeled as a service specific component of the so-called NFV orchestrator (NFVO) module.

Data repositories: they store genomic data sets, auxiliary files and VM images which includes some pipelines of genomic software packages. All files are already split in chunks, ready for distribution. They make use of south-bound signaling protocol for communicating with the SDN controller. Since different pipelines can make use of the same reference/auxiliary files, and the size of these files is typically of several GBs, it is preferable not to include them in VM images, so as to improve SDN flexibility and NFV cache efficiency.

Cloud tenants: they are portions of datacenter resources configured to provide genomic processing services. They include (i) GCN_LM (GCN local manager), which manages the tenant. It checks the available resources when a service request arrives, and configures, starts, and stops the processing service assigned to the tenant; (ii) Cache module, consisting of an NFV cache module that receives file chunks; (iii) Infrastructure Managers, which are agents and interfaces that allow orchestration entities, such as NFVO, to interact and configure the underlying virtual and physical infrastructure.

### A. Problem Formulation

We model the GCN system as a directed graph G=(V,E), where V denotes a set of nodes of cardinality V, and E denotes a set of directed links of cardinality E. V is partitioned in some subsets: a subset S of data repositories, a subset of Point-of-Presence (PoP) elements, a subset N of routers, a subset U of user nodes, and a subset D of datacenters. Although PoPs can include a complex network of other nodes [17], we *model* each PoP as a single equivalent router. PoPs connect data repositories, datacenters, and user nodes to the GCN system. Datacenters are managed by a cloud operating system (e.g. OpenStack) which organizes datacenter resources as tenants. Network links may be either physical or virtual, built by using an overlay virtualization technology. Each tenant $i \in \mathcal{D}$ is characterized by the triple $<P_i(t), R_i(t), H_i(t)>$, denoting the available computing, memory and storage resources at time t.

Datasets required by processing tools are organized in a catalogue $\mathcal{M} = \{1,..,M\}$. We assume that they are stored in chunks of equal size $c$. Given a file $m \in \mathcal{M}$, $s(m)$ denotes its size and $m_j$ denotes the $j$th chunk of the file.

We assume that datacenter nodes can include a cache module $C_n$, $n \in \mathcal{N} \cup \mathcal{D}$, used for caching chunks. The number of chunks of the file $m$ equals $I_m = \lceil s(m)/c \rceil$. $z_{m_j}^{C_n}(t)$ is a binary variable that takes value 1 if $m_j$ is stored in $C_n$ at time $t$.

For what concerns personal genome files, they must be protected against privacy issues. For example, it is preferable to avoid caching them in the network. Service requests are indexed by $k$, $k \in \mathbb{Z}^+$. Each request is mapped to one of the genomic pipelines available in the pipeline catalogue $\mathcal{W} = \{1,..W\}$. $f(\cdot)$ denotes a function that maps $k$ into $w \in \mathcal{W}$, i.e. $f(k)=w$. Each pipeline $w \in \mathcal{W}$ needs of its own VM image file $v_w \in \mathcal{M}$, which includes all the programs needed to execute the desired processing. Each pipeline $w \in \mathcal{W}$ requires a set of reference and auxiliary files, referred to as $\mathcal{A}_w \subset \mathcal{M}$, with $\mathcal{A}_w = \{m_{w1},..m_{wL}\}$. $g_k$ denotes the genomic data set relevant to the request $k$. This request is characterized by the triple $<p_k, r_k, h_k>$, which denotes the minimum computing, memory, and storage requirements necessary to execute the pipeline $w=f(k)$.

### B. Design and Optimization

In this section, we formulate some optimization problems as integer programming problems. We consider distinct problems for different classes of genome processing services:

*Services without performance guarantees* (Problem A), *Services with bounded genome processing time* (Problem B), and *Services with guaranteed total time, including both network and processing time* (Problem C). In what follows we formulate all these problems.

#### 1) Problem A

This problem is relevant to a class of genomic services that do not require processing results in a guaranteed time. For

example, genome processing for medical prevention. In this case, the problem aims to minimize the infrastructure cost.

We define $\mathcal{D}^* = \left\{ i \in \mathcal{D}^* \mid \langle p_k, r_k, h_k \rangle \underline{\pi} \langle P_i, R_i, H_i \rangle \right\}$ the subset of datacenters with enough resource to handle the *k*th request. The operator $\underline{\pi}$ denotes component-wise inequality. Let $l_{n,i}$ denote the IP hops distance between a node storing the desired content and a the datacenter *i*, candidate for running the desired pipeline. $x_{m_j}^{n,i}$ is a binary variable that takes value 1 if the *j*th chunk of content *m* is downloaded from a node *n* towards a destination *i*.

The problem A is formulated as follows:

$$\min_{i \in \mathcal{D}^*} \left\{ \sum_{m \in \{\mathcal{A}_w \Upsilon v_w\}} \sum_{j=1}^{I_m} \sum_{n \in CYS} l_{n,i} s(m_j) z_{m_j}^n x_{m_j}^{n,i} + l_{u_k,i} g_k \right\} \quad (1)$$

st

$$\sum_{n \in CYS} z_{m_j}^n x_{m_j}^{n,i} = 1 \quad \forall j = 1,..I_m, \forall m \in \{v_w \Upsilon \mathcal{A}_w\}$$
$$z_{m_j}^n \in \{0,1\} \quad (2)$$
$$x_{m_j}^{n,i} \in \{0,1\}$$

The first sum represents the amount of traffic generated to transfer the VM image and the auxiliary files to the datacenter *i*, and the second one is the amount of traffic generated to transfer to datacenter *i* the non-cacheable input file of size $g_k$. The constraint (2) means a datacenter *i* can receive a chunk $m_j$ from one location only. Parallel transfer is not allowed.

A solution of the problem is easily found by using the well know Dykstra link-state routing algorithm from each node $i \in \mathcal{D}^*$ for each chunk to be downloaded. The personal genome file in (1) is considered as a content available in a single location. Given that smart implementations can make the cost for executing the Dykstra algorithm equal to $O(E\ log(V))$, the overall cost for executing of this solution of the problem A is

$$C(A) = \left| D^* \right| \left\lceil \frac{s(m)}{c} \right\rceil O(E \log(V)) \quad (3)$$

By introducing the possibility of caching the downloaded chunks in network nodes, the set of possible nodes storing the desired content changes over time, and is highly dependent of the caching policy. This aspect is considered in Section III.C.

*2) Problem B*

This problem considers a number of service requests all together, collected in a time window. The size of the window may be tuned according to the available computing machinery, and does not hinder the generality of the problem. Consider a request of some VMs of *W* types, that require an amount $<\Phi_p, \Phi_r, \Phi_h>$ of processing, memory, and storage resources. Each $\Phi_u = [\phi_{u,1,...}, \phi_{u,W}] \in \mathbb{Z}_+^W$ resource allocation, with $u \in \{p,r,h\}$, guarantees processing time. In other words, the value of each component $\phi_{u,k}$ is the number of requested size-*k* portions of resource *u*. For simplicity, we assume that all datacenters are equipped with the same $< P, R, H >$ set of resources, which are associated with requests according to *cutting patterns*. A cutting pattern is the way each resource type is subdivided and allocated to VMs for avoiding waste of resources in datacenters. For example, the *v*-th memory cutting pattern $\Theta_{r,v} = [\theta_{r.v,1}, ..., \theta_{r.v,k}] \in \mathbb{N}^C$, *k* specifies $\theta_{r.v,k}$ units of memory, $k \in \{1,..., W\}$, with

$$\sum_{k=1}^{W} \theta_{r,v,k} \le R, \quad \forall v \in \{1,...,C\} \quad (4)$$

The following problems minimizes the number of cuts $t_v$, $v \in \{1,...,C\}$, that is the number of used datacenters, for allocating the requested memory.

$$\min G = \sum_{v=1}^{C} t_v \quad (5)$$

st

$$\sum_{v=1}^{C} t_v \theta_{r,v,k} \ge \phi_{r,k} \quad \forall k \in \{1,...,W\}$$

$t_j \in \mathbb{N}$

This problem corresponds to the well known *one-dimensional cutting stock problem*, from which we inherited the cutting pattern concept. Solutions are known [19]. The same minimization is done for each $u \in \{p,r,h\}$, and the maximum value of *G* is the solution of the problem. Clearly, if this maximum value of *G* is higher than the number of nodes the problem cannot be solved and some requests are dropped.

*3) Problem C*

In this problem, the total time $T_{tot}$ for receiving processing results after sending a request is *minimized*. Given the instantiation of VMs, we assume that the relevant processing time for the requested service is $T_j$ is known. Thus, the network time $T_N = T_{tot} - T_j$ is the upper bound for transferring all files to the selected datacenter. In order to guarantee a bounded transfer time, bandwidth of links must be *guaranteed*. For example, overlay network connections with guaranteed bandwidth are commonly offered by network operators and implemented through SDN. If $b_{n,i}$ denotes the available bandwidth of the link between node *i* and node *n*, the time needed for transferring a number of $x_n$ chunks from node *n* to node *i* is

$$q_{n,i}(c) = \frac{x_n c}{b_{n,i}} = a_{n,i} x_n \quad (6)$$

The problem can be formulated as follows:

$$\min_{i, \underline{x} = (x_1, ... x_U)} \Omega = \min_{i, \underline{x} = (x_1, ... x_U)} \left( \max \left( a_{n,i} x_n \right) \right) \quad (7)$$

st

$$c \sum_{j=1}^{U} x_j = s(m)$$

$x_j \in \mathbb{N}$

The solution of this problem can be found by first determining the optimal value for each value of *i*, and then

selecting the minimum of these values. Thus, for each value of *i* the following sub-problem is considered:

$$\min_{\underline{x}=(x_1,\ldots x_U)} Y(i) \tag{8}$$

st

$Y(i) - a_{n,i}x_n \geq 0 \quad \forall n$

$\sum_{n=1}^{Q} x_n = s(m)$

$x_n \in \mathbb{N}$

We relax this problem by considering two generic conditions, the continuous domain, which is acceptable since $c << s(m)$, and the mathematical equality:

$$\min_{\underline{x}=(x_1,\ldots x_U)} Y(i) \tag{9}$$

st

$Y(i) - a_{1,i}x_n = 0$

$Y(i) - a_{m,i}x_m = 0$

$x_n + x_m = \alpha < s(m)$

$x_n \in \mathbb{R}_0^+$

$x_m \in \mathbb{R}_0^+$

where $\alpha$ is a constant generic portion of the original message. The solution of this problem is given by:

$$a_{n,i}x_n = a_{m,i}\ x_m = \alpha \frac{a_{n,i}a_{m,i}}{a_{n,i} + a_{m,i}} = \frac{\alpha}{\frac{1}{a_{n,i}} + \frac{1}{a_{m,i}}} \tag{10}$$

Extending the calculus to all pairs, the optimum value is given by

$$Y(i) = a_{n,i}x_n = \frac{s(m)}{\sum_n \frac{1}{a_{n,i}}} \qquad \forall n \in \{1,\ldots,Q\} \tag{11}$$

The solution of the problem is immediately found for each value of *i*. The optimum value of $\Omega$ is therefore

$$\Omega_{opt} = \min_i Y(i) \tag{12}$$

If $\Omega_{opt} \leq T_N$ the problem is solved, otherwise it is necessary to drop requests that maximize the value of (8). Since we have relaxed the problem, a feasible solution requires rounding to the next integer, which is tolerable for small chunks.

### C. SDN-enabled Caching

For what concerns *Problems A* and *C*, caching has a direct effect on the optimization metric. As for *Problems B*, which relates to processing time only, caching has no effects on it, but it is clear that the presence on cached contents to be downloaded is extremely useful in operation. The up-to-date approach for implementing network caching functions is resorting to the NFV paradigm. Thus, caches are deployed as VNFs into NFVi-PoPs, where the necessary amount of computational and storage resources are available. Requests of content redirection toward these caches can leverage the SDN paradigm to intercept the traffic flowing from content servers to the content requestors, and steer it through the datacenter hosting the caches, as detailed in Section III.D.

In this paper, we limit our analysis to a very popular cache strategy, the *least recently used* (LRU) eviction policy. According to this policy, the least recently used item is discarded first. This approach requires keeping track of the usage of chunks by associating an *age* with each of them. When the cache memory is full and another chunk not already stored in the cache arrives, the chunk with the highest age is evicted and the newly arrived chunk is stored in the cache. If a chunk already present in the cached is requested, its age is re-set to the minimum value.

### D. SDN-based GCN operational algorithm

Fig. 1 illustrates the procedure used for managing a service request. The flow diagram shows the deployment of a solution of *problem A*, but it can be easily adapted to the other problems. Due to the centralized management of services, the GCN_OE has a complete view of available computing resources, bandwidth, as well as the current status of contents stored in NFV caches, maintained as a chunk bitmap for each cacheable file. When the GCN_OE receives a genomic service request (step 2), it classifies it according to the three service classes identified in section 0, and estimates the computing resources needed to carry out the request. This allows doing a preliminary screening of candidate datacenter tenants with enough computing resources (i.e. the subset $\mathcal{D}^*$). Once some nodes have been pruned, the GCN_OE sets up and solves the relevant optimization problem (step 3), the solution of which consists of an abstract network configuration, including the selection of the datacenter tenant where the genomic processing will be executed and in the set of NFV caches (or data repositories) from which the chunks of the VM image file and the genomic auxiliary files needed to the processing pipeline will be downloaded. This abstract network configuration maps to the network path that connects the selected datacenters with the selected data sources (caches or servers) through the set of downstream caches computed by the GCN_OE. This way, this practical configuration can be enforced on the actual infrastructure, which depends on the NFV platform used to deploy the GCN service. We divide the enforcement and configuration of SFC network path on an SDN based infrastructure in two sub-task. The first one consists of steering the content requests, which flow from requestor to the original data server, through the WAN which connects the several NFVi-PoPs where caches are hosted, in order to let these requests to reach the datacenter nodes where the cache selected by the problem solution reside. The second task is to steer the above-mentioned traffic of requests within the datacenter network so to reach the actual virtual machine hosting the caching service. Same tasks are clearly needed also for the data traffic generated from the caches to the content requestor, and through all the caches which compose the path computed by the GCN_OE as a solution of the optimization problem. After these rules have been enforced, the GCN_OE submits the processing job to the selected datacenter (step 5). This way, the GCN_LM of the datacenter in charge of

executing the processing pipeline can set up a VM with the requested computing resources and is informed of the locations from which the chunks can be downloaded. Then, the GNC_LM asks a local cache in the datacenter tenant to download the chunks of VM image file (step 6). This request is forwarded to the node indicated by the GCN_OE. Intermediate network nodes of the network slice devoted to the GCN service, and equipped by NFV caches, intercept this request according to the procedure presented in Section III.C, cache GCN data (step 7), and notify the GCN_OE of new cached contents, (step 7bis). A cache stores any new content if it has enough available free storage space, or if it can evict a stored content, by using the LRU policy. Once the VM image file is downloaded and injected into the local datacenter image repository (the Glance component in the OpenStack architecture [7] in our experiments), the GCN_LM triggers VM spawning. In the meanwhile, the GCN_LM has requested the download of the additional auxiliary genomic files (steps 8 and 9), which will be uploaded into the VM. Once the VM is ready, the GCN_OE requests also to upload the genomic dataset from the user into the VM. This step is not shown in the figure since, due to privacy issues, these files are not cached in intermediate network nodes. Upon processing is completed (step 10), the GCN_OE receives the results (step 11) and the notification that the computing resources associated with the VM have been released. Finally, output data are passed to the SaaS GCN interface, and the user is notified about their availability (step 13).

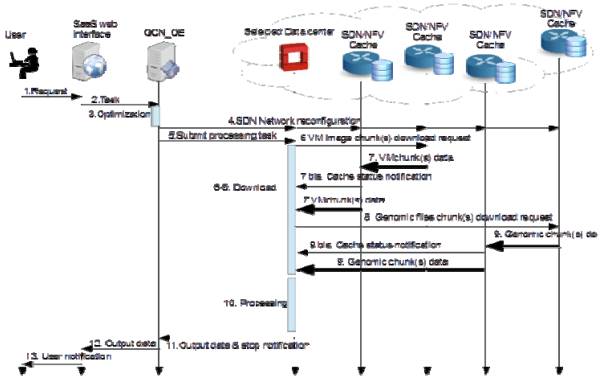


Fig. 1. Signaling flow between the main GCN entities

### *E. SDN and Application Signaling*

For what concerns SDN and application-layer signaling, we make use of the SigMA signaling framework [20], which offers a unified signaling framework addressing not only SDN devices but also applications managers (both central management systems and local instances).

## IV. Experimental Results

For what concerns *Problem B* and *Problem C*, performance verification simply consisted of verifying that the requested service and network times are respected, according to the theoretical results. As for *Problem A*, the analysis of results is multifaceted, due to the joint effects of optimization and network caching on the exchanged network traffic.

To implement the GCN system, we have ported the SDN-based content management procedures in the testbed used in [18], where we analyzed a distributed approach. It consists of three clusters of servers. Two clusters are connected by a Gigabit Ethernet link and the third cluster is connected to one of the other clusters by a GRE tunnel, which passes through a Fast Ethernet connection, which is the network bottleneck of the system. The deployed infrastructure includes 160 CPU cores, 498 GB of RAM, and 37 TB of storage space. Each server has two Gigabit Ethernet cards used for exchanging management and service traffic, respectively. Datacenters have been implemented in as simplified way, and consists of a single VM that stores the necessary contents. We have implemented the topology shown in Fig. 2, which emulates most of the Géant network. White and black circles shown in Fig. 2 represent PoPs and datacenters, respectively. Each of them is implemented in a VM equipped with 4 vCPUs, 7 GB of RAM, and 150 GB of storage space. The cache size $B$ in nodes is equal to 40 GB. Each tenant is allocated 16 CPUs, 64 GB of RAM, and 2 TB of storage space. 180 GB are used by the VM that includes both the cache NFV module and the SDN Controller. Our experiments included 20 different genomic services, each implemented by a pipeline of software programs. Each pipeline is implemented in a VM with 2 CPUs and specific RAM size from 5 to 32 GB.

Service requests are generated in user nodes that are selected randomly. Their arrival process is Poisson with mean inter-arrival time of 20 minutes. The associated content size is uniformly distributed in the rage [1, 15] GB.

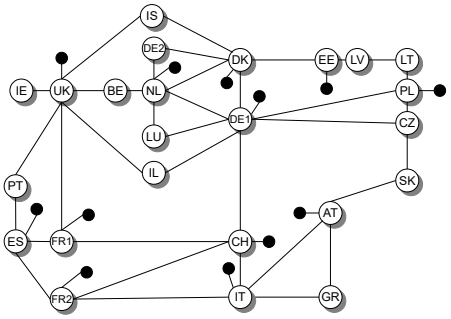


Fig. 2. Testbed logical topology, inspired to the Géant network.

The popularity of the auxiliary files follows the Mandelbrot-Zipf distribution: the probability of accessing an element at rank $i$ out of a catalog size $M$ as follows:

$$p(i) = \frac{K}{(i+q)^{\alpha}} \quad , \quad K = 1/(\sum_{i=1,\ldots,M} 1/(i+q)^{\alpha}) \tag{13}$$

In our experiment we used $\alpha$=1.3 and $q$=4 (see also ranges in [21]). We have compared the GCN approach with two simpler solutions. One of them, denoted as RAND, selects datacenters randomly, among those with sufficient resources to execute the requested pipeline. The other one, referred to as

MDU, selects the datacenter with sufficient resources staying at the minimum IP distance from the user location.

Fig. 3 shows the amount of network traffic (a) and throughput (b) generated per request, for the RAND, MDU, and GCN approaches, by using chunk sizes *c* of 1 and 2 GB. Mean values are indicated by asterisks. We can observe that GCN significantly outperforms the other solutions. Network traffic, which is the optimization metric of *Problem A*, is more than halved. In addition, the relevant throughput is more than doubled. Fig. 4 shows the effects of the cache size on the system performance. We used different *B* values, equal to 10, 20, and 40 GB, with chunk size *c*=1GB. *B* is clearly larger than the dataset managed by the most popular pipeline. An unexpected behavior emerges for what concerns the throughput performance, which seems to improve for smaller cache size. We have investigated this aspect, and the reason is that, in our testbed, some caches are hosted in external disks connected via bottleneck USB 3.0 connection. What has emerged is that by using a reduced cache size the probability of accessing these caches is smaller.

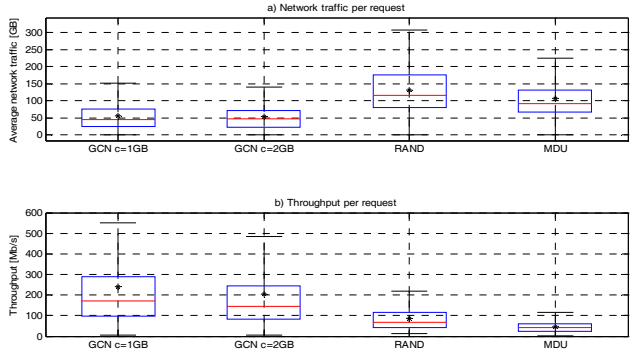


Fig. 3.Box plot with network traffic per request (a) and throughput per request (b) for GCN and baseline solutions. Cache size for GCN is B=40 GB. Asterisks indicates average values.

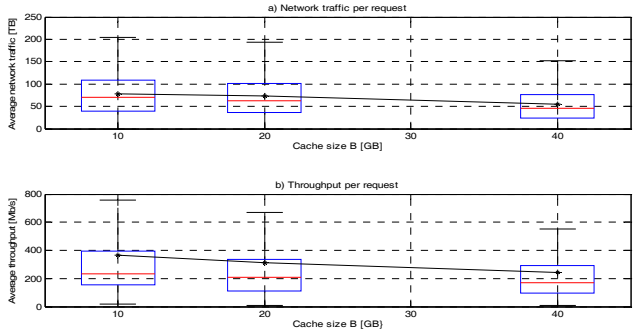


Fig. 4. Network traffic per request (a) and throughput per request (b) vs cache size. Chunk size is c=1 GB. Asterisks indicates mean values.

## V. Conclusion

In this paper, we show an SDN-based architecture for Cloud Genomics, called Genome Centric Networking (GCN), which addresses the emerging demand of genome processing services. We have formulated three different problems corresponding to three classes of genome processing services. These classes depend on the urgency of the processing results, which request the optimization of different metrics, which are traffic volume, processing time, and service time. Network performance are improved by network caching, managed by the LRU content eviction policy. Caches are implemented by NFV caching modules. An extensive experimental campaign has validated the expected achievements.

## Acknowledgment

This work is supported by the EU Horizon 2020 projects 5G-EVE (grant No. 815974) and 5G-CARMEN (grant No. 825012).

## References

[1] International Human Genome Sequencing Consortium, “Finishing the euchromatic sequence of the human genome”, Nature, 431(7011), 2004.

[2] P.C. Church, A.M. Goscinski, “A Survey of Cloud-Based Service Computing Solutions for Mammalian Genomics”, IEEE Transactions on Services Computing, vol. 7, no. 4, October-December 2014.

[3] N. Savage, "Bioinformatics: Big data versus the big C", Nature, vol.509, 29 May 2014, S66-S67, doi: 10.1038/509S66a.

[4] Stephens ZD et al., "Big Data: Astronomical or Genomical?", PLoS Biol, 13(7): e1002195. doi:10.1371/journal.pbio.1002195

[5] E.E. Schadt and al., "Computational solutions to large-scale data management and analysis", Nature Reviews Genetics, 11(9), 2010.

[6] ETSI ISG NFV, "Network Functions Virtualisation (NFV); Management and Orchestration", ETSI GS NFV-MAN 001 V1.1.1, Dec. 2014.

[7] OpenStack web site, http://www.openstack.org/.

[8] Mills RE et al, “1000 Genomes Project. Mapping structural variation at fine scale by population scale genome sequencing”, Nature. 2011 Feb 3;470(7332):59-65.

[9] Karczewski KJet al.,"STORMSeq: An Open-Source, User-Friendly Pipeline for Processing Personal Genomics Data in the Cloud", PLoS ONE 9(1): e84860, 2014.

[10] D. Kreuts, F.M.V. Ramos, P.E. Verissimo, “Software-Defined Networking: A Comprehensive Survey”, Proceedings of the IEEE Volume: 103, Issue: 1, 2015.

[11] Fusaro VA et al., “Biomedical Cloud Computing With Amazon Web Services”, PLoS Comput Biol 7(8): e1002147. (2011).

[12] Elixir project, http://www.bbsrc.ac.uk/science/international/elixir.aspx.

[13] C. Mazurek et al., "Federated Clouds for Biomedical Research: Integrating OpenStack for ICTBioMed", IEEE CloudNet 2014, 2014.

[14] Heath, Allison P et al. “Bionimbus: a Cloud for Managing, Analyzing and Sharing Large Genomics Datasets.” Journal of the American Medical Informatics Association : JAMIA 21.6 (2014): 969–975.

[15] M. Femminella, G.Reali, D. Valocchi, E. Nunzi, "The ARES Project: Network Architecture for Delivering and Processing Genomics Data", IEEE NCCA, 2014.

[16] G. Xylomenos et al., "A Survey of Information-Centric Networking Research," IEEE Communications Surveys & Tutorials, 16(2), 2014.

[17] N. Spring et al., “Measuring isp topologies with rocketfuel”, IEEE/ACM Transactions onNetworking, vol. 12, no. 1, pp. 2–16, 2004.

[18] M. Femminella, G. Reali, D. Valocchi, "Genome centric networking: A network function virtualization solution for genomic applications", IEEE Conference on Network Softwarization (NetSoft), July 2017.

[19] C.Chan, "The one-Dimensional Cutting Stock Problem", Mathematical Programming, OR630 Fall 2006, October 19, 2006. Lecture 16.

[20] D. Valocchi; D. Tuncer; M. Charalambides; M. Femminella; G. Reali; G. Pavlou, "SigMA: Signaling Framework for Decentralized Network Management Applications," IEEE Transactions on Network and Service Management, 2017, in press, doi: 10.1109/TNSM.2017.2723726.

[21] Open Source MANO (OSM) project website https://osm.etsi.org/.