Published in: Future Generation Computer Systems journal.
This version of the article has been accepted for publication, after peer review and is subject to Elsevier terms of use.
The Version of Record is available online at: http://dx.doi.org/10.1016/j.future.2015.01.017

GRASP-based Resource Re-Optimization for Effective Big Data Access in Federated Clouds

Francesco Palmieri[a,*], Ugo Fiore[b], Sergio Ricciardi[c], Aniello Castiglione[d]

[a] *Department of Industrial and Information Engineering, Second University of Naples, Aversa (CE), Italy*
[b] *Information Services Centre, Federico II University, Napoli, Italy*
[c] *Department of Computer Architecture, Technical University of Catalonia, Barcelona, Spain*
[d] *Department of Computer Science, University of Salerno, Fisciano (SA), Italy*

Abstract

Federated cloud organizations, spanning across multiple networked sites that provide both computing and storage resources, can be considered the state-of-the-art solutions for providing multi-tenant runtime services in modern distributed processing environments. In these scenarios, by re-optimizing the communication paths between virtual machines and big data sources, at evenly spaced interval or when required by circumstances, the overall communication and runtime resource utilization on the cloud infrastructure is re-balanced, so that more virtual machines can be allowed to access the needed big data sources with adequate bandwidth, thereby significantly improving the perceived performance and quality of service. The problem of re-optimization is tackled with a powerful meta-heuristic, the *greedy randomized adaptive search procedure* (GRASP), augmented by path re-linking. In order to evaluate the proposed approach, extensive simulations have been performed, leading to very interesting results, demonstrating the effectiveness and validity of the underlying ideas and their applicability to real large-scale federated cloud scenarios.

Keywords: Federated Clouds, Big Data Access, Resource Re-optimization, Meta-heuristics, GRASP.

*Corresponding author.
Email addresses: francesco.palmieri@unina.it (Francesco Palmieri), ugo.fiore@unina.it (Ugo Fiore), sergio.ricciardi@ac.upc.edu (Sergio Ricciardi), castiglione@ieee.org (Aniello Castiglione)

1. Introduction

Nowadays the proliferation of the data sources available on the Internet and the widespread deployment of network-based applications are fostering the emergence of new architectures referred as "big data", characterized by the need of capturing, combining and processing an always growing amount of heterogeneous and unstructured data coming from new mobile devices, emerging social media and human/machine-to-machine communication. The distributed nature of these resources implies remotely accessing and moving data at volumes and rates that push the frontiers of current networking technologies, so that the Internet, due to its known scalability limitations, becomes the major bottleneck for these network-based data-intensive applications. This results into an abrupt shift from the classic *application-centric* paradigm, characterized by static applications triggering on demand data transfers towards their sites, to a novel *data-centric* model, in which applications themselves are dynamically moved through the network in order to be run in the most convenient places, where enough communication capacity is available to provide effective access to their data.

Clouds are an excellent accelerator for such a shift because they make the deployment of new large-scale data warehousing architectures, supporting data processing or storage activities, extremely easy and affordable. Accordingly, most of the services based on big data processing are currently delivered in cloud style, relying on the orchestrated usage of geographically sparse data centers to satisfy their huge run-time and storage needs. Thus, cloud providers are starting to offer virtual data center services through the cloud, according to the *Infrastructure as a Service (IaaS)* model, where such a virtualized infrastructure is built on a set of federated individual tenants scattered throughout the world and interconnected via virtual end-to-end connections realized over the Internet. In order to achieve this, cloud providers use the virtualization technology, providing the abstraction of computing, storage, and network resources

from their physical counterparts (according to the server, storage, and network virtualization paradigms) so that:

- the physical server machines located in the federated data centers can run multiple *Virtual Machines (VMs)* capable of hosting their own operating systems and applications as well as even acting as virtual routers or switches;
- the storage resources available on the involved sites are managed in a totally flexible way by aggregating or partitioning them into virtual storage units or blocks providing the storage space needed by VMs. Such space can be dynamically extended or shrunk based on the changing VMs requirements.

Server and storage virtualization provides enormous advantages like elastic resource management, potentially unlimited scalability, reduced power usage, increased security/reliability and lower user downtimes. In particular, VMs can connect to big data repositories providing a virtually unlimited storage space, growing in an on-demand fashion across multiple data centers/sites and multiple physical storage systems, according to an horizontal scaling paradigm. However, implementing such a fully virtualized architecture requires the availability of enough transmission bandwidth between the sites hosting the VMs and those ones providing the virtual storage blocks they access, in order to avoid that network communication becomes a bottleneck, making access to remote big data repositories unpractical and hence invalidating the whole architectural framework. This implies setting up (or tiering down) on demand dedicated end-to-end traffic engineering interconnections between the involved sites, providing a certain amount of guaranteed bandwidth between the accessing VMs and the remote storage repositories. Such task can be easily accomplished by relying on the *Generalized Multi-Protocol Label Switching technology (GMPLS)* technology, supporting the creation of bandwidth guaranteed *Label Switched Paths (LSPs)* upon a mesh of IP-based and optical network infrastructures, that is the basic composition of the modern communication facilities such as the global

Internet as well as dedicated high-performance transport backbones/links or a hybrid mix of both the aforementioned alternatives. The creation of such traffic-engineered paths is performed at the federated cloud middleware level under the control of the resource brokering/scheduling system. However, the dynamic on-demand creation or deletion of these LSPs has the drawback of unbalancing resource usage in the long run, introducing unwanted congestion effects on several network sections. Typically this will happen on the links which are most often requested, i.e., those providing the best connections towards frequently accessed big data repositories. Precisely, the paths for new data access requests are calculated one by one, according to a bandwidth-constrained route selection criterion, from the actual status of network resource usage which includes the existing LSPs. As the access pattern evolves over time, such solutions clearly become sub-optimal since the evolution process may lead to significant changes in the aforementioned status, resulting from VMs' varying demands for access to different repositories. This may cause bandwidth unavailability towards some repositories and/or run-time resource exhaustion on some data centers so that no more accesses are allowed due to lack of network communication resources or run-time space, while a more efficient redistribution of LSPs and VMs would have allowed satisfaction of all the access requests.

Optimal behavior may be restored only by introducing a periodic offline re-optimization process in charge of rerouting some already set up LSPs over alternative paths, or moving VMs on different sites/data centers, by reclaiming the lost capacity and also achieving a load re-balance on the overall cloud infrastructure. However, re-optimization carries an unavoidable cost with it. Indeed, both LSP rerouting and live VM migration affect the service continuity and increase the network management overhead. Also, live VM migration across network boundaries implies an additional re-optimization effort due to the need of establishing a set of new LSPs towards the data repositories accessed by the involved VMs, and contextually tearing down the previously existing ones (according to a *make-before-break* mechanism), with the obvious consequence in terms of complexity on the overall resource management problem.

Thus, the offline re-optimization activity has to be handled carefully, by using flexible adaptive and effective strategies, to be used only when necessary to recover stranded capacity, by maximizing the positive re-balancing effects on the resources usage.

Starting from the above considerations, we propose an efficient re-optimization strategy for LSP rerouting and VM migration based on a greedy randomized adaptive search meta-heuristic procedure (GRASP) [1], in conjunction with a final solution refinement procedure [2][3][4][5]. The whole re-optimization approach, essentially operating within the cloud resource management framework, requires a certain degree of network visibility at the cloud middleware layer and is based on a totally flexible federated cloud model, encompassing heterogeneous run-time, storage and communication resources, in which the capacity and computing power can be independently specified for each communication link and data center site, respectively. We evaluated the effectiveness of the proposal by simulating the whole re-optimization framework on realistic cloud topologies and big data access demands, and measuring the available communication bandwidth as well as the amount of access requests toward specific data repositories that the cloud is able to support before and after re-optimization. The results from these experiments showed that this idea may lead to significant improvements in cloud resource management by acquiring the needed data from the right places, at any time, and by using the right paths throughout the underlying communication networks. Obviously, the perceivable effects on the overall cloud economy grow with both the scale of the federated cloud and its load, and consequently with the amount of data to be produced and processed.

The remainder of this work is organized as follows: after a survey of relevant related work (Section 2), Section 3 quickly summarizes the fundamental points about the characteristics of data access traffic, as well as about meta-heuristics and GRASP. Following that, the system model is stated in Section 4 and the proposed strategy and its assumptions and constraints are discussed in Section 5. Section 6 describes the simulations performed and discusses the results. Section 7 presents the conclusions and some implementation perspectives that

may lead to further performance improvements.

2. Related Work

Resource management in distributed infrastructures is a general topic that has attracted much attention in recent years (see, for example, the survey in [6] or the contributions reported in [7] for more specific cloud-related issues). With the exponential growth of data volumes to be managed, leading to the well-known big data processing problems, the location of data together with the available network communication capacity become factors of paramount importance for scheduling performance, in order to avoid unacceptably long transfer times for tasks and VMs that need to access and process huge amounts of remote data. The first experiences considering network capacity within a distributed scheduling framework have been presented in [8] and [9]. Despite considering network connectivity as a first class resource to be taken into account within the overall scheduling objectives, these experiences considered the location of data within the network as an element of secondary importance. The role of data placement in distributed scheduling was first analyzed in [10] and [11] followed by other proposals for data-aware task placement mainly targeted for grid environments [12],[13]. In particular, the framework presented in [14] considered computing and storage resources together in order to run applications on sites that are well connected to the required storage repositories.

In addition, the Data Intensive and Network-Aware grid meta-scheduler (DIANA) [15] proposal included data volumes, network conditions, and computational power information to determine an overall cost for task execution. Because clouds share many underlying concepts with grids, the ideas behind many algorithms can be easily adapted to clouds. The Traffic-aware VM Placement Problem (TVMPP) has been defined in [16], where the placement of VMs within data centers is addressed by means of a two-stage heuristic based on a grouping approach, improving the efficiency in the use of bandwidth within the data center and scalability. Alicherry *et al.* [17] considered the VM placement problem

in geographically distributed clouds, proved its NP-hardness and proposed a heuristic algorithm for it. Ghorbani and Caesar [18] addressed the problem of optimizing VM migration by proposing an heuristic that determines a "good" sequence of VM moves and a set of forwarding state changes. In addition, good results in distributed scheduling have been obtained by data-aware VM allocation methods that leverage on clustering together similar data access request [16][19]. Anyway, all these solutions are characterized by an unacceptable computing overhead, essentially due to the problem dimension/complexity in large scale scenarios. While GRASP has been already used in several challenging problems, associated to online virtual circuit routing [20] and optical network re-optimization [21], at the best of our knowledge this is the first time that this adaptive search strategy is used to solve such kind of complex network-aware resource brokering problem for data-intensive federated cloud infrastructures. GRASP demonstrates to be particularly promising for solving huge and complex optimization problems such as those concerning resource management in federated clouds, since it provides an excellent trade-off between optimality and computation time by ensuring rapid convergence to high quality solutions through a thorough exploration of the solutions space. This is essentially due to a successful combination between semi-greedy heuristics, local search and solution quality intensification techniques, that represents the fundamental advantage of the proposed framework respect to many of the existing approaches available in literature, both in terms of solution quality and re-optimization process efficiency.

3. Background

3.1. Data transfer dynamics in federated clouds

The traffic patterns observed within production data centers and between data center sites in federated distributed computing environments have been found [16] to possess remarkable characteristics, among which the two most interesting in the context of this work are:

- the distribution of traffic between VMs is uneven;
- the traffic demand tends to be relatively stable over time and concentrated towards specific sites, assuming the role of big data repositories.

The first observation indicates that only a few VMs are responsible for a substantial portion of traffic. Such VMs, apart from those running highly popular network-based application or services, accessed by hundred of thousand users, are typically those involved in big data processing activities. In fact, the large-scale aggregation and partitioning practices, characterizing cloud-driven big data processing systems that operate by using the Map-Reduce model, need huge communication bandwidths in order to efficiently manage the propagation of big data chunks among a significant number of sites, required for performing data aggregation tasks between mapper and reducer VMs. That is, the intermediate results incoming from a large number of mapper VMs running on multiple data center sites scattered throughout the network, each one managing up to petabytes of data, are typically aggregated within a properly selected data center for performing the needed `reduce` task on some specialized VMs. Accordingly, the overall data transfer volume for these Map-Reduce-based data processing applications ranges in the order of hundred or thousand gigabytes with an exponential growth rate trend. As an immediate consequence, the above data transfer activities between geographically distant sites within a federated cloud organization, where site-to-site communications are enabled by traditional Internet connections may imply unacceptable times. While most of the existing infrastructures essentially operate in a network-oblivious way, by completely ignoring most of the inter-site communication aspects involved in the overall resource brokering activity, the bandwidth necessary to effectively transfer such large data volumes to the requesting VMs distributed across multiple sites becomes a critical factor to be handled very carefully at both the transport and resource management levels of the cloud infrastructure.

Consequently, a specific optimization strategy is strongly necessary to guarantee the distributed access to big data in federated clouds. Migrating only

the VMs that generate high traffic on the sites that are best connected to the data repositories could be regarded as a promising strategy. However, as it will be shown in detail later, live migration of VMs does not come for free in federated environments extended across wide geographic areas. It is also evident that there are some VMs that communicate more intensely with other machines/sites in the same group than they do with machines in other ones, and in particular, VMs devoted to data intensive tasks are characterized by strong communication patterns with the sites providing or archiving the data of interest. This implies the emergence of specific areas of access aggregation around data repositories to be leveraged through proper VM placement strategies as well as by building virtual network topologies that aim at enforcing the needed bandwidth requirements and constraints. It should also be considered that the demand parameters used to guide the initial allocation can be safely used also in subsequent phases, when re-optimization is underway.

In conclusion, there is a strong motivation for attempting to jointly optimize the placement of VMs across data centers and the virtual network topology by taking into account the network features together with the bandwidth requirements, so that better utilization of network resources and increased scalability can be achieved. Within an incremental scenario where data access request come sequentially and are serviced as they arrive, network conditions change as VM allocation is accomplished, as a result of the reservation of bandwidth for the new connections to be established, thus influencing the allocation of subsequent LSPs. This may lead to uneven and unbalanced network and computing resource distributions that should be corrected by re-organizing the logical (virtual) network layout and, if necessary, migrating some VMs.

3.2. The GRASP Meta-heuristic

The Grasp meta-heuristic is a multi-start iterative procedure that operates in two phases. Since its inception in 1989 [22], it has found application in a variety of complex optimization problems. The first phase of GRASP is a greedy randomized generation of a feasible solution in the search space (construction

phase). The neighborhood of that solution is explored in the second phase, through a local search procedure which aims at finding a better solution. The entire procedure is repeated until either an acceptable solution is found or a predefined number of iterations is completed.

In detail, the procedure starts from an empty solution and works iteratively by adding a new element at each iteration in order to obtain a complete feasible solution. The choice of this new element depends on the order with which potential candidates are stored into a specific list. This order reflects a predetermined criterion based on a greedy "benefit" associated to the selection of the above element. Such benefit, however, is not constant, but adaptive in the sense that it is updated at each iteration of the construction step in order to take into account the selection of the elements occurred in previous iterations. The probabilistic component of the construction phase mainly acts in the choice of the element to be added to the solution, since such choice falls on a randomly chosen element in a restricted list of the best candidates (RCL), ordered according to the above benefit. Such randomization is also exploited in order to attempt escaping local minima. The size of the RCL is limited by a specific tunable parameter. The local search phase that follows the construction of the feasible solution aims at further improving its quality, finally depending in a synergistic way on both the phases. The construction algorithm is executed multiple times, generating each time a new solution, but only the best solutions are taken as the starting points for the local search. Natively, the GRASP procedure is memory-less: a solution found at an iteration has no influence on the next one. However, in this way, valuable information gained about the solution space is at risk of being lost at each iteration. Path re-linking attempts to take profit of randomization while at the same time using good solutions found in previous iterations by keeping a set (elite set) of good solutions and using them to guide the exploration of search space. This introduces memory in the original GRASP scheme in order to drive the search towards better solutions by intensifying the search activity in the most promising areas of the solution space. It can also operate on a set of available high quality solutions by recombining them in order to generate new

and better ones.

4. Modeling Data Access Dynamics in Federated Clouds

Let us consider a set $S = \{s_1, \ldots, s_{|S|}\}$ of *cloud sites*, where computing and/or storage resources are physically resident. The data available on the cloud storage resources are organized into a set $D = \{d_1, \ldots, d_{|D|}\}$ of big data repositories, with $D \subseteq S$. These repositories must be effectively accessed by several VMs that must be properly scheduled for running on another set $R = \{r_1, \ldots, r_R\}$ of geographically sparse *computing sites*, with $R \subseteq S$, providing computing resources for VMs within the federated cloud. Storage and computing resources may co-exist in the same sites so that the intersection between R and D may be not empty ($R \cap D \neq \emptyset$). The total quantity of runtime resources located in the site r_j, expressed in generic computational power units, is denoted as p_j. Analogously the amount of storage blocks available on the repository d_j will be expressed in generic storage block units and will be denoted as q_j. Each individual virtual machine m_i in the set $M = \{m_1, \ldots, m_M\}$ of VMs operating within the federated cloud, is characterized by a computing demand χ_i, a lifetime t_i and runs on a computing site $r_k \in R$, where $r_k = \sigma(m_i)$ is individuated by the scheduling function $\sigma : M \mapsto R$. Clearly, its is necessary that:

$$\sum_{i,k:m_i \in M | \sigma(m_i) = r_k} \chi_i \leq p_k. \tag{1}$$

In addition, a data access profile $\Delta^i = \left\{\delta^i_1, \ldots, \delta^i_{|D|}\right\}$ is associated to each VM m_i specifying its data access demand (in Gbps) towards all the data repositories in D. It is straightforward considering that $\delta^i_k \leq q_k \ \forall \ m_i \in M, d_k \in D$. The computing and storage sites communicate each other by using semi-permanent end-to-end connections provisioned through traffic engineering practices over a physical network infrastructure that is quite heterogeneous in communication technology, coverage and capacity, ranging from dedicated optical fiber strands or WDM channels, leased virtual lines on public or private packet switched infrastructures, to the more generic Internet access connections. We can model

such connections as a full mesh of direct end-to-end virtual links between each pair of sites s_i and s_j characterized by a total bandwidth capacity $c_{i,j}$, determining the maximum quantity of bits per time unit that can potentially pass across the connection from i to j by using the underlying transport infrastructure. This is represented by the matrix $\mathbf{C}$, that can hence be considered a complete representation of the network-based data-access capacity constraints. We also assume that there are no "isolated" sites, that is, all the data center sites are provided with a non-zero capacity network connection: $\forall(j,k), c_{j,k} \neq 0$, with $j \neq k$. Furthermore, in order to represent the co-location of storage and computing resources on the same site, an infinite capacity value will be used in the main diagonal of the network capacity matrix $\mathbf{C}$, so that $c_{i,j} = \infty \; \forall i = j$. This models the case of virtual machines and accessible storage repositories that are placed on the same LAN infrastructure, with the consequent benefits in terms of both access speed and network resources availability. Clearly:

$$\sum_{i,k:m_i \in M | \sigma(m_i) = r_k} \sum_{j:d_j \in D} \frac{\delta_j^i}{t_i} \leq c_{k,j}. \tag{2}$$

Obviously, an occupation matrix $\mathbf{A}$ is associated to the capacity matrix $\mathbf{C}$, so that $a_{i,j}$ will represent at any time the currently allocated capacity between sites s_i and s_j, with $a_{i,j} \leq c_{i,j} \; \forall s_i, s_j \in S$. Finally, a set $\Lambda^i = \left\{\lambda_1^i, \ldots, \lambda_{|\Lambda^i|}^i\right\}$ of LSPs is associated to each runtime site $r_i \in R$, describing the active paths from that node to $|\Lambda^i|$ repository sites in D, with $|\Lambda^i| \leq |D|$, each characterized by a minimum guaranteed bandwidth demand $\beta(\lambda_h^i)$ so that:

$$\beta(\lambda_h^i) = \sum_{i,k:m_k \in M | \sigma(m_k) = r_i} \sum_{j:d_j \in D | arg(d_j, \Lambda^i) = h} \frac{\delta_j^k}{t_k} \tag{3}$$

where $arg(d_j, \Lambda^i)$ determines the position of the element associated to the data site d_j in the set of LSPs Λ^i.

An instance of the federated cloud status, can be represented by a set of $|M|$ 4-uples $\xi = \left\{(m_1, \sigma(m_1), \Delta^1, \Lambda^{x_1}), \ldots, (m_{|M|}, \sigma(m_{|M|}), \Delta^{|M|}, \Lambda^{x_{|M|}})\right\}$, where $x_i = arg(\sigma(m_i), R)$, considering the VM allocation on the computing sites $r_{x_i} \in R$, together with their bandwidth demands towards the data repositories in D that are enforced through a set of LSPs carved on the underlying

network. Clearly, each solution ξ depends on the sets M, Δ and Λ, as well as on the mapping function $\sigma(\cdot)$, but in order to keep the notation more simple we do not directly highlight such a functional dependency. The re-optimization process must start from the current status, reached as a consequence of the dynamic online allocation of the available VMs one by one, and of the establishment of the associated LSPs needed to satisfy their access demands to big data repositories. It should try to rearrange the overall set of LSPs Λ^{x_i} and the VM allocation function $\sigma(\cdot)$ in order to re-balance the communication and runtime resources usage and maximize the number of future VM allocations on the cloud. This is clearly a complex optimization problem that has to simultaneously consider the VM scheduling goals on the cloud sites and the LSP routing on the underlying communication network. Accordingly, it needs to maximize the benefits introduced by recovering residual communication capacity by simultaneously containing the adverse effects on the overall VM allocation in order to avoid service disruption and other issues/problems associated to live VM migration. Coping with such objectives, that are most often independent and sometimes conflicting, implies the harmonization of bandwidth management and network engineering strategies together with distributed VM scheduling policies considering all the runtime resources available on the federated cloud. The corresponding optimization problem, known as the Reconfiguration Sequencing Problem for virtual topologies is known to be NP-hard [23]. Consequently, a new re-optimization strategy that reorganizes all the currently active LSPs (without minimum disruption) in oder to re-balance the underlying transport network usage, by redistributing VM load on the available sites and freeing sufficient available communication capacity between computing and data repository site pairs, has to be determined by using some heuristic practices that introduce minimum run-time overhead. Accordingly, we propose a new heuristic-based adaptive re-optimization strategy, to be activated on a periodic or event-triggered basis. Such strategy is based on a straightforward greedy randomized approach to iteratively generate new candidate solutions whose overall quality is constantly improved during the re-optimization process by relying on properly crafted local search and elite

selection (path-relinking) schemes.

5. Re-optimizing Federated Cloud Infrastructures

The aforementioned re-optimization process should be activated periodically, or each time a tunable cloud resource utilization limit is passed, in order to make the maximum amount of free communication bandwidth towards big data repositories available to cloud run-time resources, by introducing the minimum amount of disruption or performance degradation to the currently operating cloud services. Managing the cloud resources during the re-optimization process is a quite complex and harmful task, since both virtual topology reorganization and live VM migration, without being associated to particular routing and VM management practices (e.g. path pre-signaling and fast switch as well as address space extension for VM migration) may significantly affect several critical cloud operation activities, and therefore must be handled very carefully. A key feature of the proposed heuristic re-optimization strategy is the capability of achieving maximum benefits from LSP paths reorganization, before performing any change in VM allocation, that should be considered as a last-resort option to be used only when LSP optimization is no more a feasible alternative for minimizing the adverse effects of VM live migration.

5.1. Basic Assumptions and Operating Constraints

We assume that the initial placement of VMs is decided either by a capacity planning tool, seeking to minimize the consumption of physical computing resources [16], or by an ad-hoc meta-scheduling algorithm adopted by the federated cloud infrastructure. This procedure should take care of constraints as well as of the "distance", in terms of network throughput, separating the big data repositories from the VMs. There are many solutions in the literature, facing with this challenge e.g. [19]. We further suppose that runtime requests for new VMs, with their own data access demands, continue to arrive while the previous ones are serviced. Consequently, new data access requests are served

incrementally, by determining where the new VMs should be placed and enforcing the access bandwidth constraints towards the data repositories of interest, by properly setting up a set of LSPs between the involved sites. Such LSPs are routed on the transport network one by one according to a bandwidth constrained *shortest-path-first* (SPF) strategy where each path selection only takes into considerations the communication resources allocated so far (by the already existing LSPs). Thus, the status of the federated cloud organization will change from a service request to the next, and the dependency on the order of arrival is likely to lead to waste of both computing and communication capacity. Consequently, after a certain number of successful service requests, resource fragmentation will occur on computing sites and simultaneously, congestion phenomena due to communication resource exhaustion will be experienced towards the most accessed data repositories, strictly requiring a reorganization of both communication paths and VM allocations, to be handled within the context of the aforementioned re-optimization process. In theory we have three possible options that can be used in performing such a re-optimization:

1. moving – or replicating – the data in order to bring them closer (from the network throughput point of view) to the VMs which need them;
2. moving the VMs so that available bandwidth can be used to allow them to reach the data they need;
3. re-arranging the virtual topology constituted by the LSPs needed to enforce the bandwidth constraints between computing and data repository sites.

Unfortunately, we can immediately discard the first option by considering that big data cannot be easily moved nor replicated on an Internet-wide scale, due to the enormous amount of storage space and the huge bandwidth needed over geographical connections to perform replication between sites. Apart from consideration of sheer volume that advises against replication, it has been found that replicating big data can have adverse effects, particularly as far as larger/more complicated hybrid clouds were concerned [24]. In fact, all the well-known *4*

V's properties (Volume, Variety, Velocity, and Variability) characterizing big data make bulk replication senseless in presence of both heterogeneous access patterns from the VMs and of a certain degree of variability/volatility that affects several data sources (e.g,. in real time stream processing/analysis of sensor data). Consequently, this option will not be considered at all in our re-optimization strategy.

Regarding the second option, we emphasize that that VMs cannot be easily moved across sites without introducing perceivable service disruption. While the mobility of VMs within the same data center has no disadvantages and introduces valuable benefits in terms of reliability, load balancing and agile workload distribution, a massive use of the live migration functionality over geographically distant places can not be regarded as an inexpensive operation, because it has its own consequences that should be accurately evaluated. First of all, Service-Level Agreements (SLAs) significantly constrain the freedom of moving VMs across distant sites. Such SLAs specify the computing power that should be made available to the tenant by the cloud service provider. Agreements often include directions for system parameters, that need to be in a narrow range of values to ensure that VMs can be seamlessly moved. For example, the number of available cores per single server might be required to be set at a certain value. This is most important in applications that handle parallelism at the application level and explicitly partition the computation over cores. While it is easy to comply with the SLAs by using servers located in the same data center, guaranteeing the desired uniformity with servers from widely different locations in the world, activated at different times, with different local legislation, raises significant concerns. Servers are indeed heterogeneous under various respects, including for example the processor type, the amount (and speed) of RAM, and the degree of control over the operating frequency in order to save power. Obviously, also powerful servers support smaller-equipped VMs. However, this is undesirable, because the residual computing power on the server may limit the server's ability to run high-end VMs for other tenants, consequently leading to loss of revenue. Other SLAs govern the contractual relationship between the

cloud service provider and the power supply operator. Power supply operators are interested in obtaining reliable consumption forecasts that will allow them to plan their networks accurately, on the basis of the expected load, reducing the need for over-provisioning. In this view, often SLAs set out a threshold level of power draw. Until the draw stays under the threshold, power will be provided at a low tariff to the customer. Draws exceeding the threshold will, instead, trigger higher rates. The wide variety of power supply contracts in various locations is another factor that would restrict the ease with which VMs can be moved to different places. In addition, climatic differences in different regions also may impact the efficiency attainable with cooling systems: colder areas may benefit of a reduced cooling capacity. To complicate things further, all the above consideration do not mention the availability of "green" power sources, which may have higher costs but could nevertheless be preferable with respect to traditional sources, due to their reduced environmental footprint. Second, also routing issues can severely affect live VM migration across geographical network boundaries. Performing live migration implies moving a VM from a physical data center to another one such that the critical network state information, including IP and MAC address, is preserved. This is needed to keep existing TCP connections from being stopped and restarted, with obvious advantages in terms of disaster recovery, business continuity, infrastructure scalability, etc. One of the limitations with live migration today is that the VM can only be migrated to locations within its own IP subnet (hence over the same LAN), and this is not acceptable within a truly scalable multi-tenant federated cloud organizations, where the VM should be able to migrate to any location, independent of the subnet boundaries. Supporting live VM migration to any other location across subnet boundaries, without the need of any change to VM network state and without service disruption can only be accomplished by extending the source LAN to the new location through sophisticated tunneling techniques. All the packets to/from the moving VM must be transmitted through a bi-directional tunnel between the destination network and the source one. The VM must be deactivated on the source site and reactivated on the des-

tination one by simultaneously and dynamically reconfiguring the above tunnel. In order to make the VM migration process completely transparent (to both the VMs' applications and the outside users), the migration environment of the VMs should be regarded as a mobile network environment, and the migrating VM is regarded as a mobile node. Unfortunately, virtual local area networking (VLAN) and virtual private networking (VPN) techniques that are traditionally expected to support such migration, mobility, and interconnections have too many limitations in providing seamless and effective address space extension and mobility support between sites. Problems are exacerbated by the presence of middleboxes (e.g., load balancers, firewalls). Once a migration has taken place, both the inbound and outbound traffic of the VM has to be re-routed efficiently. However, middleboxes will still need to be traversed after a migration, thus enjoining the use of shorter, more efficient network paths in favor of unnecessary long ones. Consequently, for all these reasons live VM migration must be considered as a last-chance option in our re-optimization strategy, to be used only when no other options are available for recovering residual capacity.

Finally, virtual topology re-optimization only implies the rerouting of the existing LSPs one at a time. Such operation is supported by well known and reliable routing practices (e.g. constrained SPF routing) also relying on robust control plane signaling mechanisms providing fast switch between the previously existing and the newly established LSP with minimal service disruption.

5.2. GRASP-based federated cloud re-optimization

With the aim of easily determining good-quality solutions to the aforementioned re-optimization problem, we introduce a new scheme based the well-known GRASP exploration technique. In order to correctly implement a GRASP solution search procedure, we need to first consider a certain number of basic choices associated to the specific structural features of the involved problem. First of all, a greedy selection criterion must be established in order to drive the iterative process leading to the construction of feasible solutions, by performing one basic choice at a time. Such criterion has to work in a fully adaptive

way so that its behavior should progressively change with the addition of each new choice in the solution under construction by reflecting the effect of each new decision being taken. Furthermore, a restriction strategy must be determined for the construction of the restricted candidate list (RCL), from which on each step the candidate element to be introduced within the partial solution is extracted. Finally, a probabilistic selection policy has to be established in order to extract random elements from the restricted candidate list, together with the fundamental architectural choices associated to the local search process (the objective/cost function to be used for evaluation, the neighbor selection criteria and the search strategy). In particular, the objective function is needed to rank and compare the overall quality of the solutions under examination, by properly considering some of the fundamental system variables that are needed to drive the re-optimization process towards the best solutions. Hence its value must be minimized on each iteration by also guaranteeing the respect of all the problem constraints. The meta-heuristic re-optimization process starts from the current cloud configuration represented by its actual status $\xi = \left\{(m_1, \sigma(m_1), \Delta^1, \Lambda^{x_1}), \ldots, (m_{|M|}, \sigma(m_{|M|}), \Delta^{|M|}, \Lambda^{x_{|M|}})\right\}$ encompassing all the running VMs. Such initial status represent a feasible solution to our re-optimization process, where the generic element describes the allocation/scheduling site of choice $r_{x_i} = \sigma(m_i)$ (remember that $x_i = arg(\sigma(m_i), R)$), associated to the VM m_i, together with its set of data access demands Δ^i, and LSPs Λ^{x_i} defining the portion of virtual network topology related to m_i. We reconsider all the currently running VMs by selecting them one-by-one according to a specific greedy criterion and hence build stepwise a new solution:

1. by rerouting the associated LSPs Λ^{x_i} towards the repositories in D, according to a constrained SPF logic based on the associated demands Δ^i and on the currently available communication resources;
2. by re-locating the involved VM when we cannot attain a feasible solution (routing of all the LSPs in Λ^{x_i} is not possible due to resource exhaustion).

Therefore, rerouting each of the LSPs in Λ^{x_i} or performing a reorganization of the VM schedule function σ only depends on the network status at the moment the LSP is considered for routing and on the current VMs displacement on the computing sites. Note that, if the currently running VMs are serviced in sequence, by starting from an empty cloud transport network and a given resource occupation status, the selection order uniquely determines the solution. We also note that in order to avoid as possible the problems associated to live VM migration, and hence minimize the VMs movements across sites, we consider the reorganization of the VM schedule σ as a last-chance option, to be used only when a re-optimization based solely on virtual network topology rearrangement is no more possible. The whole procedure operates in a maximum number of iterations to build new solutions according to the aforementioned stepwise component selection strategy, by exploring on each iteration the neighborhood of the currently determined solution to escape from local optima and relying on the cost function to look for improvements in the best solution found so far. To do this, after each iteration we compare the cost of the current solution by using the objective function defined in eq. (6) with the best one obtained until now, always saving, for comparison in future iterations, the lowest cost solution found so far.

5.2.1. Building Candidate Solutions

When generating new candidate solutions, the status 4-uples resulting from the initial allocation are considered sequentially, one at a time by selecting them according to a specific greedy criterion. Thus, the LSPs Λ^{x_i} associated to each 4-uple are re-routed one at a time, starting from an initially empty network and given the current VM schedule σ, thus building a candidate solution. If a specific configuration is reached where one of the LSPs belonging to Λ^{x_i} cannot be routed any more, construction is restarted from scratch by moving the involved VM through its reallocation in the site $\sigma(m_i)$ providing maximum currently available computing capacity. The greedy criterion assigns the highest preference to the 4-uples characterized by the largest value of a combination

$\varphi(m_i)^w \cdot \vartheta(m_i)^{1-w}$, where

$$\varphi(m_i) = \frac{1}{|D|} \sum_{j:d_j \in D} (c_{\sigma(m_i),d_j} - a_{\sigma(m_i),d_j}) \tag{4}$$

is the average residual bandwidths between the site $\sigma(m_i)$ hosting the involved VM m_i and all the data repositories $d_j \in D$, and

$$\vartheta(m_i) = \frac{1}{|D|} \sum_{j:d_j \in D} \frac{\delta_j^i}{t_i} \tag{5}$$

is the average value of the bandwidth requested by all the VMs for connecting to the repositories (data access demands). Note that w is a tuning parameter modeling the relative weight of residual bandwidth with respect to demand. The most appropriate value for w is to be found empirically. Such strategy privileges the selection of the access requests characterized by greater bandwidth demands and involving communication resources offering higher residual capacity. This clearly brings a re-balancing effect in the overall resource allocation, that is no more influenced by the order in which the original access requests have been serviced, that caused the creation of bottlenecks as well as resource fragmentation and hence resulted in a sub-optimal allocation strategy. Note that the above greedy selection criterion is adaptive: the 4-uples list may be reorganized in dependence of the rerouting of the LSPs associated to the candidate solution given the current VM schedule σ, since each successful routing operation reduces the residual bandwidth available between the source site and the involved repository so that the the greedy selection behavior continuously changes depending on the current bandwidth available on each link resource along the LSP path. To reap the benefits of stochastic sampling, a restricted candidate list (RCL) is maintained, containing only a subset of the 4-uples ordered according to the above greedy criterion. One element at a time is selected from the RCL with uniform probability, so that diverse solutions can be obtained at each GRASP iteration. The size of the RCL is an important parameter that can influence the system behavior. Relatively long RCLs will make the random component prevail, with a wide diversity of solutions, while short RCLs will reduce the variability across

solutions that are found. In the extreme case that the RCL had length one, only the request entailing the highest bandwidth would be considered, leading to the production of the same solution at each iteration. Classically, two methods have been applied to decide the inclusion in the RCL:

- *cardinality-based* methods include in the RCL only the q best candidates, where q is a fixed number;
- *value-based* methods include all the candidates whose value lies in an interval, whose upper end is the value of best candidate, while its amplitude is an adjustable fraction of the total range between the best and the best candidates.

The most appropriate value for the RCL size in a given instance depends on the distribution of the required bandwidths. If that distribution is uniform, the first few values in the list are likely to capture enough of the variability needed to ensure an adequate level of diversity in the solutions. If that distribution is, however, skewed toward high (respectively, low) values, then a smaller (respectively, higher) number of candidates will be needed. Consider the former case: a significant number of candidates on top of the list will share substantially similar values of the bandwidth. Selecting a fixed number of candidates will leave out of the RCL a number of "good" candidates. On the other hand, in the latter case, the top of the list will be formed by a small number of candidates with high bandwidth, followed by a bunch of candidates with considerably smaller bandwidth values. Here, "bad" candidates will likely make their way to a fixed-size RCL. The value-based alternative will not suffer from these problems. There will still, however, be the risk that, with heavily skewed distributions, RCLs could be produced that are excessively long or too short, to the detriment of efficiency or solution diversity. In addition, a threshold depending on the entire range from the best and the worst candidates could be too loose if that range is wide, and too strict if the opposite is true. We chose an hybrid, adaptive approach to dynamically adjust the RCL size to the distribution of the required bandwidths. To this end, two parameters q_{min} and q_{max} were set, to specify

the minimum and maximum allowed lengths for the RCL. Then, *hierarchical clustering* was applied to the ordered candidate list, and elements from the top-ranked clusters were sequentially added to the RCL until the list size gets into the allowed range. In case the addition of all elements in the last added cluster would make the RCL exceed its maximum allowed length, only the first elements of the last cluster are added, so that the final RCL contains exactly q_{max} elements. The homogeneity of data belonging to the same cluster ensures that substantially comparable alternatives are given the chance to be selected by the random choice.

5.2.2. Local Search

The solutions provided by the construction phase may not be optimal. Local search will explore variations in these solutions, attempting to improve their quality. Ensuring a priori that the solutions $\xi, \xi', \xi'', etc.$, generated by each GRASP iteration are locally optimal it is practically impossible, so that any previously determined solution can be potentially improved by performing a local search step in the space of its neighbor solutions. Local search works iteratively by sequentially replacing the candidate solution with a better one found in its own neighborhood, typically differing from the former one in a single decision/choice or "move" in the solution space (e.g. the rerouting of an individual LSP). However, a blind local search often presents the drawback of attracting the exploration process into local optima. To avoid this risk, local search must be driven by properly crafted heuristic strategies, carefully choosing the exploration starting point as well as the neighborhood space exploration criterion. In our specific problem, the neighborhood exploration in the current solution space is achieved by rerouting one LSP $\lambda_j^i \in \bigcup_h \Lambda^h$ at a time. Any change in VM mapping function $\sigma(\cdot)$ is not considered in local search since it potentially introduces substantial modification to the solution, bringing the search process out of its neighborhood space (and hence local scope). The objective function considered for evaluating the quality of neighbor solutions considers the potentiality of the infrastructure to improve access to big data

repositories for all the currently running VMs, that will be approximated by considering the total communication bandwidth still available to them:

$$cost(\xi) = \sum_{m_i \in M} \varphi(m_i). \tag{6}$$

A list ρ containing a predetermined number $LSize = |\rho|$ of the best rerouting actions, in terms of the objective bandwidth gain obtained, is maintained. $LSize$ is a system parameter controlling the amount of resources to be to be committed to the local search phase. A rerouting action may involve a single LSP, or an ordered sequence of LSPs that are re-routed one after another. For the initial population of the list, all the currently active LSPs λ_j^i are sorted in decreasing order of their bandwidth demand $\beta(\lambda_j^i)$, and the gains in the objective function resulting from their rerouting are used as a sorting criterion for insertion in the ρ list. Once all $|\rho|$ entries in the list have been filled, further solutions are generated by taking the best set of actions $\tau \subset \rho$ and considering LSPs for rerouting in the network status where actions in τ have been performed. If the resulting move yields a gain better than the lowest-ranked action, it will replace it the ρ list. The process continues until no further improvement is possible, and the best solution determined so far is returned. This results in a depth-first search of a properly determined portion of the solutions space of each candidate solution ξ.

5.2.3. Path Relinking

Path relinking is an elite selection strategy whose goal is selecting the highest quality elements to be added in new solutions, by using these elements to determine other solution that most significantly improve the current one. In doing this, it starts from a population of elite solutions by combining them in order to generate new and better ones. By starting from these elite solutions, that are the most promising ones found so far, paths in the solution space leading towards other elite solutions are thus generated and explored in the search for better solutions. At least a guiding solution must be determined for generating such paths in the solution space. The generation of new solutions is accomplished

by selecting moves (i.e., LSP routing actions, but not VM rescheduling ones, as in local search) that introduce the elements contained in the guiding solutions. This may be viewed as a strategy that seeks to incorporate the components of high quality solutions, by favoring them in the selected moves. When the set of elite solutions reaches its maximum dimension $MaxElite$, for each solution ξ found within a GRASP iteration, one of the elite solutions ε is selected and path relinking is accomplished on the pair of solutions (ξ, ε), otherwise the solution ξ is simply inserted into the elite set. Anyway, also if the elite set is full and ξ is better that the worst of the elite solutions, then the worst solution is replaced. Path relinking on the pair (ξ, ε) is accomplished by starting from the initial solution ξ by progressively inserting into it new components taken from the guiding solution ε until ξ becomes equal to ε. This results in the set of actions that has to be applied to candidate solution ξ to reach the guiding one ε. Then, by using the solution ξ as a starting point, the best change options that have not yet been considered are tried one by one, until the guiding solution is reached. The best solutions determined during such process, that can be seen as a path in the solution space, are also kept as new candidates to be inserted in the elite set. The entire process is iterated until further improvements in the elite set are possible.

6. Performance Evaluation

An extensive experimental study has been performed, by using an ad-hoc discrete event simulation environment realized in Java® and running on a 3.07 GHz Intel® Core^{TM} i7-950 CPU @ with 16 GB RAM, in order to evaluate the performance of the proposed re-optimization scheme.

6.1. The Simulation Environment

The Geant2 pan-European research network has been taken as a real-world reference topology for the simulated multi-site federated cloud organization used in our experiments. It consists of 34 nodes, connected as shown in Fig. 1,

Algorithm 1 procedure GRASP $(MaxIter, w, q_{min}, q_{max}, LSize, EliteSize)$

1: $Candidates \leftarrow \xi$ {ξ is the set of 4-uples characterizing the current status}
2: $BestSolution \leftarrow Candidates$
3: $Iterations \leftarrow 0$
4: $EliteSet \leftarrow \varnothing$
5: **while** Iterations $\leq$ MaxIter **do**
6: $Solution \leftarrow \varnothing$
7: **while** Solution not feasible **do**
8: RCLsize $\leftarrow$ DetermineRCLSize($Candidates, q_{min}, q_{max}$)
9: RCL $\leftarrow$ MakeRCL($Candidates$, $RCLsize$)
10: $s \leftarrow$ RandomElement(RCL)
11: $Solution \leftarrow Solution \cup s$
12: $Candidates \leftarrow Candidates \setminus \{s\}$
13: UpdateResourceAllocation($Candidates$)
14: **end while**
15: $RemoveRedundantElements(Solution)$
16: $NewSolution \leftarrow$ LocalSearch($Solution$, $LSize$)
17: **if** $|EliteSet| = EliteSize$) **then**
18: $NewSolution \leftarrow$ PathRelinking($NewSolution, EliteSet$)
19: UpdateElite($NewSolution, EliteSet$)
20: **else**
21: $EliteSet \leftarrow EliteSet \cup NewSolution$
22: **end if**
23: **if** cost($NewSolution$) $<$ cost($BestSolution$) **then**
24: $BestSolution \leftarrow NewSolution$
25: **end if**
26: $Candidates \leftarrow BestSolution$
27: **end while**
28: **return** $BestSolution$

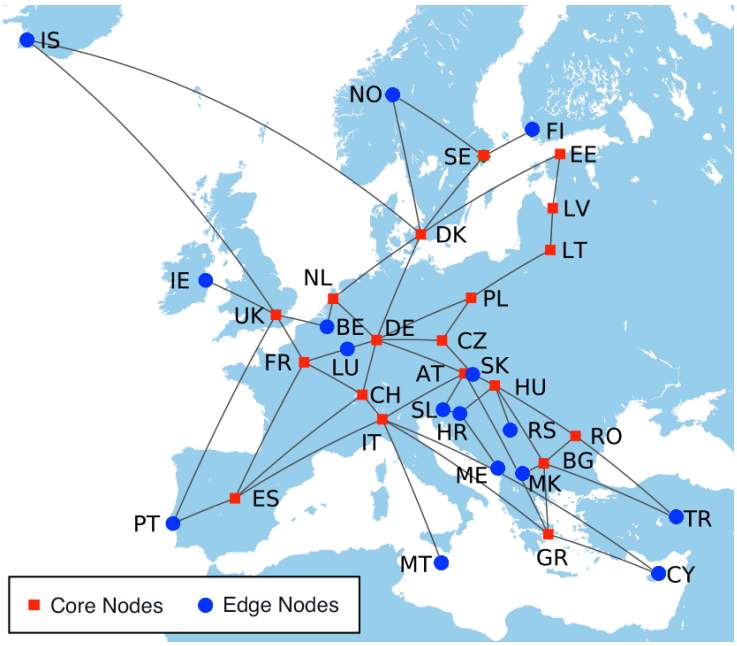

Figure 1: The GEANT-based simulated federated cloud topology.

each one providing both runtime resources and storage repositories. Nodes are equipped in different ways with central (and most strategic) ones, realizing the core infrastructure, providing plenty of storage space, made available through multiple high speed network connections.

VM connection request towards big data repositories are generated by randomly selecting pair of nodes, respectively assuming the role of runtime site, hosting the requesting VM, and storage site, providing the data needed by the VM. Each request is characterized by a specific lifetime (random time units) as well as by a minimum guaranteed bandwidth (randomly chosen 100 Mbps units) and triggers the establishment of a corresponding LSP (providing the bandwidth requested over a properly crafted traffic engineered path) between source and destination sites. LSP connections are established and torn down as the requesting VMs start or terminate their data access activities, allowing the simulation of a fully dynamic cloud environment. While data access requests, to be served as they arrive, are randomly generated without any prior

knowledge on the involved node pairs, central nodes, belonging to the core, have a significantly higher probability to be selected as data repositories than the ones located on the edge of the cloud. That is, the availability of storage space influences the selection of big data sources, incrementing their chances to be accessed by VMs in their activities/requests. The cloud has been progressively loaded by starting from an empty infrastructure and continuously adding new access requests, until 1000 simultaneous requests from active VMs to big data repositories are present, by observing the number of requests that can be simultaneously satisfied (in other words the service rate), and performing a complete re-optimization every 400 new active requests, in order to cope with the saturation phenomena that cause the cloud to reject (service blocking) further access request when no more bandwidth towards big data repositories is available. In order to focus our attention exclusively on the big data access problem, that is the main goal of this work, we assume that VMs are already running on their source runtime sites, and if a rescheduling (VM movement between sites) is needed during the re-optimization process, since no further LSP rerouting steps leading to a feasible solutions are possible, some runtime space is always available on each computing site, so that the VM can be simply rescheduled on the first site providing available bandwidth according to a *first-fit* scheme. We also assume that LSPs are rerouted by using a shortest path first strategy, constrained by the requested bandwidth. Finally, for space availability reasons, in all the presented results, we used some reasonable values for the GRASP parameters that have been empirically predetermined as those providing the best tradeoff between execution time and re-optimization performance (i.e., $q_{min} = 3$ and $q_{max} = 10$, $MaxIter = 30$, $EliteSize = 10$). In order to improve the results' consistency, each simulation run has been repeated 20 times and the average values associated to the performance metrics of interest have been calculated.

6.2. Results Analysis

The main performance metric that is shown in Fig. 2 is the amount of virtual machines-to-storage repositories access requests that have been satisfied (by establishing the needed LSPs) in presence of: 1) bare allocation (i.e., simple allocation without any re-optimization), 2) bare allocation + GRASP-based re-optimization performed at 400 connections and 3) bare allocation + re-optimizations at 400 and 800 connections. As it can be seen, the number of accepted requests for the bare allocation case grows almost linearly up to 500 connections, when its acceptance rate begins to slow down due to the network congestion on the best available paths between the most requested VM-storage node pairs. By performing a first GRASP-based re-allocation of the network resources in presence of 400 access requests, when the cloud is already experiencing a certain degree of congestion (the number of requested connections grows over the established one), a significant increment in the number of accepted connections is observed. However, after the re-allocation, the acceptance rate of the new incoming requests presents almost the same trend as the bare allocation (the two line maintain the same relative distance). This behavior is due to the fact that after the re-allocation performed by the GRASP re-optimization process, the new requests are served according to the same criterion used in bare allocation (i.e., one by one, as they arrive, without prior knowledge on future requests). Therefore, the infrastructure is prone to reach saturation again, even in presence of an higher acceptance rate. Finally, a second re-allocation is performed at 800 connections, which further increases the amount of established access connections, by rationalizing the resources utilization. The employment of a second GRASP-based re-optimization step has, clearly, to be weighted by evaluating the benefit/disruption trade-off.

The gain in terms of additional VM-storage accesses that is introduced by each re-optimization step is shown in the Fig. 3. As it can be seen, the major benefit is obtained with the first re-optimization, performed at 400 requests. Then, there is no increment in the number of extra-access requests that can be satisfied. On the contrary, the acceptance rate slowly decreases, due to

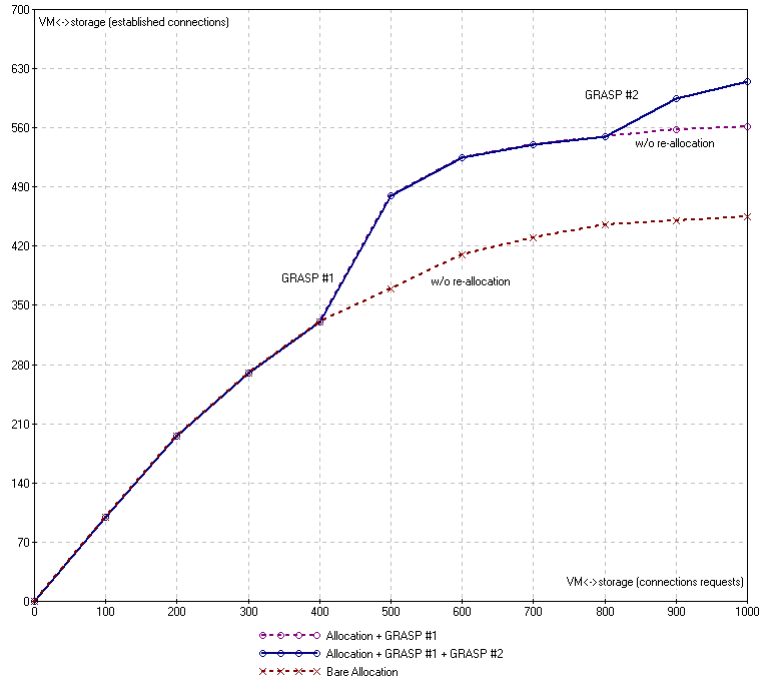

Figure 2: Established VM-storage connections according to 1) bare allocation (i.e., without re-optimization), 2) allocation + re-optimization performed at 400 connections and 3) allocation + re-optimizations at 400 and 800 connections.

the "blind" incremental allocation strategy following the first GRASP-driven re-organization. Performing a second re-optimization step at 800 connections introduced an additional gain by further improving the number of VM-storage access request at the expense of computational power and service disruption. However, the increase in the amount of extra accesses introduced by the second re-optimization is less than half of the first one, justifying it only in the cases in which the extra re-organization effort is offset by the gain associated to the extra connections.

In the Fig. 4, the amount of aggregated bandwidth available on the overall cloud infrastructure is reported for the aforementioned three cases. The pseudo-sinusoidal behavior that can be observed is common in traffic-related phenomena and is mainly due to the periodical reduction in link's spare capacities [21][25] as

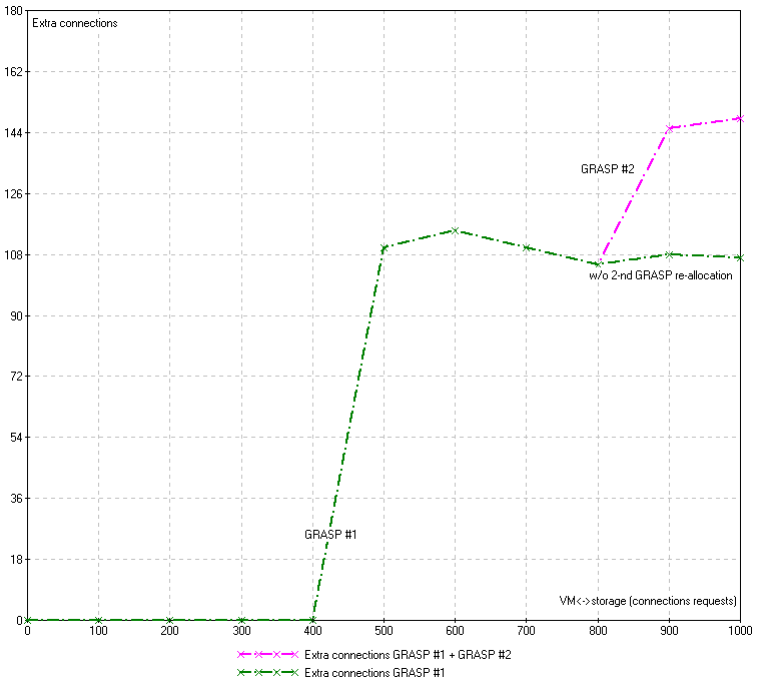

Figure 3: Extra VM-storage connections gained with re-optimization performed at 400 and 800 connections.

a result of the LSPs that are dynamically established and torn down in the network. The bare allocation stays in the same band along the whole set of access requests whilst, when re-optimization is performed, an increase in the available aggregated bandwidth can be observed, as a result of the better resource usage that the GRASP meta-heuristic introduces by deeply exploring the set of feasible solutions. The same marked increment in the available bandwidth can be observed after the second re-optimization step performed at 800 connections, fast converging at almost the same values of the first re-allocation.

The observation of the improvements, both in terms of additional access requests that can be satisfied (as shown in Fig. 3) and in terms of communication bandwidth recovered between the cloud sites (Fig. 4) gives us an immediate confirmation of the effectiveness of the proposed GRASP-based adaptive re-optimization strategy, mainly when coping with large geographically distributed

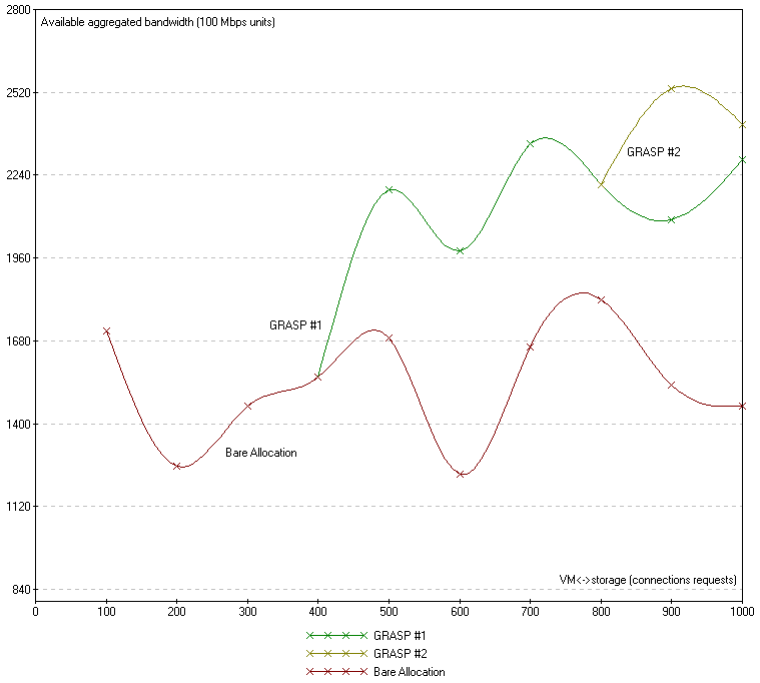

Figure 4: Available aggregated bandwidth *vs* VM-storage connections requests according to 1) bare allocation (i.e., without re-optimization), 2) allocation + re-optimization at 400 connections and 3) allocation + re-optimizations at 400 and 800 connections.

infrastructures and in presence of massive data access activities. This means that the basic ideas underlying such strategy are promising and deserve further exploitation by real world cloud solutions designers.

7. Conclusions

The recent encroachment of big data-related challenges in the computing scenario, and the possible solutions offered by federated clouds drove us to investigate the potential of re-optimization of the connections between VMs and data silos. A GRASP-based meta-heuristic, coupled with the analysis of possible rearrangement of the positioning of VMs, provided us a unified model and framework that can lead to a better utilization of the network resources and to improved scalability. An especially appealing characteristic of the proposed

GRASP-driven framework, making it the solution of choice for re-optimizing large-scale federated cloud infrastructures built on hundreds or thousands of participating sites scattered throughout the network, is its very simple and effective implementation. A small number of tunable parameters have to be properly fixed in order to have the whole framework operating effectively on virtually any kind of cloud infrastructure, so that all the implementation efforts can be focused on realizing efficient data representations and algorithmic solutions aiming at speeding GRASP iterations as possible. It should be also considered that, the greedy solution search activity can be easily parallelized on the participating cloud sites, by letting computing nodes, working as partially independent processing instances and using an independent pseudo-random sequence for selections within RCL, to simultaneously manage multiple GRASP iterations, by only sharing a single common variable that contains the best solution determined so far by all the involved processing instances. Such a straightforward parallelization strategy allows to substantially increase the number of iterations, and hence the overall solution quality, by reducing both the convergence time and the communication overhead on the transport network.

References

[1] T. A. Feo, M. G. Resende, Greedy randomized adaptive search procedures, Journal of Global Optimization 6 (2) (1995) 109–133.

[2] F. Glover, M. Laguna, R. Martí, Fundamentals of scatter search and path relinking, Control and Cybernetics 39 (3) (2000) 653–684.

[3] F. Glover, Scatter search and path relinking, New ideas in optimization (1999) 297–316.

[4] M. G. Resendel, C. C. Ribeiro, Grasp with path-relinking: Recent advances and applications, in: Metaheuristics: progress as real problem solvers, Springer, 2005, pp. 29–63.

[5] M. R. de Andrade, P. M. de Andrade, S. L. Martins, A. Plastino, Grasp with path-relinking for the maximum diversity problem, in: Experimental and Efficient Algorithms, Springer, 2005, pp. 558–569.

[6] B. Jennings, R. Stadler, Resource management in clouds: Survey and research challenges, Journal of Network and Systems Management (2014) 1–53.

[7] F. Pop, M. Potop-Butucaru (Eds.), Adaptive Resource Management and Scheduling for Cloud Computing, Vol. 8907 of Lecture Notes in Computer Science (LNCS) / Theoretical Computer Science and General Issues, Springer, 2015.

[8] F. Palmieri, Network-aware scheduling for real-time execution support in data-intensive optical grids, Future Generation Computer Systems 25 (7) (2009) 794–803.

[9] H. Casanova, F. Berman, G. Obertelli, R. Wolski, The AppLeS parameter sweep template: User-level middleware for the grid, in: ACM/IEEE 2000 Supercomputing Conference, IEEE, 2000, pp. 60–60.

[10] A. H. Alhusaini, V. K. Prasanna, C. S. Raghavendra, A unified resource scheduling framework for heterogeneous computing environments, in: Proceedings, Eighth Heterogeneous Computing Workshop (HCW '99), IEEE, 1999, pp. 156–165.

[11] K. Ranganathan, I. Foster, Simulation studies of computation and data scheduling algorithms for data grids, Journal of Grid Computing 1 (1) (2003) 53–62.

[12] F. Xhafa, A. Abraham, Computational models and heuristic methods for grid scheduling problems, Future Generation Computer Systems 26 (4) (2010) 608–621.

[13] J. Kołodziej, S. U. Khan, Data scheduling in data grids and data centers: a short taxonomy of problems and intelligent resolution techniques, in:

Transactions on Computational Collective Intelligence X, Springer, 2013, pp. 103–119.

[14] J. Basney, M. Livny, P. Mazzanti, Harnessing the capacity of computational grids for high energy physics, in: Conference on Computing in High Energy and Nuclear Physics, 2000.

[15] R. McClatchey, A. Anjum, H. Stockinger, A. Ali, I. Willers, M. Thomas, Data intensive and network aware (diana) grid scheduling, Journal of Grid Computing 5 (1) (2007) 43–64.

[16] X. Meng, V. Pappas, L. Zhang, Improving the scalability of data center networks with traffic-aware virtual machine placement, in: INFOCOM, 2010 Proceedings IEEE, IEEE, 2010, pp. 1–9.

[17] M. Alicherry, T. Lakshman, Network aware resource allocation in distributed clouds, in: INFOCOM, 2012 Proceedings IEEE, IEEE, 2012, pp. 963–971.

[18] S. Ghorbani, M. Caesar, Walk the line: consistent network updates with bandwidth guarantees, in: Proceedings of the first workshop on Hot topics in software defined networks, ACM, 2012, pp. 67–72.

[19] U. Fiore, F. Palmieri, A. Castiglione, A. De Santis, A cluster-based data-centric model for network-aware task scheduling in distributed systems, International Journal of Parallel Programming 42 (5) (2014) 755–775.

[20] M. G. Resende, C. C. Ribeiro, A grasp with path-relinking for private virtual circuit routing, Networks 41 (2) (2003) 104–114.

[21] F. Palmieri, U. Fiore, S. Ricciardi, A GRASP-based network re-optimization strategy for improving RWA in multi-constrained optical transport infrastructures, Computer Communications 33 (15) (2010) 1809–1822.

[22] T. A. Feo, M. G. Resende, A probabilistic heuristic for a computationally difficult set covering problem, Operations Research Letters 8 (2) (1989) 67–71.

[23] Y. Xin, M. A. Shayman, R. J. La, S. I. Marcus, Reconfiguration of survivable mpls/wdm networks., in: GLOBECOM, 2006.

[24] J. Taheri, A. Y. Zomaya, H. J. Siegel, Z. Tari, Pareto frontier for job execution and data transfer time in hybrid clouds, Future Generation Computer Systems 37 (0) (2014) 321 – 334.

[25] C. Casetti, R. Lo Cigno, M. Mellia, M. Munafo, Z. Zsoka, A realistic model to evaluate routing algorithms in the internet, in: Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE, Vol. 3, 2001, pp. 1882–1885 vol.3. doi:10.1109/GLOCOM.2001.965901.