Smart Partitioning of Geo-Distributed Resources to Improve Cloud Network Performance

Hooman Peiro Sajjad
School of Information and Communication Technology
KTH Royal Institute of Technology
Stockholm, Sweden
Email: shps@kth.se

Fatemeh Rahimian
SICS Swedish ICT
Stockholm, Sweden
Email: fatemeh@sics.se

Vladimir Vlassov
School of Information and Communication Technology
KTH Royal Institute of Technology
Stockholm, Sweden
Email: vladv@kth.se

***Abstract*—Cloud Computing systems with geo-distributed resources are becoming more popular for enabling a new family of applications, which are latency sensitive or bandwidth intensive, e.g., Internet of Things and online video gaming services. The approach is to host the cloud services at the network edges to reduce the latency and bandwidth consumption. However, the topology of the existing networks is not necessarily optimal for hosting Cloud services. Moreover, how the services are placed on the nodes, can affect the performance of the applications and the whole network. Therefore, we propose a novel algorithm to partition a distributed infrastructure into a set of computing clusters, each called a *Micro Data Center*. Our proposed algorithm is a decentralized community detection algorithm that does not require any global knowledge of the network topology. We compare our solution with a geolocation based clustering solution and demonstrate our preliminary results based on a real world network data set. We show that micro data centers increase the minimum available bandwidth in the network to up to 62%. Likewise, the average latency can be reduced to 50%.**

***Keywords*—geo-distributed cloud; community detection; cloud network performance; multiple data centers**

I. Introduction

Many new geo-distributed models for Cloud Computing have recently emerged, e.g., Fog Computing [1], Carrier Cloud [2], Edge Cloud [3] and Community Network Cloud [4]. Regardless of their initiators, the common approach in these models is to push the Cloud Computing paradigm (virtualization, on-demand resources and elastic services) to the network edges. The idea is to build a *distributed infrastructure* over a set of geo-distributed resources to enable hosting Cloud services closer to the end users. We refer to these resources as *nodes*. These resources can be colocated with the network devices such as routers, base stations, or the community network nodes. This enables a new category of latency sensitive, bandwidth intensive and mission critical applications to reside on the Cloud (e.g., Internet of Things, HD-quality video streaming and online gaming).

Distributed infrastructures are often built by exploiting the nodes and links of an existing network. Therefore, the topology of the network is not necessarily optimal for hosting Cloud services. This leads to a high variance in the network performance and traffic costs, due to different factors such as links heterogeneity, over-utilization of links and the number of hops between the communicating nodes. In such an infrastructure, a service or an application can be hosted on multiple nodes. Different assignments of services/applications to the nodes, affect the performance of the service/applications and the whole system.

Although the geo-distributed Cloud Computing models are becoming reality, there has been few studies on management and design of the distributed infrastructures. In this work, we provide a solution for partitioning a distributed infrastructure into a set of computing clusters. Each cluster is called a *micro data center*. Micro data centers are a set of geo-distributed resources, each having a Cloud manager and able to host Cloud applications. Later in Section V, we demonstrate the benefits of having multiple data centers and their impact on Cloud network performance.

To form multiple micro data centers, we propose a novel community detection algorithm, which does not require to know the entire network topology. Our community detection algorithm is based on multiple biased random walks and discovers clusters of nodes that have high connectivity. Consequently, we improve the quality of Cloud services and increase the the network performance (e.g., bandwidth and latency).

Our contributions in this work include:

- an approach to build distributed Clouds by smartly partitioning of geo-distributed resources into multiple micro data centers in order to increase the Cloud network performance;
- a novel decentralized community detection algorithm that does not need to know the whole network topology;
- evaluation of the proposed approach and the corresponding algorithm on a dataset based on a real-world network. The preliminary results show up to 62% increase in bandwidth and 50% decrease in the average latency in the network.

The rest of the paper is structured as follows. We explain the related work in Section II. In Section III, we overview a geo-distributed Cloud system and introduce our model. Section IV describes our proposed community detection algorithm. In Section V we present the evaluation results and conclude the work in Section VI.

II. Related Work

Community detection methods have been used a lot in the areas of social networks, biological networks, neuroscience and computer vision [5][6]. However, most of the algorithms only consider either undirected/directed or weighted/unweighted graphs. To our knowledge, there is no decentralized algorithm that can be applied to a weighted directed graph without having to know the whole topology of the network.

To our knowledge this is the first work that studies the idea of grouping distributed Cloud resources to build micro data centers in order to provide higher network guarantees. The closest work to ours is [7]. In that paper, the authors address a problem of having multiple virtual machines (VM) with different latencies and try to group VMs with lower latencies in order to host a distributed application on them. The algorithm iterates through all the nodes, sequentially and based on a threshold put the nodes in different groups. In our work, we group the distributed resources by accessing to their network infrastructure. We model the problem as a weighted graph so that it can be applied to either bandwidth/latency or any customized weights.

There has been other works using community detection to improve communication network. In [8], the authors propose a centralized community detection algorithm for clustering wireless sensor networks modeled in an unweighted undirected graph, in order to decrease the lost message rate and to increase the network coverage. Hui *et al.* [9] propose three community detection algorithms for reducing the cost of forwarding message in Pocket Switched Networks. They model the problem as an undirected unweighted graph. They propose distributed algorithms that find overlapping communities. This means that each node can appear in more than one community.

III. System Overview

In this section, we explain our conceptual architecture of a Cloud with geo-distributed data center and then we define the scope of this work and formulate the problem.

A. Conceptual Architecture

We define the conceptual architecture of a geo-distributed Cloud system in four layers (Figure 1):

Distributed infrastructure layer consists of a set of geographically distributed nodes. Each node has enough compute and storage resources to host one or several Virtual Machines or Linux Containers. These nodes are connected through a network infrastructure and can be even embedded in the network devices themselves. In other words, network edge devices or even end devices, such as routers, base stations, access points or community network nodes are enabled to host some of the Cloud services.

Micro data centers are Cloud data centers configured on top of the distributed infrastructure. A *micro data center recommender* partitions the distributed infrastructure into micro data centers by running our proposed algorithm. On each micro data center, a *Cloud infrastructure manager* (such as OpenStack [10]) can be set up. A Cloud infrastructure manager transforms a data center into an Infrastructure-as-a-Service.

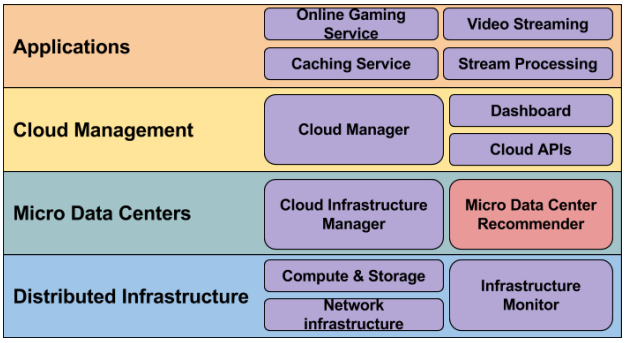


Fig. 1: Conceptual Architecture of a Geo-Distributed Cloud System

It has multiple components such as compute, storage and networking. Therefore, every micro data center has a infrastructure manager and can host Cloud services independently.

Cloud Management layer provides a unified access point to the Cloud services. This layer provides API services and user interfaces required for use and administration of the Cloud. In this conceptual architecture, we defined a component called *Cloud manager*, which is enrolled for all the inter micro data center tasks. Cloud manager consists of several services, such as monitoring of micro data centers, resource management, job orchestration and allocation, service migration and billing.

Applications are implemented by the service providers to be hosted in the Cloud. These applications, for example, can be some modern data intensive or latency sensitive applications, such as stream processing on the network edge, caching services that could reduce the traffic and latency by offering data closer to the users, video streaming or online gaming services.

As mentioned, the scope of this work is the algorithm to recommend micro data centers. This algorithm is used as a micro data center recommender shown in Figure 1. The algorithm is explained in the next section.

B. Model

We model a geo-distributed infrastructure as a set of heterogeneous nodes. We assume that all the nodes have, at least, the minimum required hardware to host Cloud service components. In addition, we assume that all the nodes and their connections have the same reliability rate in terms of failure. The nodes are connected with heterogeneous network connections, linking them with different inbound and outbound bandwidths and latencies. In this work, we do not assume a specific network topology and our target is a general model of a geo-distributed infrastructure. Having knowledge about the topology of the network can help to optimize the solution but it does not affect the general idea. We assume there is a routing path between every two nodes, that is the graph is strongly connected. Links are weighted and directed. Every pair of nodes that are connected, have the links in both directions, but the property of these links does not have to be the same. This is a valid assumption in real networks, as nodes usually have different upload and download bandwidth.

Every node v has a set of attributes including a unique identifier (UID), latitude (lat) and longitude (lon). Every edge

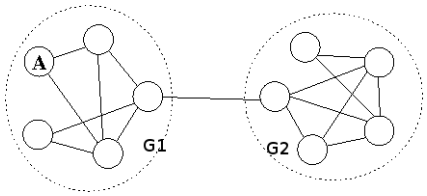


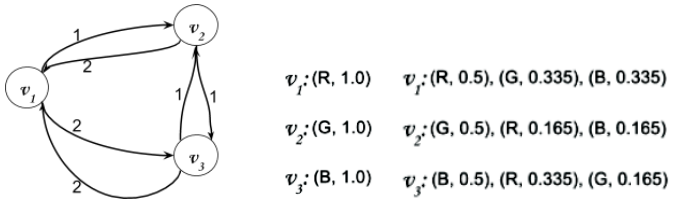
Fig. 2: An example of graph with two subgraphs. There is a higher probability that a random walk starting from node A ends up at a node in G_1 rather than a node in G_2.

e has attributes UID, source ID, destination ID, latency (l) and bandwidth (bw). Adding more attributes to the model does not change the general idea, and the same approach mentioned in this paper can be applied to the model. A weight (w) is assigned to every edge. The value of the weight is based on the edge attributes, such as bandwidth/latency or both. In this paper, we consider bandwidth as the edge weight, which is explained in detail in section V-B.

IV. Diffusion Based Community Detection Algorithm

To get the best out of a distributed infrastructure containing geo-distributed resources, we propose to partition the resources into a number of computing clusters, which we refer to as micro data centers. To better exploit these micro data centers we would like to have many high quality connections inside the clusters. This is in accordance with the well-known *modularity* metric in graph theory, which measures the fraction of the edges that fall within the given groups minus the expected such fraction if edges were distributed at random. The exact definition of modularity is provided in Section V-C. We propose a community detection method to increase this metric and find the natural community structures in a given network. Our solution is fully decentralized, does not require any knowledge about the global topology of the graph, and can be applied to directed/undirected weighted/unweighted graphs. It is an agglomerative community detection algorithm [11], in which the communities are built by merging smaller communities. Each node of the graph only needs to communicate with its adjacent nodes through message passing.

Our solution is based on random walk. When you perform a random walk on a graph, starting from a random source node, the walker is more likely to get trapped in the dense region where the source node belongs to. For example, in Figure 2, if we start from node A in subgraph G_1 and take a few steps, the probability that we end up at a node in G_1 is higher than a node in G_2. However, if we continue the random walk, at some point we will jump to G_2 and from then on we are more likely to stay in G_2 than to leave it. [12] proves that if we continue to walk indefinitely, the probability of reaching a node would only depend on its degree (the higher the degree, the higher the probability), and would not any longer reflect the topological structure of dense regions which are the communities. Hence, for our community detection algorithm to work, we will use a biased version of the random walk. We assume the walker can either stay in its current node or leaves it with a probability that is inversely proportional to its degree. If a node has a degree of one, then the random walker will leave the node, but the higher the degree, the smaller the probability of leaving the node.



(a) (b) (c)

Fig. 3: An example of the diffusion based algorithm. (a) an example graph. (b) after initialization phase. (c) after first round of color diffusion. R, G and B stand for Red, Green and Blue colors respectively.

We use multiple random walker, one starting at each node of the graph. In order to identify these walkers we give each one a unique color. Every node is initiated with a unit of a unique color and in each iteration it will send out a fraction of its color through its outgoing edges to its neighbouring nodes, proportional to the weight of edges. Note that in the unbiased random walk, the amount of color at each node would show the probability of the random walker ending at that node in that iteration. However in our case, the hubs, i.e., nodes with high degree, will be very sticky places that discourage colors to traverse from one region to another.

Finally, the color that has the highest quantity in the neighborhood of each node, is selected as its *dominant color*. Nodes that observe the same dominant color are the ones that are collocated in the same dense region of the graph. Therefore, they are considered to be in the same community. The role of the hub nodes are important in this phase too. They will become the most decisive nodes in their region, because they keep more color than other nodes. Therefore, the nodes that are connected to the same hub node, are more likely to observe the same dominant color and, thus, choose the same community.

We will now explain all the steps in more detail. Each node has a repository to hold the colors it receives. This could be implemented as an array or hashmap that holds a real value per color. We assume that the set of colors in a graph is denoted by C, where $|C| = |V|$, and $c_v \in C$ denotes the unique color that is initially given to node v. The algorithm proceeds in three phases: initialization, diffusion and community selection, which are as follows.

1) Initialization: In this phase, a unit of a unique color is assigned to each of the graph nodes. As an example, a graph with three nodes are demonstrated in Figure 3a. After the initialization phase, as shown in Figure 3b, v_1, v_2 and v_3, each are assigned a unit of red, green and blue, respectively.

2) Diffusion: This is the iterative part of the algorithm. Every diffusion step is equivalent to a random walk step. Each node of the graph sends a fraction of its existing colors to its adjacent nodes. The amount of colors that are sent out is inversely proportional to the degree of the node. This amount is then distributed between all the adjacent nodes, proportional to the weight of the edge connecting the two nodes. Assuming that $A(u)$ is the set of adjacent vertices of the vertex u, the amount of colors ($R_{u\to v}$) being sent from vertex u to the vertex v is as follows:

$$R_{u\to v} = \frac{R_u}{|A_u|} \times \frac{w_{uv}}{\sum_{\forall k \in A(u)} w_{uk}} \tag{1}$$

where R_u holds the available colors at node u. This step is shown in Algorithms 1, 2 for sending and receiving colors respectively. Figure 3c shows the color repositories of vertices $v1$, $v2$ and $v3$ after the first round of the color diffusion phase. As the color diffusion process continues, the color of each node will be spread across the graphs. Since the amount of color traversing the edges is a function of edge weights, the nodes with strong connections will keep the colors among each other for a longer period of time.

$WD_u = \sum_{\forall k \in A(u)} w_{uk}$; wighted degree
for all $v \in A(u)$ **do**
  $R_{u \to v} = \frac{R_u}{|A_u|} \times \frac{w_{uv}}{WD_u}$;
  sendColorsTo(v, $R_{u \to v}$);
end for

Algorithm 1: Diffusion phase, sending colors

$WD_u = \sum_{\forall k \in A(u)} w_{uk}$; wighted degree
$R'_u = R_u \times (1 - \frac{1}{|A_u|})$;
for all $v \in A(u)$ **do**
  $R_{v \to u}$ = recieveColorsFrom(v);
  $R'_u = R'_u + R_{v \to u}$;
end for
$R_u = R'_u$;

Algorithm 2: Diffusion phase, receiving colors

3) Community Selection: Upon termination (See Section IV-A), each node individually and only by receiving information from its adjacent nodes, can determine to which community it belongs to. In this phase (Algorithm 3), each node will ask for the list of colors from its adjacent nodes. It aggregates the colors of its adjacent nodes with its own colors and finds the *dominant* color, that is the color with the maximum amount. Then every node inform its neighbors about its selected dominant color, and verifies if there exist any other neighbor with that same dominant color. If there is, then the dominant color represents the community that the node belongs to. Otherwise, the node selects the second best dominant color and performs the verification again. Note that the verification step only makes the single node communities to merge with their neighbouring communities.

$S_u \leftarrow R_u$;
for all $R_v \in A_u$ **do**
  $S = S + R_v$;
end for
$dominant = argmax(S)$;

Algorithm 3: Community selection phase

A. Termination and Resolution

Due to the random walk properties, the more steps we take, the further colors are diffused in the graph and the more communities are merged together. We can indeed use this fact to our advantage, because partitioning the graph into communities could be performed with different granularity or resolution. Hence, we use the number of iterations as an input parameter to control the resolution of the found communities. The longer we run our algorithm, the more coarse grain communities will be found, that is fewer communities with bigger sizes.

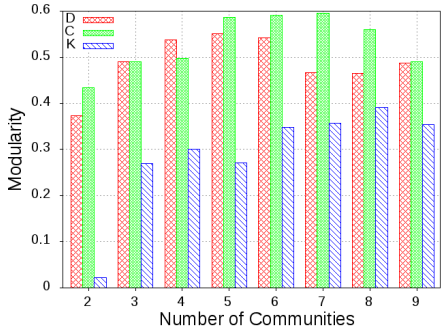


Fig. 4: Modularities of partitions.

V. Evaluation

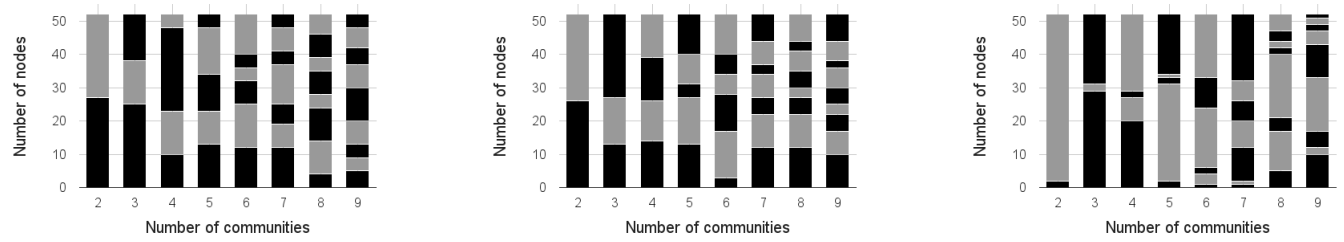
In this section, we compare Cloud network performance of a single geo-distributed infrastructure (the whole network infrastructure without applying any partitioning) against the micro data center solution. To form micro data centers, we use three different approaches: 1) a geolocation based solution using KMeans clustering (we use R software [13] configured to use Lloyd algorithm [14]); 2) a centralized community detection solution based on [15] and [16] (we use its implementation in Gephi software [17]); 3) our decentralized diffusion-based community detection. In the rest of this section, we discuss our evaluations. In all the figures, we present the results of single geo-distributed infrastructure with S, the centralized community detection solution with C, the decentralized community detection solution with D and KMeans clustering with K.

A. Modelling the Distributed Infrastructure

We use the data set of the QMP Sants-UPC network [18] in order to evaluate our work. QMP Sants-UPC (QMPSU) is a wireless multi-hop network. In this network, all the nodes provide the same routing functionality to relay the network traffic [19]. There is a monitoring system available, which collects nodes and links information on an hourly basis. We collect the 24 snapshots of the network (24 hours), starting from March 15, 2015. We calculate the average bandwidth and latency of the links in this period. Monitoring information of some links are missing in all the 24 snapshots. Either these links have a very bad quality or they're nodes are off. Therefore, we remove these links. We also exclude all the disconnected nodes. The final data set contains 52 nodes and 224 edges. We assume that all the nodes have enough resources to participate for hosting Cloud services.

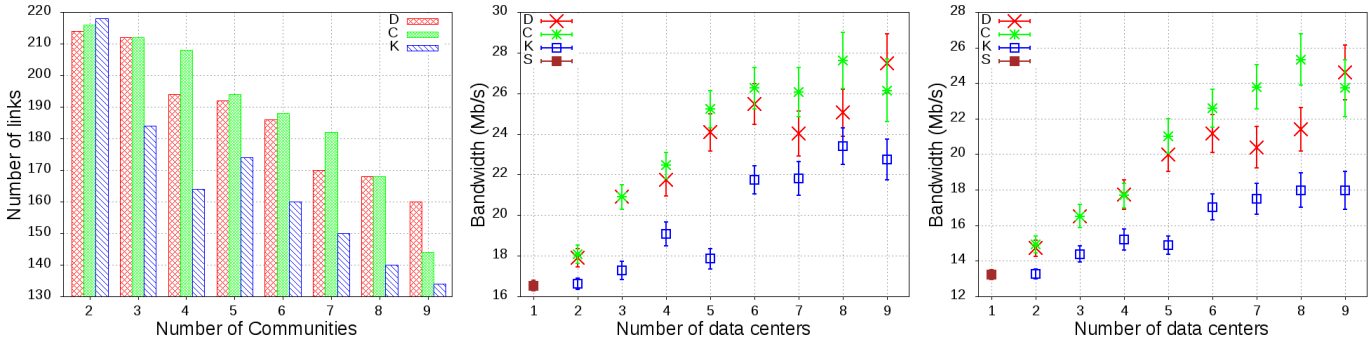
B. Parameter Configuration

Edge weight is an important parameter, which both the centralized and our decentralized solutions work based on. Both latency and bandwidth are important metrics in the quality of the micro data centers. We consider bandwidth of the links as the edge weight. There are different sources of latency in a wireless network [20], e.g., delays related to transmission path, link capacity and over subscribed nodes and links. Therefore, as we will see in the results, specifying the bandwidth as the edge weight, not only improves the connectivity inside the partitions regarding the available bandwidth, it also reduces the latency. In addition, the latency will be improved by decreasing the number of bottleneck nodes and the number of hops between nodes.



(a) Centralized (b) Decentralized (c) Geolocation based (KMeans)

Fig. 5: Distribution of the size of the communities for different number of partitions. In each column, every segment represents a distinct community.



(a) (b) (c)

Fig. 6: (a) Number of intra MDC links; Mean of minimum available bandwidths between each pair of nodes colocated in the same MDC for routing protocol as shortest path with (b) bandwidth as edge weight; (c) latency as edge weight; error bars present standard error.

In the centralized solution, we achieve different number of communities by modifying the resolution parameter. The resolution parameter takes a real number. The higher resolution values result less communities and the lower values result more communities. In our decentralized solution, different number of communities can be achieved by changing the number of iterations that the algorithm should run. We evaluate the results for partitions with 2 up to 9 number of communities. The community detection method is not for having a specific number of communities. However, we demonstrate the results in partitions with different number of communities. As we will see, our solution is better than geolocation based clustering in each partition regardless of the number of communities.

C. Modularity

We calculate the modularity for the different partitions with different number of communities. Modularity measures the density of links inside communities comparing to a random null model, where the edges are distributed randomly among the same nodes. More precisely, modularity for weighted graphs is calculated by considering the edge weights. The modularity (Q) for a weighted directed graph with n number of communities is as follows:

$$Q = \frac{1}{m}\sum_{c=1}^{n}\sum_{u,v\in P_c}\left(w_{uv} - \frac{d_u^{out}d_v^{in}}{m}\right)$$

w_{uv} is the edge weight from vertex u to vertex v and its value is zero if there is no connection between u and v. P is the set of communities and P_c is the set of vertices inside community c. $d_u^{out} = \sum_v w_{uv}$ and $d_v^{in} = \sum_u w_{uv}$ are in-degree and out-degree of a node considering its edge weights. m is also the sum of all the edge weights in the graph. The modularity gives a scalar value between -1 and 1, and higher values show better connectivity in the clusters, i.e., micro data centers.

The results are shown in Fig. 4. Geolocation clustering does not consider the connectivity between nodes and its partitions are solely based on the geolocation of the nodes. Therefore, the quality of the clusters with respect to the modularity are low. In a network, such as our data set, where the correlation between geographical distance of nodes and the quality of the links are low, the gap between the community detection based and the geolocation based solutions will be even more.

Our evaluations show another advantage of the community detection algorithms over geolocation based clustering, which is the size of the communities. KMeans clustering does not care about the size of the clusters. Very small size communities, such as single node communities, are discouraged in the community detection algorithms. As it can be seen in Figure 5a and 5b, the community sizes are more uniform than the clusters found by KMeans clustering (Figure 5c).

D. Micro Data Centers and Distributed Applications

We evaluate the quality of the Micro Data Centers (MDCs) recommended by the centralized, decentralized and geolocation based solutions. We evaluate the different solutions with respect to the distributed applications demands from the network infrastructure. In this regard, applications can fall into the groups of latency-sensitive, bandwidth-intensive or both.

The number of available links connecting each pair of node inside the MDCs shows the connectivity of MDCs. Figure 6a, demonstrates the number of available intra MDC links in the

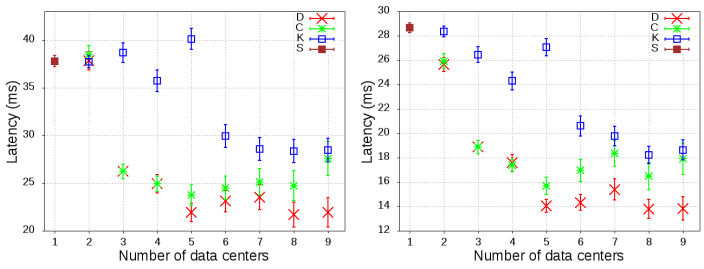


Fig. 7: Mean of latency between each pair of nodes colocated in the same MDC for routing protocol as shortest path with (a) bandwidth as edge weight; (b) latency as edge weight; error bars present standard error.

partitions produced by the different algorithms. As it can be seen, except for the partition with two communities, the rest of the results are in favor of the community detection based solutions. The reason that KMeans clusters have more intra data center links in the two MDCs partition, is due to its size of communities. As we depicted in Figure 5c, the MDCs produced by KMeans are very unbalanced.

Beside the number of available links in MDCs, the quality of the paths between the nodes are also important. Bandwidth intensive applications, require high bandwidth paths between nodes. The quality of each path is dominated by the link with minimum bandwidth. Therefore, we collected the minimum bandwidth link in every path. To do this, we assumed a routine protocol, which routes according to the shortest path between each pair of nodes. We calculated the shortest paths between every pair of nodes inside each MDC while considering the edge weights. We computed the shortest paths, separately, for bandwidth and latency as edge weights. Figure 6b and 6c show the minimum available bandwidth of the paths between the nodes colocated in the same MDC. Comparing to a single infrastructure, the minimum network bandwidth can be improved up to 48% in a configuration with 5 number of micro data centers and 62% with 8 number of micro data centers. In addition, in all the partitions, the community detection based solutions have higher minimum bandwidth between the nodes than the geolocation based solution. In addition, in most of the partitions, the result of our decentralized solution is so close to the centralized solution.

Distributed latency sensitive applications require a low latency between the nodes hosting their distributed components. To compare the MDCs produced by the different solutions with respect to such a requirement, we calculated the latencies for the shortest paths between each pair of nodes colocated in the same MDC. The results are depicted in Figure 7a and 7b. As it can be seen, the intra MDC paths between each pair of nodes have lower latencies in the community detection based algorithms. The mean of the latencies is improved by 50% in the partition with 5 communities. Due to the heuristic nature of the community detection solutions, we can see that our decentralized solution finds partitions having paths with lower latencies than the centralized solution.

In a network infrastructure, some of the nodes can become bottlenecks due to the topology of the network. In an infrastructure hosting multiple distributed applications generating network traffic between their different components, the bottleneck nodes will become overloaded and this will reduce the performance of the whole system. One good measurement for finding the bottleneck nodes is the betweenness centrality score [21]. In Figure 8a and 9a, we show the betweenness centrality of the nodes in different partitions, respectively considering bandwidth and latency as the edge weights. As it can be seen, betweenness centrality score in the partitions obtained by D and C are roughly the same and both are less than the partitions obtained by K. This is because our community detection algorithms, try to increase the intra MDC communications. Therefore, nodes that are hubs between two communities with high edge densities will be only joined to one of the communities. To demonstrate the effect partitioning on the links, we calculated the edge betweenness centrality score [22] for every edge and divided it by the bandwidth of each edge as a parameter to measure the capacity of the links when they are overloaded. Figure 8b and 9b shows that by partitioning based on the community detection algorithms, we can decrease the number of bottleneck edges and, therefore, have more available bandwidth for the applications colocated inside the MDCs.

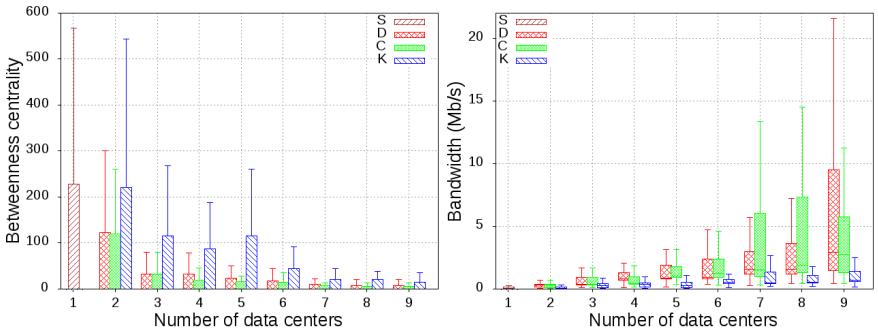
VI. Conclusion

We have shown that we can gain from partitioning of geo-distributed Cloud resources based on community detection methods. We call each groups of nodes a Micro Data Center. We showed that Micro Data Centers improve the quality of Cloud services and increase the predictability of the network performance. This is done by building Micro Data Centers with the nodes having higher connectivity. In addition, Micro Data Centers will reduce the number of potential bottleneck nodes and links. Our proposed decentralized community detection solution provides partitions with qualities competitive to the centralized one. The decentralized solution does not need to know about the whole network topology. We compared our decentralized algorithm with a geolocation based clustering method. We showed that by leveraging network characteristics we can do smarter partitioning rather than using mere geolocation based clustering. The micro data centers increase the minimum available bandwidth in the network to up to 62%. Likewise, the average latency can be reduced to 50%.

As a future work, we will continue our research in two dimensions. We will introduce a set of constraints to our proposed solution, such as minimum and maximum community size and available resources (compute and memory). We consider the model as a graph of heterogeneous nodes. Encouraged by our results, we also plan to apply the solution for partitioning Cloud VMs into the groups in order to have more predictable network guarantees.

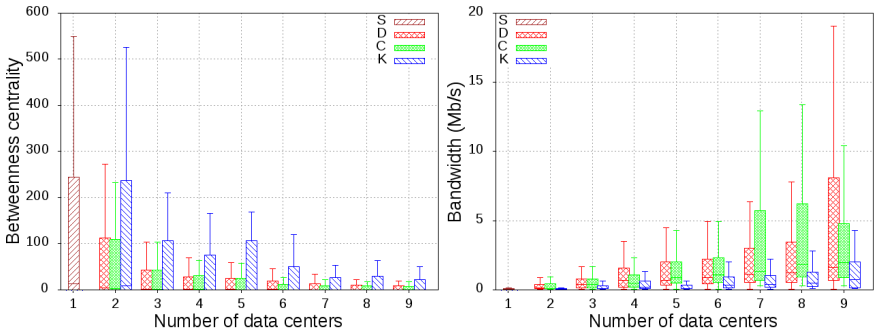
Acknowledgement

This work was supported by the FP7 project CLOMMUNITY funded by the European Commission under EU FP7 GA number 317879, the End-to-End Clouds project funded by the Swedish Foundations for Strategic Research under the contract RIT10-0043.



(a) (b)

Fig. 8: (a) Betweenness centrality of the nodes, considering bandwidth as the edge weight; (b) Minimum $\frac{bw}{EdgeBetweennessCentralityScore}$ in each intra MDC shortest path; boxes depict median, 25th, and 75th percentiles; Whiskers depict 5th and 95 percentiles



(a) (b)

Fig. 9: (a) Betweenness centrality of the nodes, considering latency as the edge weight; (b) Minimum $\frac{bw}{EdgeBetweennessCentralityScore}$ in each intra MDC shortest path; boxes depict median, 25th, and 75th percentiles; Whiskers depict 5th and 95 percentiles

REFERENCES

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12. New York, NY, USA: ACM, 2012, pp. 13–16.

[2] T. Taleb, "Toward carrier cloud: Potential, challenges, and solutions," *Wireless Communications, IEEE*, vol. 21, no. 3, pp. 80–91, June 2014.

[3] H. Chang, A. Hari, S. Mukherjee, and T. Lakshman, "Bringing the cloud to the edge," in *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, April 2014, pp. 346–351.

[4] J. Jimenez, R. Baig, P. Escrich, A. Khan, F. Freitag, L. Navarro, E. Pietrosemoli, M. Zennaro, A. Payberah, and V. Vlassov, "Supporting cloud deployment in the guifi.net community network," in *Global Information Infrastructure Symposium, 2013*, Oct 2013, pp. 1–3.

[5] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010.

[6] F. D. Malliaros and M. Vazirgiannis, "Clustering and community detection in directed networks: A survey," *Physics Reports*, vol. 533, no. 4, pp. 95–142, 2013.

[7] S. Malik, F. Huet, and D. Caromel, "Latency based group discovery algorithm for network aware cloud scheduling," *Future Generation Computer Systems*, vol. 31, pp. 28–39, 2014.

[8] L. Ferreira, A. Pinto, and L. Zhao, "Qk-means: A clustering technique based on community detection and k-means for deployment of cluster head nodes," in *Neural Networks (IJCNN), The 2012 International Joint Conference on*, June 2012, pp. 1–7.

[9] P. Hui, E. Yoneki, S. Y. Chan, and J. Crowcroft, "Distributed community detection in delay tolerant networks," in *Proceedings of 2nd ACM/IEEE international workshop on Mobility in the evolving internet architecture*. ACM, 2007, p. 7.

[10] (2015, Feb.) Openstack. [Online]. Available: https://www.openstack.org

[11] S. Fortunato, "Community detection in graphs," , vol. 486, pp. 75–174, Feb. 2010.

[12] J. D. Noh and H. Rieger, "Random walks on complex networks," *Physical review letters*, vol. 92, no. 11, p. 118701, 2004.

[13] B. D. Ripley, "The r project in statistical computing," *MSOR Connections*, vol. 1, no. 1, pp. 23–25, 2001.

[14] S. Lloyd, "Least squares quantization in pcm," *Information Theory, IEEE Transactions on*, vol. 28, no. 2, pp. 129–137, 1982.

[15] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 2008.

[16] R. Lambiotte, J.-C. Delvenne, and M. Barahona, "Laplacian dynamics and multiscale modular structure in networks," *arXiv preprint arXiv:0812.1770*, 2008.

[17] M. Bastian, S. Heymann, M. Jacomy *et al.*, "Gephi: an open source software for exploring and manipulating networks." *ICWSM*, vol. 8, pp. 361–362, 2009.

[18] (2015, May) Qmp sants-upc. [Online]. Available: http://dsg.ac.upc.edu/qmpsu/index.php

[19] L. Cerdà-Alabern, A. Neumann, and P. Escrich, "Experimental evaluation of a wireless community mesh network," in *Proceedings of the 16th ACM International Conference on Modeling, Analysis & Simulation of Wireless and Mobile Systems*, ser. MSWiM '13. New York, NY, USA: ACM, 2013, pp. 23–30.

[20] B. Briscoe, A. Brunstrom, A. Petlund, D. Hayes, D. Ros, J. Tsang, S. Gjessing, G. Fairhurst, C. Griwodz, and M. Welzl, "Reducing internet latency: a survey of techniques and their merits," *IEEE Communications Surveys & Tutorials*.

[21] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, pp. 35–41, 1977.

[22] L. Lu and M. Zhang, "Edge betweenness centrality," in *Encyclopedia of Systems Biology*. Springer, 2013, pp. 647–648.