Large distributed virtual infrastructure partitioning and provisioning across providers

Zhou, H.; Shi, Z.; Hu, Y.; Donkers, P.; Afanasyev, A.; Koulouzis, S.; Taal, A.; Ulisses, A.; Zhao, Z.

DOI
10.1109/SmartIoT.2019.00018

Publication date
2019

Document Version
Author accepted manuscript

Published in
2019 IEEE International Conference on Smart Internet of Things

Large distributed virtual infrastructure partitioning and provisioning across providers

Huan Zhou†, Zeshun Shi†, Yang Hu†, Pieter Donkers†,Andrey Afanasyev†, Spiros Koulouzis†,
Arie Taal†, Alexandre Ulisses‡ and Zhiming Zhao†*
†System and Networking Lab, University of Amsterdam, Amsterdam, the Netherlands
‡MOG technologies, Portugal
Email: {h.zhou, z.shi2, y.hu, a.taal}@uva.nl
pieterdonkers_3@hotmail.com, alexandre.ulisses@mog-technologies.com,
*z.zhao@uva.nl

***Abstract*—Cloud environments provide elastic capacity and flexible pay-as-you-go business model, and can significantly reduce the operational cost for resource-intensive applications like big data, deep learning and the Internet of Things (IoT). The virtual infrastructure, including networked virtual machines, for large scale applications, is often distributed, and involve resources not only from data centers, but also distributed Fog and Edge nodes. It is not always feasible to use resources all from a single provider. The provisioning of a complex virtual infrastructure requires dynamic partitioning of the topology and seamless configuration of the network, to meet the latency constraints of geolocation devices, to optimize the price and Service Level Agreements of the application. In many cases, the partition solution of a virtual infrastructure cannot be done in the design phase and has to consider the actual resource availability of providers at the provisioning stage. This paper presents a virtual infrastructure partitioning and provisioning solution with consideration of the QoS and resource constraints from the application.**

***Keywords*—Cloud computing; virtual infrastructure; graph partitioning; Fog and Edge**

I. Introduction

Cloud computing paradigm has been widely used in many fields for offering elastic resources, e.g., enhancing the computing capacity of robotics [1], storing and streaming Virtual Reality content from the Cloud [2] annotating named entities with crowdsourcing [3], and natural language processing for distributed machine learning [4]. The virtualisation technology in Cloud abstracts underlying system resources and operate them concurrently for multiple independent applications in an isolated way. Typical virtualisation techniques include hypervisors (e.g., KVM, XEN), containers (e.g., Docker, KATA, LXD) and unikernels (e.g., Unikraft).

The virtualized infrastructures in Cloud, e.g., customized virtual machines (VM), and application definable network topology and bandwidth connecting those VMs, are important services (namely Infrastructure-as-a-Service) for deploying and operating applications. In many cases, distributed applications, such as workflows of scientific computing tasks [5], [6], service chains for big data analytics [7] and early warning based on remote sensing [8], often require a sizeable virtual infrastructure. During the development and operation lifecycle of a Cloud application, the customization and provisioning of a suitable virtual infrastructure is an important step. Unlike traditional physical infrastructure, the virtual infrastructures provide developers with flexibility for planning not only capacity and type of the VM, but also their locations, i.e., the chosen clouds and data centers used [9]. Moreover, provided network resources can also be customized [10], [11].

In many cases, the virtual infrastructure of an application often has to be provisioned across different data centers or providers. There are several reasons for it. First, when the application involves geo-location bounded Edge or Fog nodes, which require Cloud servers in the virtual infrastructure to meet required latency, e.g., disaster warning based on large distributed sensor networks [8]. Second, when a data center meets its resource limit or performance failure, the application has to migrate part or the entire virtual resources to different data centers or providers, e.g., for the real-time online gaming, extra servers in different data centers are often employed to assure high quality of experiences [12]. Last but least, the resource budget energy efficiency and diverse Service Level Agreement (SLA) may also require careful selection and combination of the infrastructure resources from different providers or data centers, e.g., migrating tasks among regions based on time zones can optimize the costs for energy and resources [13].

The Cloud providers use the IaaS model for provisioning virtual infrastructures. For example, AWS CloudFormation[1], provided by the Amazon Web Service Cloud, is a useful tool to create and manage AWS resources, supporting automated provisioning and update. Nevertheless, it mainly supports for orchestrating web applications. It is a vendor lock-in solution, which can only be used on Amazon EC2 infrastructure. There are also tools to manage the infrastructures from different Clouds, avoiding vendor lock-in, such as Libcloud[2], jclouds[3] and fog[4]. However, these tools focus on the provisioning and manipulation on each individual VM instead of the entire infrastructure topology. In comparison, there are some environment-centric [14] tools to help developers orchestrate

[1]https://aws.amazon.com/es/
[2]http://libcloud.apache.org/
[3]https://jclouds.apache.org/
[4]http://fog.io/

their applications, which include Puppet[5], Chef[6], Ansible[7], JuJu[8] and Nimbus [15]. Those tools assume the existence of underlying virtual infrastructure which has already been provisioned. Some of them leverage the concept of managing "Infrastructure as Code" [16] during application lifecycle. However, the effective solution to handle the dynamic partitioning of large infrastructure has not been included in those tools. To tack this challenge, we developed a provisioning agent in the context of EU H2020 SWITCH[9] and ARTICONF project[10], to handle this problem.

The paper is structured as follows. First we will analyse the requirements for sizable virtual infrastructure provisioning from some use cases, and then introduce the software solution called Co-located and Orchestrated Network Fabric (CONF) we are developing in the EU project ARTICONF. After that, the implementation details and a case demonstration will be discussed.

II. Background

We shall first take a look at the requirements for including virtual infrastructure in the application development lifecycle, and then review the key technologies for partitioning virtual infrastructure when multiple data centers or providers are involved.

A. Problem context

During the Cloud application lifecycle, the virtual infrastructure is programmed, optimized and provisioned together with the application logic. The infrastructure provisioning is a crucial step for continuous application integration and deployment. We can see this from a number of application scenarios collected from the recent EU project SWITCH[11] and ARTICONF[12] authors are involved.

Example 1: Collaborative real-time business collaboration platform. Real-time communication plays an increasingly important role for many business applications, video conferences, cooperative working environment, and remote diagnosis. However, renting very high bandwidth or special connection links is not affordable for many business users in particular when the number of peers increases so the resources needed also increase considerably on the server side, where all the video and audio data mixing is done. An ideal real-time business collaboration platform should be provisioned and controlled based on the location of the users, the number of users, quality requirements for the expected interactions needed by the application.

[5]https://puppet.com/
[6]https://www.chef.io/
[7]https://www.ansible.com/
[8]https://jujucharms.com/
[9]European Horizon 2020 project, Software Workbench for Interactive, Time Critical and Highly self-adaptive cloud applications, http://www.switchproject.eu/.
[10]European Horizon 2020 project, smART socIal media eCOsytstem in a blockchaiN Federated environment, https://articonf.eu/.
[11]http://www.switchproject.eu
[12]http://www.articonf.eu

Example 2: The elastic disaster early warning system for natural disasters is an important challenge for many countries. An early warning system often collects data from real time sensors, processes the information using tools such as predictive simulation, and provides warning services or interactive facilities for the public to obtain more information. The implementation of this kind of system faces several challenges, as the system must: 1) collect and process the sensor data in nearly real time, 2) detect and respond to urgent events very rapidly (i.e. this is a time-critical scenario), 3) predict the potential increase of load on the warning system when public users (customers) increase, 4) operate reliably and robustly throughout its life time, and 5) be scalable when the deployment of sensors increases.

Example 3: Crowdsourcing based live event acquisition and broadcasting. In a live event, like a football match or a music concert, the broadcaster or production company has to deploy a large number of personnel and many items of equipment fully to cover it. Multiple cameras are placed around the event venue (stadium, forum) to cover all the different angles that the director considers relevant. Besides the cameras connected to an OB (outdoor broadcast), the media created by the event participants, or public are also an important input for the broadcasting company. Moreover, those users are in general distributed at different locations and produce different types of media, e.g., photos, video or audio. It is important to virtualize the basic components that an OB van may have (typically the video switchers) to the Cloud, and allow directors to interact with different video sources in the station via the Cloud environment instead of physically staying in the OB van. To realise this scenario, not only does the network for delivering different video streams need to be of high quality, but also the connectivity needs to be fully reconfigurable, based on the broadcast program scenarios. In the meantime, the video material should also be archived and streamed to users who want to watch it after the event.

From those use cases, we can clearly see the need for i) customizing a distributed virtual infrastructure, e.g., for connecting and processing data from distributed sensors in disaster early warning, or mobiles which contributing media

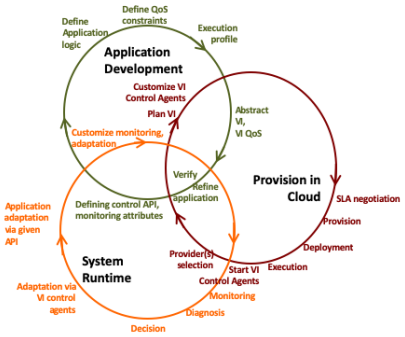


Fig. 1. Virtual infrastructure in the application lifecycle.

for crowdsourcing, ii) multiple data centers or provides, e.g., setting up a dynamic business platform for collaborative video processing among distributed teams, and iii) the flexible virtual infrastructure adaptation at runtime, e.g., scaling virtual infrastructure resources when the number of users increases, or migrating loads when network service quality drops.

B. Application development and operation lifecycle

During the application lifecycle, we can highlight infrastructure customization, provisioning and runtime adaptation as three key overlapped cycles of application development, provisioning and runtime during the application development and operation (DevOps) lifecycle. (see Fig. 1).

In the application development phase, the virtual infrastructure will be customized and planned based on the application requirements (components, dependencies among components, quality constraints for services or user experiences, locations, and resource budget).

A common description for the logic of a cloud application and its underlying virtual infrastructure enables the application developers, infrastructure capacity planner and application developers share a consistent view on the application system. It breaks the communication boundaries among different development teams, and promotes continuous development, integration and deployment in the entire development lifecycle. A formal description language is required to allow application developers to model not only functional requirements, e.g., application logic and functionality, but also non-functional aspects, e.g., quality of services and user experiences. Virtual infrastructures should include not only services from Cloud data centers, but also micro data centers, e.g., Fog nodes. Typical modelling languages include TOSCA, CloudDSL, ClouddNaas and GENTL [17].

In the virtual infrastructure provision phase, the customized virtual infrastructure will be provisioned in the Cloud environments, based on the resource budget, selection of the providers. In this phase, the application components being deployed on the virtual infrastructure. When an application requires infrastructure across Cloud, Edge and Fog, different provisioning strategies and interfaces will be considered.

Finally, the application will start its execution, and the virtual infrastructure will be monitored and diagnosed, and can be controlled at runtime.

III. Co-located and Orchestrated Network Fabric (CONF)

Co-located and Orchestrated Network Fabric (CONF) is a system being developed in the EU ARTICONF project to provides adaptive infrastructure provisioning for distributed Cloud applications over an orchestrated infrastructure. It seamlessly integrates with the Cloud edge infrastructure, able to intelligently provision services based on abstract application service requirements, operational conditions at the infrastructure level, and time-critical event triggering.

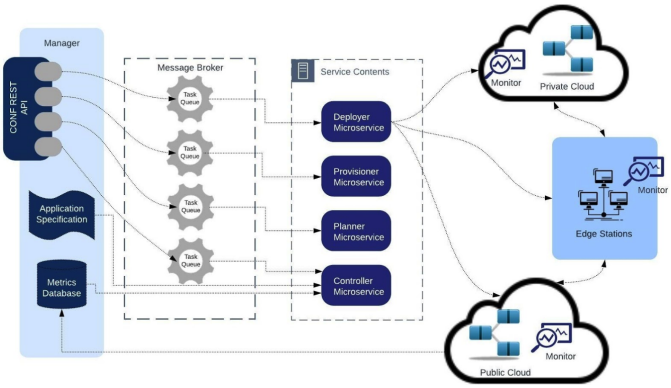


Fig. 2. The basic architecture of the CONF.

A. The basic architecture

As Fig. 2 depicts, the CONF software system provides a suite of micro-services that collectively perform the provisioning, monitoring and adaptation of customized virtual infrastructures for distributed applications. It seamlessly integrates with the cloud edge infrastructure, able to intelligently provision services based on abstract application service requirements, operational conditions at the infrastructure level, and time-critical event triggering. Henceforth, we describe the research work related to the concepts and methods of CONF components in the ARTICONF project, as well as the corresponding major opportunities and challenges.

The system architecture of CONF is shown in Fig 2. In general, CONF will adopt a microservice architecture and will be composed of the following components:

The Application-infrastructure Modelling language enables different roles of developers to communicate across the phases of application development, infrastructure planning, provisioning, deployment, and runtime monitoring. In the design phase, the developer can specify the initial QoS requirements for virtual infrastructure, and constraints between application components and the infrastructure, via a graphical editor [18]. The output can then be used for planning the infrastructure topology and the capacity, for provisioning the infrastructure in different providers, and for deploying application components over the infrastructure. At each phase, specific information will then be updated. For instance, during the provisioning phase, the public IP addresses will be updated in the description.

Manager. This component is implemented as a REST web service that allows CONF functions to be invoked by external clients. Each request is directed to the appropriate component by the manager, which is responsible for coordinating the individual components. Although the service of a single component can be called directly, it is common to perform all operations through the manager to simplify the interaction between subsystems.

Message broker. This component facilitates communication between the manager and the different components. The message brokering is an architectural pattern for message

validation, transformation, and routing, helping compose asynchronous, loosely coupled applications by providing transparent communication to independent components.

Metrics Database. This database is used by application and infrastructure agents to store predefined metrics whether they are system level metrics or application level metrics. Here we plan to use a time series database because it is capable of collecting and storing large amounts of data sorted by time. (such as Cassandra or InfluxDB which are both distributed storage systems for managing very large amounts of structured data).

Application Specifications. Each application will store and if necessary, modify its specifications (e.g., QoS, QoE, etc.). Here we need to define QoS/QoE attributes for social media application to check if they are satisfied for the given application and if there have some potential bottlenecks.

Controller. This is a dynamic controller that will swiftly take decisions for several aspects of social media applications or their infrastructure. These decisions shall be executed via either the provisioner or the deployer depending on the action that is needed.

Planner. This component encapsulates the infrastructure planning functionality. The planner will use several state-of-the-art scheduling and planning algorithms to produce efficient infrastructure topologies based on application requirements and constraints and will select optimal cost-effective virtual machines. Given the application description, it will try to generate the best infrastructure plan to host the application.

Provisioner. The infrastructure provisioner will automate the provisioning of infrastructure plans provided by the planner onto underlying infrastructure services. The provisioner can decompose the infrastructure description and provision it across multiple clouds, Edge or Fog infrastructure with transparent network configuration. Given the application and infrastructure description, the infrastructure provisioner will enable the SLA Negotiator to obtain the SLA contract required to ensure the QoS of the social media applications.

Deployer. The deployer installs application components onto provisioned infrastructure. The deployer is able to schedule based on network bottlenecks and maximize the satisfaction of deployment deadlines. It is also responsible for deploying supporting services, notably the execution and control agents needed to coordinate and run the application, and an instance system required to monitor and control the application and its underlying infrastructure autonomously.

B. Application-infrastructure description

The CONF uses TOSCA as the basic description language for application and virtual infrastructure. TOSCA describes the infrastructure/topology, the components, the relationships between them and the processes of composite cloud applications.

A description is divided into two parts, the topology description and the management plans. *Topology Description* defines the infrastructure of the application. It represent the infrastructure by using nodes and relations, which types are unspecified. For example, nodes can be hardware like machines but also software like applications and more. And connections can be connections, like database connections, or dependencies, like libraries, etc. *Management Plans* resemble a series of actions that should be taken for nodes. They can be triggered on certain moments in a node's lifetime, like configuring dependencies on creation of a virtual machine. The management plans are mostly comparable with configuration management tools like Ansible. It should be noted that these plans are imperative in contrary to the topology being declarative.

C. Privosioner: virtual infrastructure partitioning and provision

The infrastructure provisioner uses a graph partitioning component to dynamically map the virtual infrastructure onto the selected data centers (can be possibly from different providers), and then transparently configure the network among the subgraph of the infrastructure after each of them has been provisioned.

The CONF engine interprets the description, and

1) Check the resource availability of the providers, and issue the Service Level Agreement with individual provider;
2) Partition the virtual infrastructure, and create the subgraph of virtual infrastructure (called sub-virtual infrastructure);
3) Invoke the provisioning API of individual provider to provision corresponding sub-virtual infrastructure;
4) Configure the network of among different sub-virtual infrastructures, so that they still remain the connectivity as developer designed.

D. Infrastructure partitioning

Application's virtual infrastructure can be represented as a graph, where vertices are nodes and edges are link lines exchanging data. Represented graph need to be partitioned according to the developer requirements. This leads to well-known graph partitioning problem. Recent systematic reviews show that the Graph Partitioning Problem (GPP) is in focus of many researchers [26]. Due to the high demand in science and IT industry this field of mathematics is in active development where different graph partitioning algorithms exist.

Over the past two decades, a multiple graph partitioning tools were developed as open and closed source. Buluc et al., [26] summarises and discusses different graph partitioning software tools.

Table I shows a brief summary of the graph partitioning tools, in which the METIS is available as a package in Debian Stretch repository[13]. METIS supports multilevel recursive-bisection and multilevel k-way partitioning schemes. It can be used for both edge cutting and node clustering. METIS uses a classical Fiduccia-Mattheyses (fm) algorithm with optional One-sided node-based (sep1sided), and Two-sided node-based (sep2sided) refinement components. Those components

[13]https://packages.debian.org/stretch/metis

TABLE I
COMPARISON OF PROPOSED GRAPH PARTITIONING TOOLS AND DEFINED REQUIREMENTS

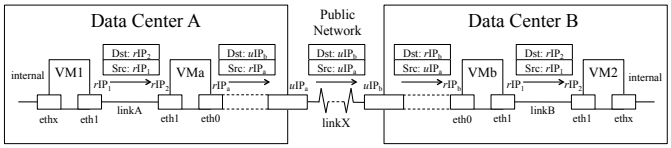
Tool	License	Latest stable release	Programming language	Cross-Platform	Python wrapper
Chaco [19]	GPL	v2.0, 1998	ANSI C	No (Unix-like)	No
KaHIP [20]	GPLv2	v2.00, 2017	C++	No (Unix-like)	Yes
KaHyPar-CA [21]	GPLv3, 2017	C++	Yes(Unix-like, Windows)	No	
METIS [22]	Apache 2.0	v5.1.0, 2013	ANSI C	Yes(Unix-like, Windows)	Yes (multiple)
Mondriaan [23]	LGPL	v4.2, 2017	C	No (Unix-like)	No
PaToH [24]	BSD	v3.2, 2011	C	No (Unix-like)	No
Zoltan [25]	LGPL	v3.8, 2016	C	No (Unix-like)	No

developed by a team behind METIS. Additionally, a Greedy algorithm can be used too.

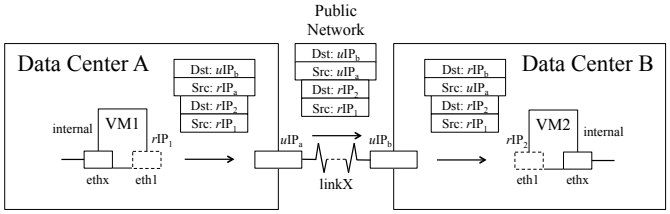
E. Network configuration

At different phases, the network information in the virtual infrastructure gets updated. In the design phase, the developer can only specify the private IP address and the logical topology among virtual machines. The partitioning of the virtual infrastructure will only manipulate the representation of the infrastructure graph, but not changing the network logical topology designed by the developer. After the provisioning of each sub-virtual infrastructure, the specific (public) IP addresses of each nodes will be assigned by the provider. Finally, the provisioner will re-connected those different sub-virtual infrastructures, and make them connected as originally designed by the developer.

Currently, two approaches are supported by the provisioner: 1) via the network address translation (NAT), and 2) tunnels. In Fig. 3, VM1 and VM2 are two VMs designed by the developer and located in two different data centers. Without CONF, they can only communicate with each other using public IP addresses, such as uIP_a and uIP_b. However, theses addresses are determined by the Cloud provider and therefore cannot be designed by the developer. CONF provides following two ways to enable VM1 and VM2 to communicate with following two private IP addresses, rIP_1 and rIP_2, which can be designed and customised by the application developer.



a. Network configuration via NAT proxy.



b. Network configuration via tunnels.

Fig. 3. The network configuration among sub-virtual infrastructures.

IV. CASE STUDIES AND PERFORMANCE CHARACTERISTICS

The software solution is developed in the context of EU SWITCH and ARTICONF project. It is used in a number of use cases. In this section, we will pick one case of crowd journalism to explain how CONF will work for the case.

The software solution is developed in the context of EU SWITCH and ARTICONF project. It is used in a number of use cases. In this section, we will pick one case of crowd journalism to explain how CONF will work for the case.

A. Use case

The concept of crowd journalism (also known as citizen journalism and crowdsourced journalism) has gained momentum in 2004 following the Indian Ocean tsunami, in which users in the location used their mobile phones to create the news story. Since then, this concept has become increasingly relevant in the discussion about the current state and future of journalism as a collaborative model. With the proliferation of news-oriented channels (traditional and online) and the increase in standard smartphone video quality, it is common for these channels to use citizen's content as they are usually the first to arrive to breaking news locations (e.g. car crashes, fires, earthquakes) and the first to capture relevant videos about the incident. The first broadcasting companies to have access to a given news are the ones that usually have the highest audience and, therefore, generate the highest revenues for the media publisher. At the same time, for example, in the case of an explosion in a chemical factory, it is much more important to inform the citizens and the community as whole as soon as possible, even by using lower quality videos than to wait for professionals to produce high-quality videos later on. As timing is crucial, it is necessary to provide tools that promote the crowd journalist, meaning the possibility to watch the different perspectives of a breaking news event through the eyes of the ones closest to it. At the same time, advances in post-processing tools and the spread of social media platforms have created an environment susceptible to the spread of rumours and fake news. The ease in which fake information spreads in electronic networks requires reputable news outlets that carefully verify third-party content before publishing it.

CONF will make the necessary provisions to transform the end-user smartphones into smart objects that capture content upon request in a wireless network facility and a scalable

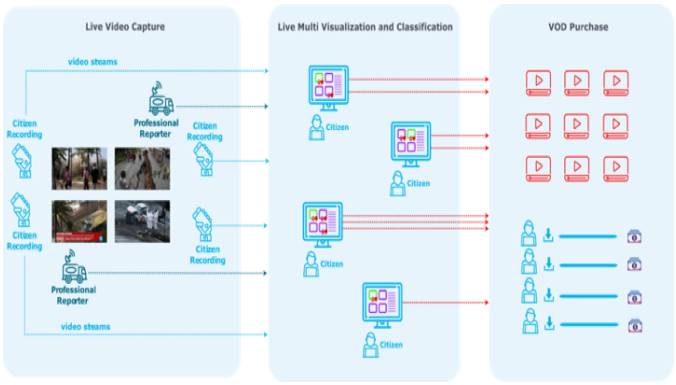


Fig. 4. Crowd journalism with news verification.

Cloud edge environment, dynamically enabling community crowd journalism and non-linear storytelling in breaking and developing news through collaborative co-creation of content. ARTICONF will make the necessary provisions to guarantee the QoS (e.g. bandwidth, latency) in the deployment and execution of the application, so that citizens at a breaking news location will have the necessary resources to transmit video streams through a nearby Cloud edge hosting resource to the media producer (e.g. journalist, director, producer). The producer selects a particular user, previews the associated active feeds, and live edits the received stream. To verify and guarantee the veracity of the content for a given piece of breaking news, the related social media feeds including crowd-generated content and images are matched.

B. Graph partitioning

The partitioning algorithms provided by METIS is wrapped and integrated in the provisioning agent. With different graph complexity, the partitioning performance is shown in Fig. 6. A graph partitioning elapsed time which takes less then 10 millisecond considered as a real-time partitioning.

- There are three different size (10, 100, 1000 nodes) regular graphs in degree of 3 were generated. Those graphs represent different scenarios.
- Partitioning done using a balanced mode with edge-cuts minimizing. It is suitable for the scenario that we want to balance the VM distribution in different data centers and minimise the communication cost at the same time.

All experiments were conducted on the physical machine with Ubuntu Server 16.04.3. Fig. 5 shows experiment of three random undirected graphs of different sizes. Table II represents characteristics of those graphs. Each of this graph represent a different situation.

TABLE II
GRAPHS CHARACTERISTICS USED DURING EXPERIMENTS

Id	Nodes	Edges	Maximum Degree
1	10	15	3
2	100	150	3
3	1000	1500	3

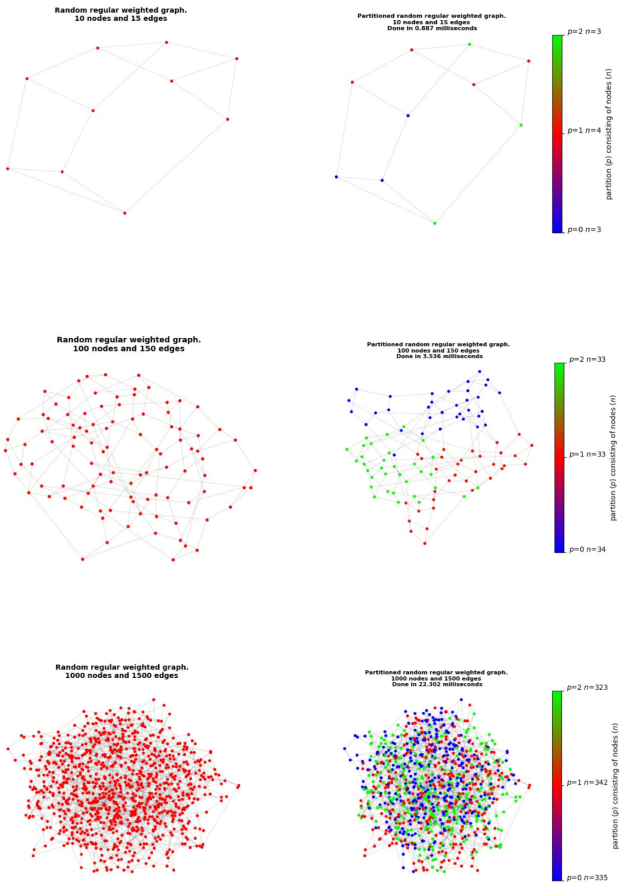


Fig. 5. Graph partitioning performance with three graph sizes: 10, 100 and 1000 nodes.

Further analysis on characterizing algorithms for specific virtual infrastructure topology and QoS constraints is still ongoing.

V. SUMMARY

A. Discussion and related work

Virtual infrastructure optimization is an important step in the lifecycle of Cloud application development and operation. For the application with critical constraints on time or system performance, virtual infrastructure and network aspects is crucial to be included in the optimization loop. In the current Cloud DevOps frameworks, Cloud resource optimization is often treated as part of the capacity planning, and mainly for the single provider. However, in this paper we argued that there are situations we will have to consider the distribution of provisioning across different data centers.

There are many cloud orchestration tools currently in the industry. Orchestration tools like Terraform[14] also allow multi

[14]https://www.terraform.io

cloud orchestration, in comparison to orchestration tools developed by cloud providers. All of these tools have no integrated planner and partitioner. It requires the user to specify exactly what they want with what provider. In contrast our tool only requires the user to specify what resources are required and our tool will choose where to place them. Their language also is not fully cloud agnostic, declaring an instance for AWS or Azure will require different code.

A graph partition problem (GPP) is a well-known non-deterministic polynomial-time (NP-hard) complexity problem. There are many graph partition existing algorithms and some of them are implemented in software tools [26]. This research is focusing on investigating literature proposed constraints provided from developer, application and data center perspective in relation to virtual infrastructure. During research a partitioning METIS as a software tool was selected. A prototype based on Python wrapper of tool was invented in order to measure graph partitioning elapsed time of each implemented algorithm. During experiment was used a 10, 100 and 1000 nodes graphs of 3 degree each with 15, 150 and 1500 edge retrospectively. A Fiduccia-Mattheyse algorithm with One-sided node-based refinement perform better than other algorithms for all types of defined graphs, but not significantly compare to two-sided node-based refinement. However, it was observed that this combination is more computational intensive compare to Greedy algorithm or even classical Fiduccia-Mattheyse.

B. Conclusions

From the discussion, we can conclude that large distributed virtual infrastructure require flexible provision solution for handling effective mapping between application requirements and the diverse providers. The provisioning agent discussed in the paper, enable the CONF handle the transparent network configuration for sub-virtual infrastructures when they are partitioned and provisioned across data centers.

C. Future work

As the next step, careful profiling of the graph partitioning algorithm, with characteristics of non-functional infrastructure constrains like fault tolerance, response time and energy efficiency, will be studied.

ACKNOWLEDGMENT

This work was supported by the European Unions Horizon 2020 research and innovation program under grant agreements No. 643963 (SWITCH project), No. 654182 (ENVRI^{PLUS} project), No. 824068 (ENVRI-FAIR project) and No. 825134 (ARTICONF project).

REFERENCES

[1] J. Wan, S. Tang, H. Yan, D. Li, S. Wang, and A. V. Vasilakos, "Cloud robotics: Current status and open issues," *IEEE Access*, vol. 4, pp. 2797–2807, 2016.

[2] P. Vecchio, F. Mele, L. T. De Paolis, I. Epicoco, M. Mancini, and G. Aloisio, "Cloud computing and augmented reality for cultural heritage," in *International Conference on Augmented and Virtual Reality*. Springer, 2015, pp. 51–60.

[3] T. Finin, W. Murnane, A. Karandikar, N. Keller, J. Martineau, and M. Dredze, "Annotating named entities in twitter data with crowdsourcing," in *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. Association for Computational Linguistics, 2010, pp. 80–88.

[4] L. Niu, X. Dai, J. Zhang, and J. Chen, "Topic2vec: learning distributed representations of topics," in *2015 International conference on asian language processing (IALP)*. IEEE, 2015, pp. 193–196.

[5] Z. Zhao, A. Belloum, C. De Laat, P. Adriaans, and B. Hertzberger, "Using jade agent framework to prototype an e-science workflow bus," in *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07)*. IEEE, 2007, pp. 655–660.

[6] Z. Zhao, A. Belloum, C. de Laat, P. Adriaans, and B. Hertzberger, "Distributed execution of aggregated multi domain workflows using an agent framework," in *2007 IEEE Congress on Services (Services 2007)*, July 2007, pp. 183–190.

[7] R. Mork, P. Martin, and Z. Zhao, "Contemporary challenges for data-intensive scientific workflow management systems," in *Proceedings of the 10th Workshop on Workflows in Support of Large-Scale Science*. ACM, 2015, p. 4.

[8] H. Zhou, A. Taal, S. Koulouzis, J. Wang, Y. Hu, G. Suciu, V. Poenaru, C. de Laat, and Z. Zhao, "Dynamic real-time infrastructure planning and deployment for disaster early warning systems," in *International Conference on Computational Science*. Springer, 2018, pp. 644–654.

[9] H. Ziafat and S. M. Babamir, "Optimal selection of vms for resource task scheduling in geographically distributed clouds using fuzzy c-mean and molp," *Software: Practice and Experience*, 2018.

[10] H. Zhou, J. Wang, Y. Hu, J. Su, P. Martin, C. De Laat, and Z. Zhao, "Fast resource co-provisioning for time critical applications based on networked infrastructures," in *Cloud Computing (CLOUD), IEEE International Conference on*, 2016, pp. 802–805.

[11] S. Koulouzis, A. S. Belloum, M. T. Bubak, Z. Zhao, M. Živković, and C. T. de Laat, "SDN-aware federation of distributed data," *Future Generation Computer Systems*, vol. 56, pp. 64–76, Mar. 2016. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167739X1500312X

[12] W. Cai, F. Chi, X. Wang, and V. C. Leung, "Toward multiplayer cooperative cloud gaming," *IEEE Cloud Computing*, vol. 5, no. 5, pp. 70–80, 2018.

[13] S. Taherizadeh, A. C. Jones, I. Taylor, Z. Zhao, and V. Stankovski, "Monitoring self-adaptive applications within edge computing frameworks: A state-of-the-art review," *Journal of Systems and Software*, vol. 136, pp. 19–38, 2018.

[14] J. Wettinger, U. Breitenbücher, O. Kopp, and F. Leymann, "Streamlining devops automation for cloud applications using tosca as standardized metamodel," *FGCS*, vol. 56, pp. 317–332, 2016.

[15] K. Keahey and T. Freeman, "Contextualization: Providing one-click virtual clusters," in *eScience'08. IEEE Fourth International Conference on*, 2008, pp. 301–308.

[16] M. Artac, T. Borovssak, E. Di Nitto, M. Guerriero, and D. A. Tamburri, "Devops: introducing infrastructure-as-code," in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 2017, pp. 497–498.

[17] A. Bergmayr, U. Breitenbücher, N. Ferry, A. Rossini, A. Solberg, M. Wimmer, G. Kappel, and F. Leymann, "A systematic review of cloud modeling languages," *ACM Comput. Surv.*, vol. 51, no. 1, pp. 22:1–22:38, Feb. 2018. [Online]. Available: http://doi.acm.org/10.1145/3150227

[18] P. Štefanič, M. Cigale, A. C. Jones, L. Knight, I. Taylor, C. Istrate, G. Suciu, A. Ulisses, V. Stankovski, S. Taherizadeh, G. F. Salado, S. Koulouzis, P. Martin, and Z. Zhao, "SWITCH workbench: A novel approach for the development and deployment of time-critical microservice-based cloud-native applications," *Future Generation Computer Systems*, p. S0167739X1831094X, Apr. 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167739X1831094X

[19] B. H. R. Leland and B. Hendrickson, "The chaco users guide: version 2.0," Technical report, Tech. Rep. SAND94-2692, Sandia National Labs, Albuquerque, NM, Tech. Rep., 1995.

[20] P. Sanders and C. Schulz, "Think locally, act globally: Highly balanced graph partitioning," in *International Symposium on Experimental Algorithms*. Springer, 2013, pp. 164–175.

[21] T. Heuer and S. Schlag, "Improving coarsening schemes for hypergraph partitioning by exploiting community structure," in *16th International*

Symposium on Experimental Algorithms (SEA 2017). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[22] D. LaSalle and G. Karypis, "Multi-threaded graph partitioning," in *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*. IEEE, 2013, pp. 225–236.

[23] D. M. Pelt and R. H. Bisseling, "A medium-grain method for fast 2d bipartitioning of sparse matrices," in *2014 IEEE 28th International Parallel and Distributed Processing Symposium*. IEEE, 2014, pp. 529–539.

[24] Ü. V. Çatalyürek, K. Kaya, and B. Uçar, "Integrated data placement and task assignment for scientific workflows in clouds," in *Proceedings of the fourth international workshop on Data-intensive distributed computing*. ACM, 2011, pp. 45–54.

[25] K. D. Devine, E. G. Boman, L. A. Riesen, U. V. Catalyurek, and C. Chevalier, "Getting started with zoltan: A short tutorial," in *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2009.

[26] A. Buluç, H. Meyerhenke, I. Safro, P. Sanders, and C. Schulz, "Recent advances in graph partitioning," in *Algorithm Engineering*. Springer, 2016, pp. 117–158.