Co-located and Orchestrated Network Fabric (CONF): An Automated Cloud Virtual Infrastructure for Social Network Applications

Zeshun Shi[1], Huan Zhou[1], Yang Hu[1], Spiros Koulouzis[1], Carlos Rubia[2], and Zhiming Zhao[1]✉[0000−0002−6717−9418]

[1] Informatics Institute, University of Amsterdam, Amsterdam, Netherlands
{z.shi2, h.zhou, y.hu, s.koulouzis, z.zhao}@uva.nl
[2] Agilia Center (AGI), Sevilla, Spain
carlos.rubia@agiliacenter.com

Abstract. Cloud environments can provide virtualized, elastic, controllable and high-quality on-demand infrastructure services for supporting complex distributed applications. However, existing IaaS (Infrastructure-as-a-Service) solutions mainly focus on the automated integration or deployment of generic applications; they lack flexible infrastructure planning and provisioning solutions and do not have rich support for the high service quality and trustworthiness required by social network applications. This paper introduces an automated cloud virtual infrastructure solution for social network applications, called Co-located and Orchestrated Network Fabric (CONF), which was conducted in a recently funded EU H2020 project ARTICONF. CONF aims to improve the existing infrastructure support in the DevOps lifecycle of social network applications to optimize QoS performance metrics as well as ensure fast recovery in the presence of faults or performance drops.

Keywords: Cloud · Virtual infrastructure · Social network applications.

1 Introduction

With the wide deployment of smart objects, mobile devices and increased connectivity, many applications nowadays operate on an ever-growing scale with high rates of churn and unpredicted peak demand. In this environment, social network applications allow for cooperative interactions among many participants, whether via mass public engagement (e.g., in crowdsourcing content creation) or as part of a persistent online community (e.g., car sharing services). Those applications have critical time constraints and strict trust requirements and therefore require a dynamic, adaptable infrastructure for hosting system components and supporting application users.

By providing elastic capacity and flexible pay-as-you-go business model, the virtualized infrastructure offered by cloud environments can significantly reduce the operational cost for resource-intensive applications like big data, deep learning and the Internet of Things (IoT). To effectively deploy social network applications in the cloud, the capacity of planning, provisioning, monitoring and

adaptation of the application's virtual infrastructure needs to be automated and seamlessly integrated into the whole process of the development and operation (DevOps) lifecycle. Moreover, such a virtual infrastructure solution has to effectively address the increased demand for Quality of Service (QoS) and Quality of Experience (QoE). However, existing cloud infrastructure services, e.g., provided by the DevOps environment of public providers, mainly focus on the automated integration or deployment of the generic applications; they have minimal support for the high performance and quality requirements of social network applications.

In this paper, we present an automated cloud virtual infrastructure solution for social network applications, called Co-located and Orchestrated Network Fabric (CONF). The work is conducted in a recently funded EU H2020 project ARTICONF. We will first analyze the requirements for the infrastructure automated solutions and then review the state of the art. After identifying the gaps, we will present the architecture design of the CONF system, and demonstrate the functional components of the system using a car sharing use case.

2 State of the art

In this section, we will first introduce the problem context of social network applications for infrastructure services and then review the related work as well as opportunities and challenges for future research.

2.1 Problem Context

Cloud and edge/fog platforms provide a virtualized infrastructure solution that can significantly optimize application usages and reduce operational costs of social network application[2]. However, in a large-scale heterogeneous and fragmented social network with smart objects spanning geographical boundaries, resource exploitation is challenging with respect to time-critical constraints, failovers, and QoS requirements. More specifically, current social network applications and PaaS (Platform as a Service) platforms lack flexible infrastructure planning and provisioning solutions. For example, if an application or platform needs to be deployed on the cloud and scale vertically or horizontally, current infrastructure services have the problem of vendor lock-in and cannot provide efficient resources to meet QoS requirements in an economical way. Moreover, existing social media cloud services lack of pervasive monitoring services to QoS metrics, as well as a self-adaptive mechanism to recover quickly from sudden failures. Monitoring QoS related metrics is necessary to improve efficiency for customizing, provisioning and controlling heterogeneous virtual infrastructures required by time-critical social network applications. An effective self-adaptive mechanism specifically to ensure fast recovery in the presence of faults or performance drops is also needed for quality-critical social network applications.

2.2 Related Work

In the DevOps lifecycle of cloud applications, infrastructure solutions are required in the whole process of infrastructure planning, provisioning, deployment, and monitoring [2]. Based on this idea, we identified the following four research topics to review the current related work.

Infrastructure Planning. Deploying the same application on different cloud infrastructures with the same specification may lead to entirely different results [11]. Therefore, developing an infrastructure planner that is bound to a specific budget without sacrificing performance is quite essential. A good cloud infrastructure planner should generate an optimal infrastructure strategy that not only meets the QoS requirements of the application but also achieves additional objectives such as minimizing monetary cost and power consumption, low latency, etc. For this reason, cloud infrastructure planning is often more challenging than scheduling application workflows onto fixed infrastructure [18]. Existing solutions about planning infrastructures for time-constrained applications typically have a global deadline. For example, IaaS Cloud Partial Critical Paths (IC-PCP) and Critical Path-based Iterative (CPI) are two typical algorithms that calculate the critical paths for VM services. To address the problem of multiple time constraints when responding to new events, Wang et al. [13] proposed a Multi-dEadline workflow Planning Algorithm (MEPA) to plan the most cost-effective virtual infrastructure for an application workflow.

Infrastructure Provisioning. Most IaaS clouds provide dedicated virtual infrastructure resources to applications with limited programmability and controllability, which enlarges the management gap between infrastructures and applications [19][17]. To bridge this gap, there have been substantial academic research as well as commercial tools. Zhou et al. [19] designed *CloudsStorm*, which is an application-driven DevOps framework that allows cloud users to program and control the cloud infrastructure directly. *SWITCH* is a software workbench for interactive time-critical and highly self-adaptive cloud applications. It also provides a programming model and toolkit to help programmers specify the QoS and QoE metrics of their distributed applications[19]. *CloudPerfect* [9] is another toolkit-based architecture for optimizing cloud infrastructure management, evaluating performance, and providing selection processes. The trustworthiness of the service quality is crucial to guarantee the run-time performance of the virtual infrastructure; smart contracts and blockchains have been used to enforce the SLA between providers and infrastructure users as well [20].

Software Deployment for Social Network Applications. Typically, social networks are centralized platforms with a single proprietary organization controlling the network. This poses critical issues of trust and governance over created and propagated content. To solve this problem, some scholars put the idea of deploying social network applications and platforms on distributed cloud

systems [6]. In this respect, Tan and Su [12] first proposed a new architecture for the media cloud and made some suggestions on how to build a media cloud in the future. Kim and Lee [7] presented their Social Media Cloud Computing (SMCC) model, which aims to provide flexible computing resources for processing large social media data and platforms. Wu et al. [14] proposed some algorithms for dynamic, optimal scaling of a social network application in geo-distributed clouds. Moreover, Chakravorty and Rong [3] put the idea of blockchain-based social network applications. Their platform, called *Ushare*, can support decentralization, anonymity, and traceability properties for future social network networks.

Infrastructure Monitoring and Self-Adaptation. Due to the dynamic nature of the cloud, continuous monitoring of QoS attributes is necessary to optimize cloud infrastructure operations or data transfer [8]. In this respect, Bleikertz et al. [1] established an automated security system called *Cloud Radar*, which could continuously monitor virtualized infrastructures for changes. Based on these changes, *Cloud Radar* updates a graph model representation of the infrastructure and maintains a dynamic information flow graph to determine isolation properties. Yuriyama and Kushida [16] proposed a new infrastructure model called *Sensor-Cloud Infrastructure* which can manage physical sensors on IT infrastructures. Yang et al. [15] proposed and validated an extensible SDN and NFV-enabled network traffic monitoring system. Mohammed et al. [10] developed a monitor and failure prediction model with Auto-Regressive Moving Average (ARMA) which focused on high-performance cloud data center infrastructure.

When some data centers are not accessible or some part of the computing resources crashed, the adaptability of infrastructure is therefore essential for these applications to recover quickly from sudden failures. Evans et al. [4] presented an approach to application reconfiguration scenarios of a distributed real-time social network application, called *Sentinel*. Zhou et al. [19] proposed a co-provisioning mechanism to improve the automatic recovery capability of the cloud infrastructure. More recently, Gill et al. [5] introduced their *CHOPPER* model, which offers self-configuration of applications and self-optimization for maximum resource utilization.

2.3 Challenges and Opportunities

In conclusion, during the DevOps lifecycle of social network cloud applications, application developers and managers need to plan and provide virtual infrastructures based on application requirements, deploy software platforms and application components on the virtual infrastructure, schedule and monitor the application execution, and adapt the infrastructure when performance declines. However, current infrastructure supporting mechanisms for social network applications are inefficient. Existing cloud infrastructure services, e.g., provided by the DevOps environment of public providers, mainly focus on generic applications, or the automated integration or deployment; they have minimal support

for the high performance and quality requirements of social network applications. Besides, they lack techniques to provision resources geographically closer to a specific event, which can lead to failovers such as service failures or performance drops, as the number of streaming users varies together with the processing needs corresponding to an event trigger. Therefore, there is an urgent need to design a complete set of infrastructure framework to provide resources support for the lifecycle of DevOps of social network applications.

3 Architecture and Prototype

The CONF component of ARTICONF project provides a suite of micro-services that collectively perform the planning, provisioning, monitoring, and adaptation of customized virtual infrastructures for federated time and quality critical social network 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. In this section, we will first introduce the system architecture and then discuss the development and implementation plan of the CONF system.

3.1 System Architecture

Based on the current technical requirements for social network applications, we designed our CONF framework, which is shown in Fig. 1. In general, CONF will adopt a microservice architecture and will be composed of the following components:

- *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 sub components.
- *Message broker*. This component facilitates communications between the manager and the different components. The message brokering is an architectural pattern for message validation, transformation, and routing. It can help compose asynchronous, loosely coupled applications by providing transparent communications to independent components.
- *Application Specifications*. Each social media application will store and modify its specifications in a social network. Here we need to define QoS/QoE attributes for social network applications to check if they are satisfied for the given scenarios and if there have some potential bottlenecks.
- *Metrics Database*. This component is used by application and infrastructure agents to store predefined metrics. Here we plan to use time series databases (e.g., Cassandra and InfluxDB) because they are capable of collecting large amounts of data and are easy to provide monitoring services.

- *Planner*. This component encapsulates the infrastructure planning functionality. It will use several state-of-the-art scheduling and planning algorithms to produce efficient infrastructure topologies based on application retirements and constraints and will select optimal cost-effective virtual machines.
- *Provisioner*. This component 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.
- *Deployer*. This component deploys application components onto provisioned cloud/edge infrastructures. The *deployer* is able to schedule based on network bottlenecks and maximize the satisfaction of deployment deadlines. It is also responsible for deploying blockchain applications and a monitor system required to monitor the application as well as its underlying infrastructure autonomously.
- *Controller*. This component will swiftly take measures to control and change the infrastructure solutions based on the QoS metrics of social network applications and their infrastructures. These decisions shall be executed via the whole process of planning, provisioning, and deployment when some actions are needed to ensure system self-adaptability.

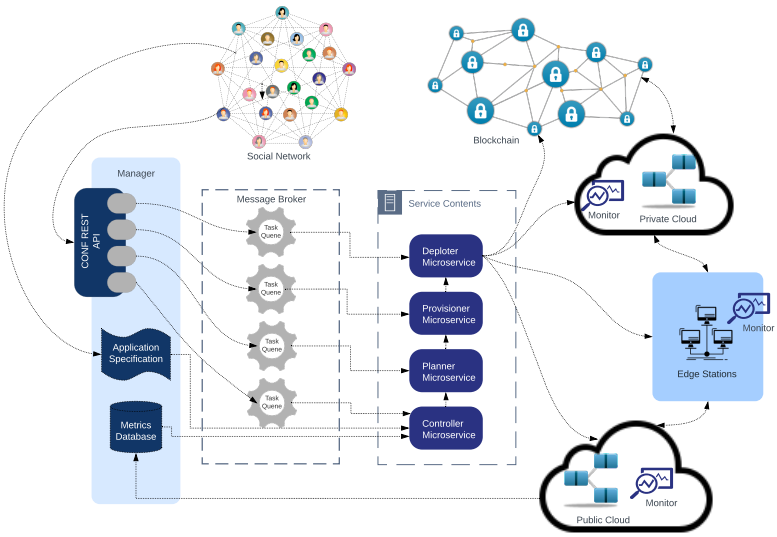

Fig. 1. The system architecture of CONF.

3.2 Development and Implementation

For CONF development and implementation, we will follow DevOps practices. Contentious testing and integration tools (e.g., Travis CI) will be leveraged for each development stage. We will define the external APIs with their documentations at first. For example, the *planner* API returns a concrete plan with resources (e.g., the number and size of VMs) based on the abstract plan for the applications. Similarly, given a concrete application plan, the *provisioner* API returns a document with provisioned resources and their specifications (IP addresses, etc.). Moreover, the *deployer* API returns a list of application components that need to be deployed based on the application specification documents produced by *provisioner*. And, given a deployed application, the *controller* API sets rules that will scale any part of the application (software components, VMs, etc.).

Next, we will define some tests for the APIs together with a suitable test environment. For each of the API endpoint, we will define tests to ensure the correct functionality of the API. Besides, it is necessary to define a simple representative application with well defined behavior to make sure the REST API performs as expected. And, a communication model should be defined between the components and the *manager*. This communication model should include action messages and status messages, in which the action messages are requests from the manager to the components while the status messages are responses about an action to the manager and will be available to users via the API.

Finally, we will implement each component of CONF one by one. More specifically, *Message Broker* will configure and define message queues that satisfy the communication model. *Manager* will implement the REST APIs and implement message queue dispatchers for each service according to the internal message model. *Planner* will implement parser to extract requirements from an abstract application specification and implement algorithms to achieve the optimal infrastructure plan. *Provisioner* will define abstract API that encapsulates the functionality of clouds (e.g. start, stop, delete and scale VMs) and implement drivers for specific cloud providers (ExoGENI, Amazon, etc.). *Deployer* will implement orchestration engine capable of configuring and installing any kind of social network applications onto VMs or other provisioned resources. *Controller* will implement error measuring and scaling decision making for changing the number and type of resources to achieve self-adaptation.

4 Case Study

Car sharing is a new collaborative model providing an alternative solution to private car ownership. This model allows customers to temporarily use a vehicle (on-demand) at a variable fee, charged depending on the distance traveled or time used. This sharing economy example, which can be business-to-consumer (B2C) or consumer-to-consumer (C2C), intends to satisfy transportation demand in a sustainable way by lowering emissions per city (due to fewer vehicles) and per vehicle (encouraging the use of electric or hybrid cars) and reducing traffic

and parking congestion. In this section, we will leverage car sharing as a use case to show how our CONF solution meet the requirements for social network applications.

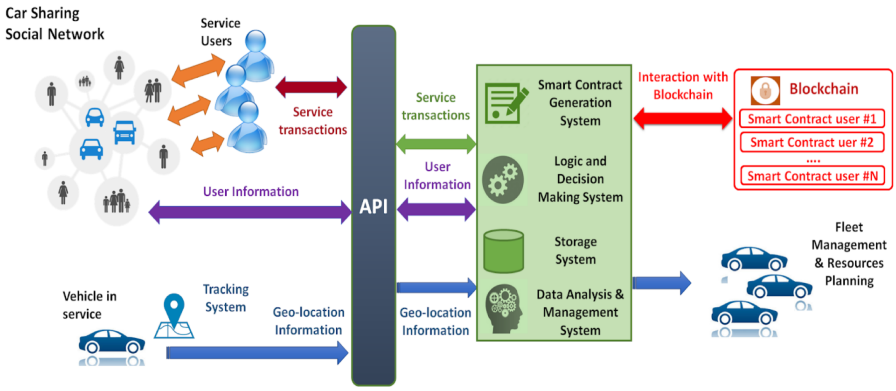

Fig. 2. The car sharing application scenario of AGI.

4.1 Application Scenario

The car sharing use case scenario of AGI (Agilia Center) is shown in Fig. 2. AGI is currently designing a new platform based on the blockchain and smart contracts to face this recent market and meet the appropriate requirements of the service. For this purpose, AGI deploys a social network platform for each city, used by customers to interact, plan (where and when a vehicle is available), hire a service, or share contents like photos and short videos. The platform allows the deployment of smart contracts in blockchain and verifies their compliance. A secondary system tracks the vehicle in service (by user) through a geo-location monitoring system that verifies the time and location in real-time, and the clauses of the contract. This business model suits car-renting companies too, as they use social networks to manage their vehicle fleet according to information provided by customers (e.g. for cleaning, repair, maintenance services). Moreover, the customer service obtains valuable information through data analysis to improve the offered service, forecast budgets, and procurement, design new services, or manage issues with unsatisfied customers. A problem is the unpredictability of the service that often leads to inefficient provisioning of resources. City events or bad weather conditions are examples that involve unexpected service demand peaks and high resource consumption, currently solved by over provisioning to maintain a time-critical response.

4.2 CONF Solutions and Benefits

The car sharing use case evaluates and takes advantage of the CONF impact in the following steps, which is shown in Fig. 3: The owners of vehicles and potential renting users together build a car sharing social network, in which the owners can post their rental advertisements while the users can interact with owners to find available vehicles. New users need to authenticate their identities to enter this social network. At the same time, the car owners need to authorise AGI to manage and track the real-time locations of vehicles. Next, the car sharing social network collects and submits the user information, the geographical location information of vehicles, and the application description data of the car rental behaviour (e.g. renting time, the departure place and the destination) to CONF. Based on these data, CONF calculates the optimal infrastructure support solution through planning algorithms and then triggers public/private cloud service providers to launch corresponding virtualized infrastructures. Besides, some edge resource stations that close to the geographical locations of the problem areas will also be triggered to ensure fast recovery in the presence of faults or performance drops. In this process, CONF has sufficient programmability and controllability to provide flexible infrastructure solutions, and it also avoids vendor lock-in problem for cloud providers in this process.

Then, CONF deploys car sharing blockchain applications on the cloud/edge virtual resource to provide real-time Blockchain-as-a-Service. The blockchain-based platform also supports penalties for contract breaches implemented in the smart contract. Moreover, when new transactions occur, CONF will continuously monitor the real-time status of cloud/edge computing resources and return the results to a metrics database. In this way, CONF can automatically control and adjust the computing resources when some unexpected happened. For example, when there are too much car rental transactions at a certain time and the existing computing resources cannot satisfy the car sharing blockchain applications, CONF will automatically start new cloud/edge infrastructures to meet the needs. Finally, CONF returns data from the cloud and blockchain to the social network so that all historical records of previous car rental transactions can provide a reference for new transactions. In conclusion, the benefits of CONF can be summarized as the following three:

Agile infrastructure planning and provisioning. CONF provides optimized planning and seamless provisioning of customized infrastructure across multiple cloud providers while ensuring a smooth horizontal and vertical scaling of the car sharing social network applications. More specifically, CONF optimize infrastructures solutions for car sharing social network applications in cloud/edge environments through the following actions: 1) Develop new algorithms for planning virtual infrastructure for a given social network application based on constraints on security, performance, locality and budget. 2) Develop an automated infrastructure provisioning engine able to deploy car sharing applications on a federated cloud infrastructure (over multiple sites if necessary). 3)

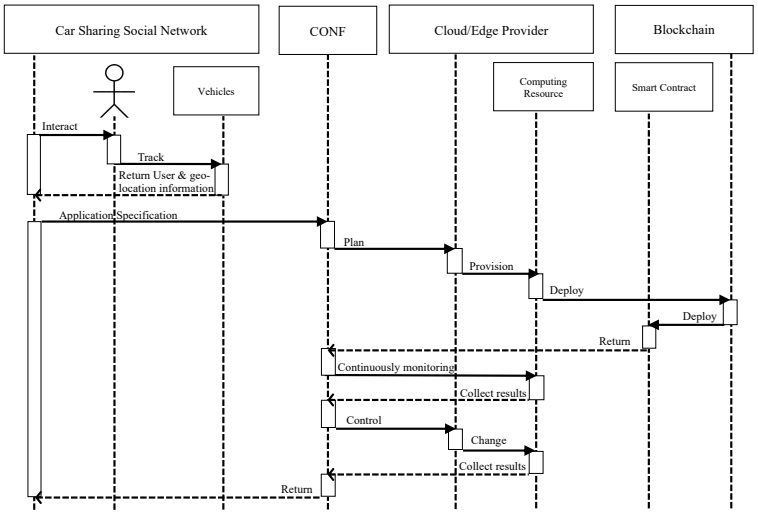

Fig. 3. The sequence diagram of the CONF solutions in car sharing use case.

Provide a secure API for the other services to invoke and query the provisioning engine.

Pervasive monitoring of applications and infrastructures. The monitoring services offered by CONF allowing systematic collection of information about the runtime status of the car sharing applications and of their underlying infrastructure and network. CONF also provide the analytic necessary to process the monitoring information in real-time and to identify indicators of reduced performance or faults through the following actions: 1) Provide an API for application integration and monitoring of key quality attributes. 2) Develop tools for monitoring the runtime state of the virtual infrastructure hosting an application. 3) Deploy a monitoring database service alongside applications with full integration.

Self-learning autonomous infrastructure adaptation. CONF will implement an infrastructure control model for time and trust-critical car sharing applications that captures the dependencies between infrastructure programmability and the applications performance. This autonomously adapts the infrastructure in response to the threats identified via the CONF's monitoring framework, or to changes in requirements triggered by the other services through the following actions: 1) Research of performance models for time and trust-critical federated social network applications. 2) Development of a control model for adapting the virtual infrastructure based on the interplay between the requirements and the metrics provided via monitoring. 3) Provision of an agent-based service for autonomous adaptation of the CONF-deployed infrastructure.

5 Conclusion & Future work

In conclusion, to improve the efficiency for customising, provisioning and controlling distributed cloud virtual infrastructures required by time-critical social network applications, CONF aims to deliver technologies providing a self-adaptive and self-monitored infrastructure over orchestrated networked services bringing two benefits. First, it will optimize QoS performance metrics (e.g. distribution time, latency) with proximity-based geo-profiling through seamless provisioning of a customised infrastructure across multiple geographical locations. Second, it will ensure fast recovery in the presence of faults or performance drops through rapid deployment and/or migration of application resources close to problem areas. As we mentioned before, the CONF solution is implemented in the ARTICONF project. Currently, some features of CONF are still under development. CONF will work with other components such as the Semantic Model with self-adaptive and Autonomous Relevant Technology (SMART), Trust and Integration Controller (TIC), and Tools for Analytics and Cognition (TAC) to provide integrated services of ARTICONF. For the future work, we will focus on continue to optimize the service components of CONF and design better algorithms to provide better infrastructure services for social network applications.

Acknowledgment. This work was supported by the European Union's Horizon 2020 research and innovation programme under grant agreements 825134 (ARTICONF project).

References

1. Bleikertz, S., Vogel, C., Groß, T.: Cloud radar: near real-time detection of security failures in dynamic virtualized infrastructures. In: Proceedings of the 30th annual computer security applications conference. pp. 26–35. ACM (2014)
2. Casale, G., Chesta, C., Deussen, P., Di Nitto, E., Gouvas, P., Koussouris, S., Stankovski, V., Symeonidis, A., Vlassiou, V., Zafeiropoulos, A., et al.: Current and future challenges of software engineering for services and applications. Procedia Computer Science **97**, 34–42 (2016)
3. Chakravorty, A., Rong, C.: Ushare: user controlled social media based on blockchain. In: Proceedings of the 11th international conference on ubiquitous information management and communication. p. 99. ACM (2017)
4. Evans, K., Jones, A., Preece, A., Quevedo, F., Rogers, D., Spasić, I., Taylor, I., Stankovski, V., Taherizadeh, S., Trnkoczy, J., et al.: Dynamically reconfigurable workflows for time-critical applications. In: Proceedings of the 10th Workshop on Workflows in Support of Large-Scale Science. p. 7. ACM (2015)
5. Gill, S.S., Chana, I., Singh, M., Buyya, R.: Chopper: an intelligent qos-aware autonomic resource management approach for cloud computing. Cluster Computing **21**(2), 1203–1241 (2018)
6. Hu, Y., Zhou, H., de Laat, C., Zhao, Z.: Ecsched: Efficient container scheduling on heterogeneous clusters. In: European Conference on Parallel Processing, pp. 365–377 (2018)

7. Kim, M., Lee, H.: Smcc: Social media cloud computing model for developing sns based on social media. In: International Conference on Hybrid Information Technology. pp. 259–266. Springer (2011)
8. Koulouzis, S., Belloum, A.S., Bubak, M.T., Zhao, Z., Živković, M., de Laat, C.T.: Sdn-aware federation of distributed data. Future Generation Computer Systems **56**, 64–76 (2016)
9. Kousiouris, G., Aisopos, F., Psychas, A., Varvarigou, T., Domaschka, J., Baur, D., Griesinger, F., Nikolov, V., Lyberopoulos, G., Theodoropoulou, E., et al.: A toolkit based architecture for optimizing cloud management, performance evaluation and provider selection processes. In: 2017 International Conference on High Performance Computing & Simulation (HPCS). pp. 224–232. IEEE (2017)
10. Mohammed, B., Modu, B., Maiyama, K.M., Ugail, H., Awan, I., Kiran, M.: Failure analysis modelling in an infrastructure as a service (iaas) environment. Electronic Notes in Theoretical Computer Science **340**, 41–54 (2018)
11. Sun, Y., White, J., Eade, S.: A model-based system to automate cloud resource allocation and optimization. In: International Conference on Model Driven Engineering Languages and Systems. pp. 18–34. Springer (2014)
12. Tan, M., Su, X.: Media cloud: When media revolution meets rise of cloud computing. In: Proceedings of 2011 IEEE 6th International Symposium on Service Oriented System (SOSE). pp. 251–261. IEEE (2011)
13. Wang, J., Taal, A., Martin, P., Hu, Y., Zhou, H., Pang, J., de Laat, C., Zhao, Z.: Planning virtual infrastructures for time critical applications with multiple deadline constraints. Future Generation Computer Systems **75**, 365–375 (2017)
14. Wu, Y., Wu, C., Li, B., Zhang, L., Li, Z., Lau, F.: Scaling social media applications into geo-distributed clouds. IEEE/ACM Transactions on Networking (TON) **23**(3), 689–702 (2015)
15. Yang, C.T., Chen, S.T., Liu, J.C., Yang, Y.Y., Mitra, K., Ranjan, R.: Implementation of a real-time network traffic monitoring service with network functions virtualization. Future Generation Computer Systems **93**, 687–701 (2019)
16. Yuriyama, M., Kushida, T.: Sensor-cloud infrastructure-physical sensor management with virtualized sensors on cloud computing. NBiS **10**, 1–8 (2010)
17. Zhao, Z., van Albada, D., Sloot, P.: Agent-Based Flow Control for HLA Components. SIMULATION **81**(7), 487–501 (Jul 2005)
18. Zhao, Z., Belloum, A., De Laat, C., Adriaans, P., Hertzberger, B.: Using Jade agent framework to prototype an e-Science workflow bus. In: Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid '07). pp. 655–660. IEEE, Rio de Janeiro, Brazil (May 2007)
19. Zhou, H., Hu, Y., Ouyang, X., Su, J., Koulouzis, S., Laat, C., Zhao, Z.: CloudsStorm: A framework for seamlessly programming and controlling virtual infrastructure functions during the DevOps lifecycle of cloud applications. Software: Practice and Experience **49**(10), 1421–1447 (Oct 2019)
20. Zhou, H., Ouyang, X., Su, J., Laat, C., Zhao, Z.: Enforcing trustworthy cloud SLA with witnesses: A game theorybased model using smart contracts. Concurrency and Computation: Practice and Experience (Sep 2019)