Dynamic real-time infrastructure planning and deployment for disaster early warning systems

Huan Zhou[1], Arie Taal[1], Spiros Koulouzis[1], Junchao Wang[1], Yang Hu[1],

George Suciu Jr. [2], Vlad Poenaru[2], Cees de Laat[1], Zhiming Zhao[1][0000-0002-6717-9418]

[1] University of Amsterdam, 1098XH, Amsterdam, the Netherlands
{h.zhou|a.taal|j.wang2|y.hu|delaat|z.zhao}@uva.nl
[2] BEIA consultant, Romania
{george.suciu|vlad.poenaru}@beia.ro

Abstract. An effective nature disaster early warning system often relies on widely deployed sensors, simulation based predicting components, and a decision making system. In many cases, the simulation components require advanced infrastructures such as Cloud for performing the computing tasks. However, effectively customizing the virtualized infrastructure from Cloud based time critical constraints and locations of the sensors, and scaling it based on dynamic loads of the computation at runtime is still difficult. The suitability of a Dynamic Real-time Infrastructure Planner (DRIP) that handles the provisioning within cloud environments of the virtual infrastructure for time-critical applications is demonstrated with respect to disaster early warning systems. The DRIP system is part of the SWITCH project (Software Workbench for Interactive, Time Critical and Highly self-adaptive Cloud applications).

Keywords: Cloud, disaster early warning, time critical systems.

1 Introduction

An elastic early warning system enables people and authorities to save lives and property in case of disasters. In case of floods, a warning issued with enough time before the event will allow for reservoir operators to gradually reduce water levels, people to reinforce their homes, hospitals be prepared to receive more patients, authorities to prepare and provide help [3-5]. An early warning system often collects data from sensors, processes the information using tools such as predictive simulation, and provides warning services or interactive facilities for the public to obtain more information [1].

Depending on factors like the spatial and temporal scale of a specific environmental degradation, early warning systems are often highly distributed [8-10]. An ideal disaster early warning system needs to minimize prevention costs and increase prevention efficiency in case of flood and other possible disaster events. But there is a trade-off between timeliness, warning reliability, the cost of a false alert, and damage avoided as a

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_51

function of lead time, which must be modelled to determine the cost efficiency of the outcome [6,7].

In this paper we focus on supporting disaster early warning systems using Cloud, and specifically highlight the challenges of customizing, provisioning, and runtime managing virtual infrastructure based on the time critical constraints from early warning systems. The research is performed in the context of EU H2020 SWITCH project. An automated infrastructure planning and provisioning tool called Dynamic Real-time infrastructure planner (DRIP) will be presented. In the rest of the paper, we will first discuss the requirement challenges of the an early warning system, and then present the basic architecture of DRIP. After that a use case is used to demonstrate the current implementation.

2 Early warning systems and challenges

2.1 A use case of early warning system

The essential structure of any early warning systems depends on the objectives of the system to provide important, timely information on specific phenomena to end-users and decision-makers, thereby enabling effective response [6].

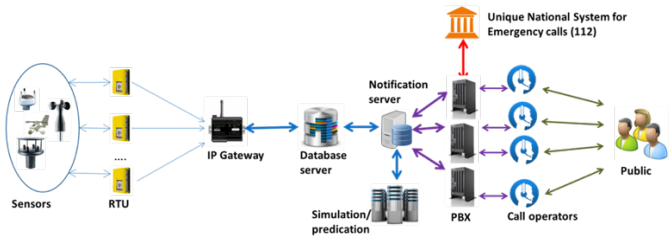

Fig. 1. Functional diagram for elastic early warning system

Fig. 1 presents a typical use case scenario. Sensors in the field transmit information to the IP Gateway. This gateway transmits the data collected to the database server. The notification server (Interactive Voice Response + Contact Center) periodically checks the data from the database, and, if they exceed certain values set, then on different communications channels, notifications are sent to an available operator that is scheduled to process the event. The operator checks statistics data received from sensors and transmits the decision whether or not to alert Unique National System for Emergency Calls (112).

2.2 Requirements and problems

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);

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_51

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;
5. be scalable when the deployment of sensors increases.

The development of such applications is usually difficult and costly, because of the high requirements for the runtime environment, and in particular the sophisticated optimisation mechanisms needed for developing and integrating the system components. In the meantime, a Cloud environment provides virtualised, elastic, controllable and quality on demand services for supporting systems like time critical applications. However, the engineering method and software tools for developing, deploying and executing classical time critical applications have not yet included the programmability and controllability provided by the Clouds; and the time critical applications cannot yet get the full potential benefits which Cloud technologies could provide.

It is still an open question whether disaster early warning systems, like the one outlined above, are suited to run in one or more private or public cloud environments. To deploy and control such time-critical systems asks for a workbench of dedicated tools each having its well defined task.

2.3 Time critical challenges

Laplante and Ovaska [11] define a real-time system as "a computer system that must satisfy bounded response-time constraints or risk severe consequences". The actual nature of individual response-time constraints varies. For example, often time constraints imposed on the acquisition, processing and publishing of real-time observations, not least in scenarios such as weather prediction or disaster early warning [12]. The ability to handle such scenarios is predicated on the time needed for customisation of the runtime environment and the scheduling of workflows [13, 23], while the steering of applications during complex experiments is also temporally bounded [14]. Time constraints are imposed on the scheduling and execution of tasks that require high performance or high throughput computing (HPC/HTC), on the customisation, reservation and provisioning of suitable infrastructure, on the monitoring of runtime application and infrastructure behaviour, and on runtime controls.

Disaster early systems we are concerned with often have multiple overlapping response-time constraints on different parts of the application workflow. Note that our concern of “time critical” constraints is not only with executing applications as quickly as possible, but also with ensuring stable performance within strict boundaries in the most cost-effective manner feasible (where ‘cost’, particularly in private Clouds, might be measured in terms of metrics other than money, such as energy consumption).

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_51

3 Dynamic real-time infrastructure planner

The Dynamic Real-time Infrastructure Planner (DRIP) is a system developed in the SWITCH project for the planning, validation and provisioning of the virtual infrastructure enlisted to support an application with time critical constraints. It is part of the SWITCH workbench, which includes two other subsystems i) GUI for composing, executing and managing applications, namely The SWITCH Interactive Development Environment (SIDE), and ii) a runtime monitoring and adaptation sub system, namely The Autonomous System Adaptation Platform (ASAP) [22].

3.1 Architecture and components

The key features are modelled as a number of micro services, which are coupled via message brokers of DRIP manager. It provide a unified interface for clients such as SIDE or ASAP, as shown in Fig. 2.

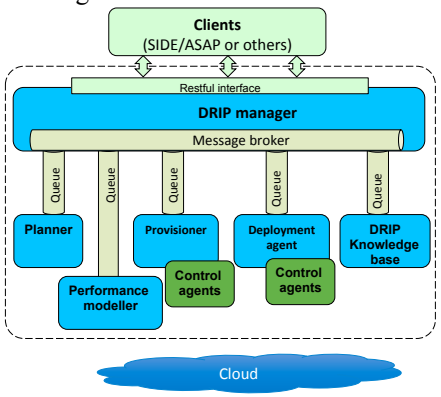

Fig. 2. DRIP implementation architecture.

1. The **infrastructure planner** uses an adapted partial critical path algorithm to produce efficient infrastructure topologies based on application workflows and constraints by selecting cost-effective virtual machines, customising the network topology among VMs, and placing network controllers for the networked VMs.
2. The **performance modeller** allows for testing of different cloud resources against different kinds of application component in order to provide performance data for use by the infrastructure planner and other components inside and outside of DRIP.
3. The **infrastructure provisioner** can automate the provisioning of infrastructure plans produced by the *planner* onto underlying infrastructure services. The provisioner can decompose the infrastructure description and provision it across multiple data centres (possibly from different providers) with transparent network configuration.
4. The **deployment agent** installs application components onto provisioned infrastructure. The deployment agent is able to schedule based on network bottlenecks, and maximize the satisfaction of deployment deadlines.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_51

5. The **infrastructure control agents** are a set of APIs that DRIP provides to applications to control the scaling containers or VMs and for adapting network flows. They provide access to the underlying programmability provided by the virtual infrastructures, e.g., horizontal and vertical scaling of virtual machines, by providing interfaces by which the infrastructure hosting an application can be dynamically manipulated at runtime.
6. The **DRIP manager** is implemented as a web service that allows DRIP functions to be invoked by outside clients as services. Each request is directed to the appropriate component by the manager, which is responsible for coordinating the individual components and scaling them if necessary. The manager also maintains a database containing user accounts.
7. The communication between the manager and the individual components is facilitated by a **message broker**. 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.
8. Resource information, credentials, and application workflows are all internally managed via a **knowledge base**. It maintains the descriptions of the cloud providers, resource types, performance characteristics, and other relevant information. The knowledge base also provides an interface for these agents to look up providers, resources and runtime status data during the execution of an application.

Fig. 3 depicts how those micro services interact.

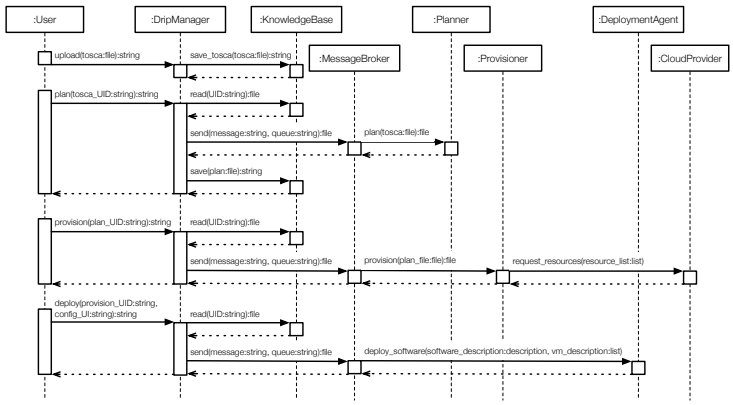

Fig. 3. Sequence diagram describing how DRIP plans and provisions virtual infrastructure and how it deploys software.

3.2 Current prototype

The prototype of DRIP is based on industrial and community standards. The *infrastructure planner* is currently specified in YAML (formerly 'Yet Another Markup Language' but now 'YAML Ain't a Markup Language') in compliance with the Topology

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_51

and Orchestration Specification for Cloud Applications (TOSCA)[1]. The *infrastructure provisioner* uses the Open Cloud Computing Interface (OCCI)[2] as its default provisioning interface, and currently supports the Amazon EC2[3], European Grid Initiative (EGI) FedCloud[4] and ExoGeni[5] Clouds. The *deployment agent* can deploy overlay Docker clusters using Docker Swarm or Kubernetes[6]. It may also deploy any type of customised distributed application based on Ansible playbooks[7]. The *infrastructure control agents* are set of API that DRIP provides to applications to control the infrastructure for scaling containers or VMs and adapting network flows. The *manager* provides a RESTful interface. DRIP uses the Advanced Message Queuing Protocol (AMQP) and RabbitMQ as its message broker where each process of each component is represented by a separate queue; this scalable architecture allows DRIP to be extended with additional components (e.g. planners) in order to handle larger workflows (e.g. in the case of a single DRIP service being provided to a large organisation for several applications).

The DRIP components are made available as open source under the Apache License Version 2.0; the software has been containerised and can be provisioned and deployed on federated virtual infrastructures within minimal configuration. They can be obtained either via the SWITCH release repository at https://github.com/switch-project or directly via the DRIP development repository at https://github.com/QCAPI-DRIP.

4 Experiments and performance characteristics

We will demonstrate how DRIP enhances the disaster early warning use case discussed in section 2.

As the first step, the application logic should be modelled as a Direct Acyclic Graph (DAG) with annotation of deadlines. Fig. 4 depicts the DAG of the scenario in Fig. 1. It will then be used as input for DRIP to automate the planning, provisioning, deployment of the application. In the early warning system workflow, 3 different deadlines can be defined as shown in Fig. 4. As the early warning system workflow is a service, the individual deadlines can be interpreted as deadlines in case data of a disaster is transmitted by the sensors in the field.

The planner in the DRIP system *uses a 'compress-relax' Multi dEadline workflow Planning Algorithm (MEPA)* method to assign each task in the workflow to the best performing VM possible such that multiple deadlines are met, as shown in Fig. 5. To find the best combination of assignments to nodes that fulfil all deadlines a Genetic Algorithm based Planning Algorithm is applied. The effectiveness of this approach is compared to a modification of the IC_PCP algorithm, Abrishami et al. [15] that allows IC-PCP to deal with multiple deadlines. Wang et al., 2017 [17] demonstrated the per-

[1] https://www.oasis-open.org/committees/tosca/
[2] http://occi-wg.org/
[3] https://aws.amazon.com/cn/ec2
[4] https://www.egi.eu/federation/egi-federated-cloud/
[5] http://www.exogeni.net/
[6] https://kubernetes.io/
[7] https://www.ansible.com/

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_51

formance of both approaches for task graphs generated by the GGen package [16] applying the 'fan-in/fan-out' methods, showing that the MEPA method can successfully cope with these kind of problems and allows for an easy adaptation in case more constraints play a role.

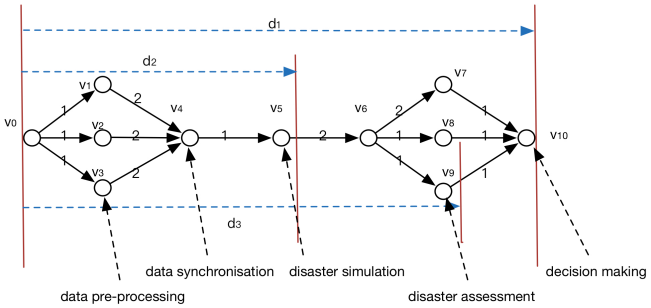

Fig. 4. Example of an abstract early warning system workflow with multiple deadlines. Global deadline d_1, and two intermediate deadlines d_2 and d_3 imposed on simulation and disaster assessment respectively.

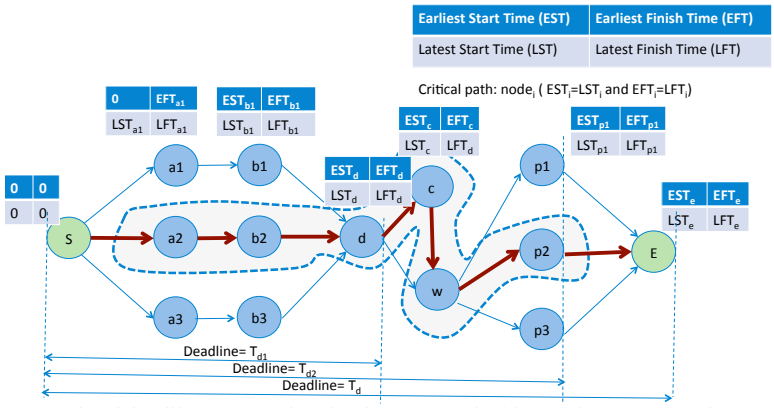

Fig. 5. Example of deadline-aware planning by DRIP. The blue nodes represent the workflow, with the critical path outlined. For each parallel group of nodes, the earliest/latest start/finish times can be extracted.

Planning heavily depends on the **Performance modeler** of the DRIP subsystem to collect performance information of cloud resources. It schedules on a regular basis one or more benchmark scenarios for different cloud providers. Information on CPU, memory, disk and network I/O are collected for different VMs offered by a cloud provider. The systematic collection and sharing of such information will allow the DRIP planner to select the most suitable resources for mission-critical applications. Elzinga et al. [19] showed the functionality of this collector using the ExoGENI infrastructure platform.

Once the planner is finished, **the provision agent** provides a *flexible inter-locale Cloud infrastructure provisioning* mechanism to satisfy time-critical requirements. It is able to provision a networked infrastructure, recover from sudden failures quickly, and scale across data centers or Clouds automatically [20, 24]. This Cloud engine is

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_51

able to set up a networked virtual Cloud across even public Clouds which do not explicitly support network topology, like EC2 or EGI FedCloud. For fast failure recovery the interplay of two agents, the provisioning agent and the monitoring agent. When some data center is down or inaccessible, a probe previously installed on the node can detect this. The monitoring agent can then invoke the provisioning agent to perform recovery. This is of importance in case sensors are geographically separated and data collections occurs in different cloud locations. The provisioning engine just needs to provision the specific part of the application hosted on the failed infrastructure. As the infrastructure description is already partitioned, it is easy for the agent to provision the same topology in another data center. Primary tests have been performed using the ExoGENI infrastructure platform; an example scenario is shown in Fig. 6.

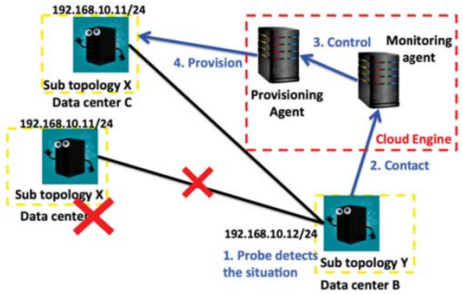

Fig. 6. Fast failure recovery.

Finally, **the deployment agent** provide a *deadline aware deployment scheduling* for time-critical applications in clouds comes into action, which accounts for deadlines on the actual deployment time of application components [21]. This is of special importance after fast failure recovery.

After the those steps, the application can be in operation for early warning, as shown in Fig. 7.

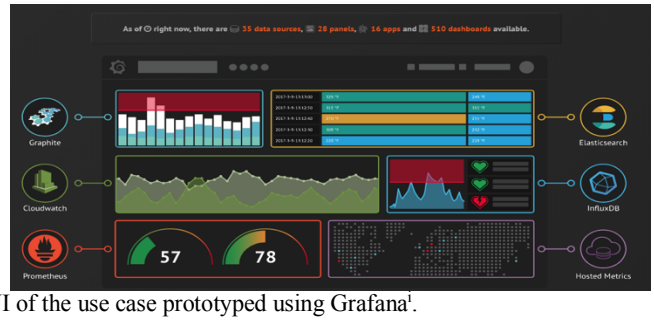

Fig. 7. The GUI of the use case prototyped using Grafana[1].

5 Summary

In this paper, we discussed the infrastructure challenges for meeting the time critical constraints for disaster early warning systems, and present a software suite called Dy-

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_51

namic Real-time Infrastructure Planner to automate the procedure for planning, provisioning and deploying early warning systems based on their time constraints. In the paper, the time critical constraints are not only referring to the as fast as possible but also to the deadlines that application has to meet.

There exist similar cloud engines for automating infrastructure provisioning such as Chef[8], also cloud job scheduling work based on IC_PCP algorithms [15]. However, compared to those existing work, DRIP shows the following unique features: 1) integrate infrastructure customization, provisioning and deployment into one service, to seamlessly bridge the gap between application and infrastructure, 2) time critical constraints are taken care of by different procedures.

We demonstrated the usage of DRIP in a specific type of application like early warning system; however, the purpose of DRIP meant to be generic. It has been used in several other use cases such as business collaboration, live event broadcast, and big data infrastructure.

One of the important future work will be further improve the optimization algorithm across the three steps of planning, provisioning and deployment.

Acknowledgement

This research has received funding from the European Union's Horizon 2020 research and innovation program under grant agreements 643963 (SWITCH project), 654182 (ENVRIPLUS project) and 676247 (VRE4EIC project).

References.

1. George Suciu, Victor Suciu, Cristina Butca, Ciprian Dobre and Florin Pop. Elastic disaster early warning system using a cloud-based communication center. Proceedings of the 13th IEEE Int'l Conf. on Intelligent Computer Communication and Processing (2017).
2. Zhiming Zhao, Paul Martin, Junchao Wang, Ari Taal, Andrew Jones, IanTaylor, Vlado Stankovski, Ignacio Garcia Vega, George Suciu, Alexandre Ulisses, Ceesde Laat, Developing and operating time critical applications in clouds: the state of the art and the SWITCH approach, In the proceedings of HOLACONF - Cloud Forward: From Distributed to Complete Computing, Elsevier, Procedia Computer Science, Vol (68) page (17-28) (2015)
3. Zschau J, Küppers AN, editors. Early warning systems for natural disaster reduction. Springer Science & Business Media; (2013)
4. Glade T, Nadim F. Early warning systems for natural hazards and risks. Natural Hazards. 70(3), pp. 1669; (2014).
5. De Groot, William J., and Michael D. Flannigan. Climate change and early warning systems for wildland fire. In Reducing Disaster: Early Warning Systems for Climate Change, pp. 127-151. Springer Netherlands, (2014).
6. Horita, F. E., Joaõ Porto de Albuquerque, Victor Marchezini, and Eduardo M. Mendiondo. A qualitative analysis of the early warning process in disaster management. In Proceedings of the ISCRAM2016 Conference–Rio de Janeiro, Brazil, (2016).
7. Cools, Jan, Demetrio Innocenti, and Sarah O'Brien. Lessons from flood early warning systems. Environmental Science & Policy 58, pp. 117-122, (2016).
8. Alhmoudi, A., and Z. U. H. Aziz. "Integrated framework for early warning system in UAE." International Journal of Disaster Resilience in the Built Environment, (2016).
9. Arcorace, Mauro, Francesco Silvestro, Roberto Rudari, Giorgio Boni, Luca Dell'Oro, and Einar Bjorgo. Forecast-based Integrated Flood Detection System for Emergency Response

[8] https://www.chef.io/chef/

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_51

and Disaster Risk Reduction (Flood-FINDER). In EGU General Assembly Conference Abstracts, vol. 18, p. 8770, (2016).
10. Udo, Job, and Nicole Jungermann. Early Warning System Ghana: how to successfully implement a disaster early warning system in a data scarce region. In EGU General Assembly Conference Abstracts, vol. 18, p. 12819, (2016).
11. P. A. Laplante, and S. J. Ovaska. Real-time systems design and analysis: tools for the practitioner. John Wiley and Sons. (2011)
12. S. Poslad, S. E. Middleton, F. Chaves, R. Tao, O. Necmioglu, and U. Bügel. A semantic IoT early warning system for natural environment crisis management. IEEE Transactions on Emerging Topics in Computing, 3(2), 246-257 (2015).
13. Zhiming Zhao, Paola Grosso, Jeron van der Ham, Ralph Koning, and Cees de Laat. An agent based network resource planner for workflow applications. Multiagent and Grid Systems, 7(6), 187-202. (2011).
14. Kieran Evans, Andrew Jones, Alun Preece, Francisco Quevedo, David Rogers, Irena Spasić, Ian Taylor, Vlado Stankovski, Salman Taherizadeh, Jernej Trnkoczy, George Suciu, Victor Suciu, Paul Martin, Junchao Wang, Zhiming Zhao. Dynamically reconfigurable workflows for time-critical applictions. In Proceedings of the 10th Workshop on Workflows in Support of Large-Scale Science (p. 7). ACM. (2015).
15. S. Abrishami, M. Naghibzadeh, and D. Epema. Deadline-constrained work- flow scheduling algorithms for infrastructure as a service clouds. Future Generation Computer Systems, 29(1):158–169, (2013).
16. D. Cordeiro, G. Mounié, S. Perarnau, D. Trystram, J. M. Vincent, and F. Wagner. Random graph generation for scheduling simulations. In Proceedings of the 3rd international ICST conference on simulation tools and techniques (p. 60). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). (2010).
17. Junchao Wang, Arie Taal, Paul Martin, Yang Hu, Huan Zhou, Jianming Pang, Cees de Laat, and Zhiming Zhao, Planning virtual infrastructures for time critical applications with multiple deadline constraints, Future Generation Computer Systems, (2017).
18. Junchao Wang, Cees de Laat, and Zhiming Zhao, Qos-aware virtual SDN network planning, in Proceedings of IFIP/IEEE International Symposium on Integrated Network Management. IEEE, (2017).
19. Olaf Elzinga, Spiros Koulouzis, Arie Taal, Junchao Wang, Yang Hu, Huan Zhou, Paul Martin, Cees de Laat, Zhiming Zhao, Automatic collector for dynamic cloud performance information, in Proceedings of 12th International Conference on Networking, Architecture, and Storage (2017).
20. Huan Zhou, Junchao Wang, Yang Hu, Jinshu Su, Paul Martin, Cees De Laat, Zhiming Zhao, Fast Resource Co-provisioning for Time Critical Application Based on Networked Infrastructure, IEEE International Conference on CLOUD, San Francisco US (2016).
21. Yang Hu, Junchao Wang, Huan Zhou, Paul Martin, Arie Taal, Cees De Laat, and Zhiming Zhao, Deadline-aware deployment for time critical applications in clouds, in 2017 International European Conference on Parallel and Distributed Computing, (2017).
22. Zhiming Zhao, Arie Taal, Andrew Jones, Ian Taylor, Vlado Stankovski, Ignacio Garcia Vega, Francisco Jesus Hidalgo, George Suciu, Alexandre Ulisses, Pedro Ferreira, Cees de Laat, A software workbench for interactive, time critical and highly self-adaptive cloud applications (SWITCH), In the proceedings of IEEE CCGrid (2015)
23. Zhiming Zhao, Dick van Albada, Peter Sloot. *Agent-based flow control for HLA components*. International Journal of Simulation Transaction. 81(7), 487-501. (2005).
24. Huan Zhou, Yang Hu, Junchao Wang, Paul Martin, Cees De Laat, Zhiming Zhao, Fast and Dynamic Resource Provisioning for Quality Critical Cloud Applications, IEEE International Symposium On Real-time Computing (ISORC), York UK, (2016).

[i]GRAFANA: The open platform for beautiful analytics and monitoring: https://grafana.com/

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_51