2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing

A Software Workbench for Interactive, Time Critical and Highly self-adaptive cloud applications (SWITCH)

Zhiming Zhao[a] Arie Taal[a] Andrew Jones[b] Ian Taylor[b] Vlado Stankovski[c] Ignacio Garcia Vega[d]
Francisco Jesus Hidalgo[d] George Suciu[e] Alexandre Ulisses[f] Pedro Ferreira[f] Cees de Laat[a]
[a] University of Amsterdam, the Netherlands {Z.Zhao|A.Taal|DeLaat}@uva.nl
[b] Cardiff University, UK, {JonesAC|TaylorIJ1}@cardiff.ac.uk
[c] University of Ljubljani, Slovenia, vlado.stankovski@fgg.uni-lj.si
[d] Wellness Telecom SL, Spain, igarcia@wtelecom.es
[e] BEIA Consult International SRL, Romania, george@beia.ro
[f] MOG Technologies SA, Portugal, {alexandre.ulisses|pedro.ferreira}@mog-technologies.com

***Abstract*—Time critical applications have very high requirements on network and computing services, in particular on well-tuned software architecture with sophisticated optimisation on data communication. Their development is often customised to dedicated infrastructure, and system performance is difficult to maintain when infrastructure changes. This fatal weakness in existing architecture and software tools causes very high development costs, and makes it difficult to fully utilise the virtualised, programmable and quality-on-demand services provided by networked Clouds to improve the system productivity. The Software Workbench for Interactive, Time Critical and Highly self-adaptive Cloud applications (SWITCH) is a newly funded project by EU H2020 to address this urgent industrial need; it aims at improving the existing development and execution model of time critical applications by introducing a novel conceptual model called *application-infrastructure co-programming and control model*, in which application QoS/QoE together with the programmability and controllability of Cloud environments can be all included in the complete lifecycle of applications.**

Key words: Time critical applications, Cloud, self adaptability.

I. Introduction

Many industrial applications have highly time-critical requirements for their performance in order to maintain their business value. For instance the application may address Quality of Service (QoS) (e.g. tsunami emergency response time) or quality of experience (QoE) (e.g. delivery of ultra-high definition television, or collaborative business interactions) issues. These applications are often called time critical applications. Time critical applications often involve distributed components, and intensive data communication. For instance applications which address disaster warning issues often include remotely deployed sensors, and many live event television broadcast scenarios require direction of multiple outdoor video sources. 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 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 methods and software tools for developing, deploying and executing classical time critical applications do not, as yet, include the programmability and controllability provided by Clouds; and time critical applications cannot yet get the full potential benefits which Cloud technologies could provide. A Software Workbench for Interactive, Time Critical and Highly self-adaptive Cloud applications (SWITCH) is thus proposed for developing software methods and tools for the entire lifecycle of time critical Cloud applications. This poster illustrates the main idea and approach of the SWITCH project.

II. State of the art and objectives

The development of time critical applications faces multiple challenges, which can be seen from the SWITCH industrial example scenarios. For instance, enabling collaborative real-time business communication networked Cloud services, collecting and processing the sensor data in nearly real time to detect and respond to urgent events, and directing and broadcasting live events by using virtualized video switches. This section analyses main requirements, reviews the state of the art, and defines objectives of our research.

A. Time critical Cloud applications and requirements

Several requirements can be enumerated from the above scenarios: 1) system level performance requirements must be satisfied to deliver acceptable performance; 2) verifiability for proving that the application will actually achieve the desired QoS; 3) integration complexity of the applications require specialised knowledge on modelling and controlling software and infrastructure parameters; 4) use of virtualized resources require profound Cloud knowledge on selection of appropriate cloud platforms and programming models; 5) configuration of the infrastructure requires a common, reusable interface for programming the infrastructure and must

978-1-4799-8006-2/15 $31.00 © 2015 IEEE
DOI 10.1109/CCGrid.2015.73

be tailored to achieve the required response times; 6) data intensive communication in the application require virtualized storage and communication configurations based on functional descriptions and to characterise data throughput and latency amongst application components; 7) adaptability for quality-on-demand: applications increasingly need to be configured to cope with quality-on-demand in the run-time environment; 8) adaptability for infrastructure in order to maintain the system level of performance; and 9) SLA negotiation is required when adapting infrastructure to enforce explicit SLA with (federated) Cloud providers dynamically.

These requirements cover the entire lifecycle of time critical Cloud applications: including development methods, verification, programming, deployment, and runtime control. Software Defined Networking (SDN) and Cloud technologies provide an elastic and flexible way of configuring and reconfiguring the infrastructure as needed (much more flexible than the traditional approach of configuring individual switches, firewalls, etc., directly) [1]. However, it is becoming increasingly apparent that the development of such time critical Cloud applications presents complex requirements to programmers as discussed above. There is therefore an urgent need to develop a new approach to developing such applications; this is the purpose of SWITCH, as we shall now explain in more detail.

B. State of the art

Support for time critical applications in Cloud environments is still at a very early stage, especially for applications which should be self-adaptable in order to maintain the required system performance. We shall review the state of the art from the three most relevant technical aspects: 1) distributed application programming, 2) advanced infrastructure, and in particular programmable infrastructure, and 3) self-adaptable performance control.

Programming distributed applications often depends on the adoption of a specific computing architecture or platform; typical examples include Message Passing Interface (MPI)-based parallel computing in a distributed memory cluster architecture, service platform-based workflow applications, and a Cloud-based Map Reduce computing model. Quality constraints are used in workflow applications for describing the abstract workflows, and for creating the runtime enactment, such as in [2] [3] [4]. However, in those applications, the creation of the application logic is mostly separated from the customisation of the runtime environment; in particular, a formal model is rarely utilised in verifying the time constraints. Compared to purely network level protocol optimisation, such as multiple path TCP, these SDN technologies allow applications 1) to customize network connectivity between services by defining suitable flow forwarding tables, or by reserving dedicated links, 2) to virtualize the network resources for different partition schemas by tuning the network slice for given set of computing and storage nodes, and 3) to control the network quality of service by either advanced reservation of links or dynamically controlling the packet flows. However, including these new features in data delivery services is still at a very early stage. Service Level Agreement (SLA) issuing and real-time negotiation technologies depend heavily on the complexity of the mapping between application requirements and the available resources, and the matching among quality requirements at different service layers [5]. Self-adaptable software architecture and performance control has attracted substantial attention amongst software engineering researchers during the past decade. Semantically harmonising different kinds of monitoring information, and in particular harmonizing this information with the application logic, still remains a challenge for self-adaptable systems. The self-adaptable mechanism has been implemented mainly using a so-called architecture style [6], in which the internal structure of the application architecture is explicitly modelled and the application can manipulate the structure at runtime based on the results of certain decision-making procedures. Esfahani et al., proposed a different approach based on the application features and use a machine learning mechanism to implement adaptability [7]. These early works either focus only on the application control, or only on the quality of the service from the provider point of view; they do not fully explore the adaptability of system which contain programmable infrastructure.

From the above review, we can see that current time-critical application programming models lack consideration of the controllability of the infrastructure; they thus do not exploit the potential benefits offered by the programmable infrastructure provided by Cloud environments. There have been many studies on the application of various optimisation mechanisms in selecting resources. However, there is currently no semantically well-modelled mapping between the application quality of user experiences, and the infrastructure level QoS attributes. The existing self-adaptation intelligence focuses either on controlling application architecture or on the service quality of the runtime infrastructure; there is a lack of co-controlling mechanism addressing both application control and the programmability of the runtime infrastructure.

C. Motivation

We propose a software workbench, namely Software Workbench for Interactive, Time Critical and Highly self-adaptive cloud applications (SWITCH). The overall objective of the SWITCH project is to address the entire life-cycle of time-critical, self-adaptive Cloud applications by developing new middleware and front-end tools to enable users to specify their time-critical requirements for an application interactively using a direct manipulation user interface, deploy their applications and adapt the infrastructure to changing requirements either automatically (using the specified requirements) or by human intervention if desired.

The SWITCH project addresses these problems by providing an interactive and flexible software workbench that, by using discovery tools at the networking level and QoS requirements from the application level, can provide the tools necessary to control the lifecycle for rapid development, deployment, management and dynamic reconfiguration of complex distributed time-critical cloud applications. In particular,

SWITCH provides novel support for defining, optimising and controlling time-critical constraints in programming, testing, deploying and executing the applications. Using a fully responsive web based interface and backend components for coordinating the data flows across the networking infrastructure, the SWITCH workbench can define dynamic application level mappings for the time critical control rules and strategies to be employed on an application-by-application basis.

III. Software Workbench for Self-Adaptable Time Critical Applications

At the core idea of the SWITCH environment, a new development and execution model, an application-infrastructure co-programming and control model, will be developed for time-critical Cloud applications. The new model brings together the application composition, execution environment customisation, and runtime control, which are normally treated in separated processes, into one optimisation loop based on the time critical requirements. In this model: 1) the application logic will be programmed with considerations of the system QoS/QoE together with the programmability and controllability of the Cloud environment; both application and the virtual runtime environment for executing the application will be programmed and optimised at the design phase; 2) a virtual runtime environment (a runtime environment created in the Cloud for executing the application) can be customised for the critical application requirements, and can be provisioned in the Cloud with time critical application oriented Service Level Agreement (SLA); and 3) the application can autonomously adapt the behaviour of its own and the virtual runtime environment when performance drops at runtime. The SWITCH environment employs formal performance reasoning mechanisms to guide each step in the development and tools are delivered to the users via three subsystems, which are shown in the conceptual diagram, Figure 1.

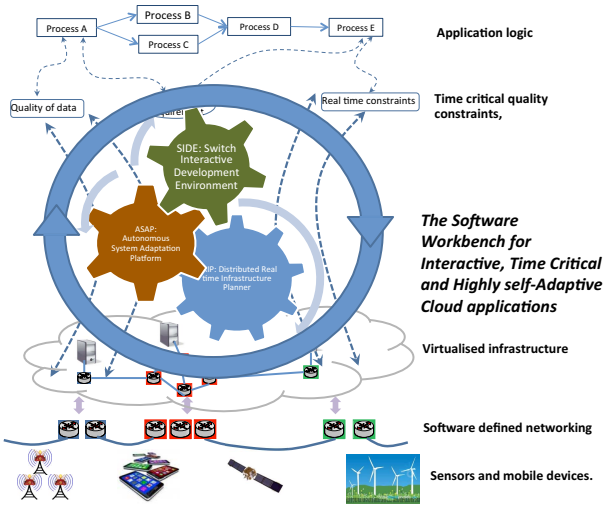

Fig. 1. The basic idea of SWITCH.

A. The SIDE sub system

The SWITCH Interactive Development Environment (SIDE) subsystem provides interfaces for all of the user- and programmer-facing tools, by exposing a collection of graphical interfaces and APIs that tie in SWITCH's services to a Web-based environment. SIDE will be engineered using fully responsive HTML5 on the front end, providing interactivity and end-user device portability, and the back-end Web services (hooks) and APIs will be constructed using Python and tools such as Django, Flask, or similar. The SIDE subsystem therefore will work closely with all other work packages in order to gather requirements and devise common API constructs and guidelines for implementing and exposing SWITCH functionality.

B. The DRIP sub system

The Dynamic Real-time Infrastructure Planner (DRIP) subsystem prepares the execution of the applications developed in the SIDE subsystem by 1) semantic modelling and linking for different QoS/QoE attributes, 2) defining an optimal virtual runtime environment, 3) creating a Service Level Agreement with the resource provider, and 4) deploying the platform required by the application.

C. The ASAP sub system

The Autonomous System Adaptation Platform (ASAP) 1) monitors the status of the application and the runtime environment, 2) examines the actual performance of the required quality attributes, 3) autonomously controls the application and runtime environment to maintain optimal system level performance against the time critical constraints, and 4) learns from its own decision history to improve its intelligence in making future decisions for autonomous control.

The SWITCH environment will provide lightweight libraries that can be used to encapsulate the application as self-contained bundles, for deployment to a runtime environment as a standalone system.

D. How the system works

To better illustrate the idea of SWITCH and to describe how these subsystems work together, we provide a sequence of events that might occur in the development and deployment of a typical pilot SWITCH application in Figure 2.

The application developer begins with composing the application logic and defining the QoS constraints, such as the latency for state visualisation, sensor event handling delay (step 1). The developer can also give an abstract network overlay to define the runtime environment (step 2). These activities can be optimized and aided using a knowledge base of successful patterns of applications and infrastructures, and a formal reasoning component (step 3). The results of step 1 and 2 will be passed from SIDE to DRIP; the developer can also specify requirements such as resource providers to be used, and the total cost budget for application execution (step 4). DRIP plans the concrete virtual runtime environment of computing, storage and network elements by reasoning with

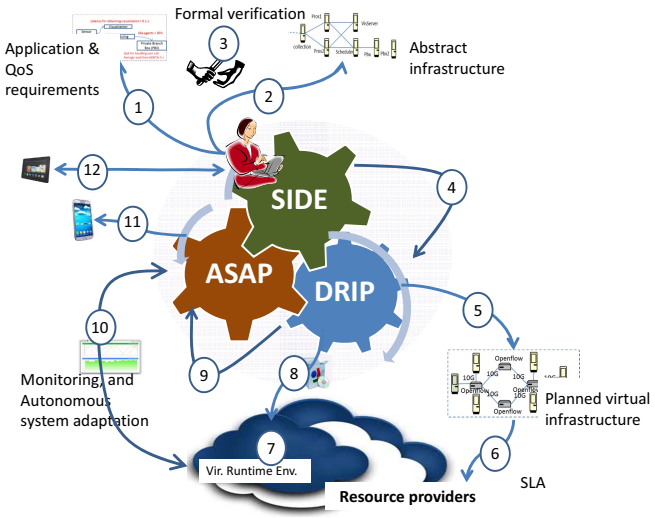

Fig. 2. How SWITCH works.

the application-level QoS constraints (step 5). Then, DRIP generates Service Level Agreements (SLAs) with the resource provider(s) (step 6), and the resource provider(s) provision the virtual environment (step 7). After that, DRIP customises the virtual environment and deploys necessary services for the application (step 8), and unbundles and executes the application (step 9). At runtime, SIDE allows the user to: query and visualise the runtime status of the application and runtime environment (step 10), receive notification of system status and inform the user (step 11), directly manipulate the system execution (step 12), and expose real-time monitoring information from the database, which is generated by ASAP. The ASAP subsystem can monitor the runtime status of the application and its environment, and pass this information (via the backend database) to SIDE, diagnose system performance and make decisions on control actions needed to restore performance where necessary, take action to maintain system performance, and learn from the history in order to improve the subsequent effectiveness of the decision making procedure (step 10).

IV. Use cases

Several industrial use cases will be used to validate and demonstrate the functionality of the SWITCH system. One of the examples is disaster early warning. Early warning 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.

In this use case, the SIDE subsystem will provide the developer of early warning systems an intuitive interface to 1) describe the application logic among: sensor data collection, data storage, processing, activation of warning services, and virtual call centre facilities, 2) define quality requirements at system level and/or at the level of each individual process, SIDE providing a formal mechanism to validate the time requirements, and creating the constraints for application development, 3) describe the cost, quality requirements for the runtime environment on Cloud, and select potential Cloud providers. Optionally, the user can also describe an abstract virtual infrastructure. The DRIP subsystem will create a concrete virtual runtime environment based on the input from SIDE, and negotiate with the resource providers to establish SLAs, and deploy the services to the virtual runtime infrastructure after it is provisioned. The ASAP subsystem will detect the key quality attributes such as potential load, communication latency between sensors, and average customer waiting time for getting connected with the call centre, and dynamically tune the execution of the application, by adapting the application components, or deploying new virtual call centre components in Cloud to rectify the possible performance drops.

The intention is for SWITCH to be a generic platform, rather than it being tailored specifically for only these use cases, however diverse they might be from each other.

V. Summary

In this poster, we introduced the basic idea and approach of a newly funded EU H2020 project SWITCH. The project started February 2015 and will last three years. The software of SWITCH will be open source. The consortium has half members from industry, and the other half are from academic institutions; the development of the SWITCH software will be tested and validated using industrial use cases. Moreover, the industrial partners will actively explore the future market value of the development.

VI. Acknowledgement

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 643963 (SWITCH project).

References

[1] Ying-Dar Lin, Dan Pitt, David Hausheer, Erica Johnson, and Yi-Bing Lin. Software-defined networking: Standardization for cloud computing's second wave. *Computer*, 47(11):19–21, Nov 2014.

[2] Zhiming Zhao, Paola Grosso, Jeroen van der Ham, Ralph Koning, and Cees de Laat. An agent based network resource planner for workflow applications. *Multiagent Grid Syst.*, 7(6):187–202, November 2011.

[3] Mattijs Ghijsen, Jeroen Van Der Ham, Paola Grosso, Cosmin Dumitru, Hao Zhu, Zhiming Zhao, and Cees De Laat. A semantic-web approach for modeling computing infrastructures. *Comput. Electr. Eng.*, 39(8):2553–2565, November 2013.

[4] Hao Zhu, Karel van der Veldt, Zhiming Zhao, Paola Grosso, Dimitar Pavlov, Joris Soeurt, Xiangke Liao, and Cees de Laat. A semantic enhanced power budget calculator for distributed computing using ieee 802.3az. *Cluster Computing*, 18(1):61–77, 2015.

[5] C. Muller, M. Oriol, X. Franch, J. Marco, M. Resinas, A. Ruiz-Cortes, and M. Rodriguez. Comprehensive explanation of sla violations at runtime. *Services Computing, IEEE Transactions on*, 7(2):168–183, April 2014.

[6] Betty H. Cheng, Rogério Lemos, Holger Giese, and et al. Software engineering for self-adaptive systems. chapter Software Engineering for Self-Adaptive Systems: A Research Roadmap, pages 1–26. Springer-Verlag, Berlin, Heidelberg, 2009.

[7] N. Esfahani, A. Elkhodary, and S. Malek. A learning-based framework for engineering feature-oriented self-adaptive software systems. *Software Engineering, IEEE Transactions on*, 39(11):1467–1493, Nov 2013.