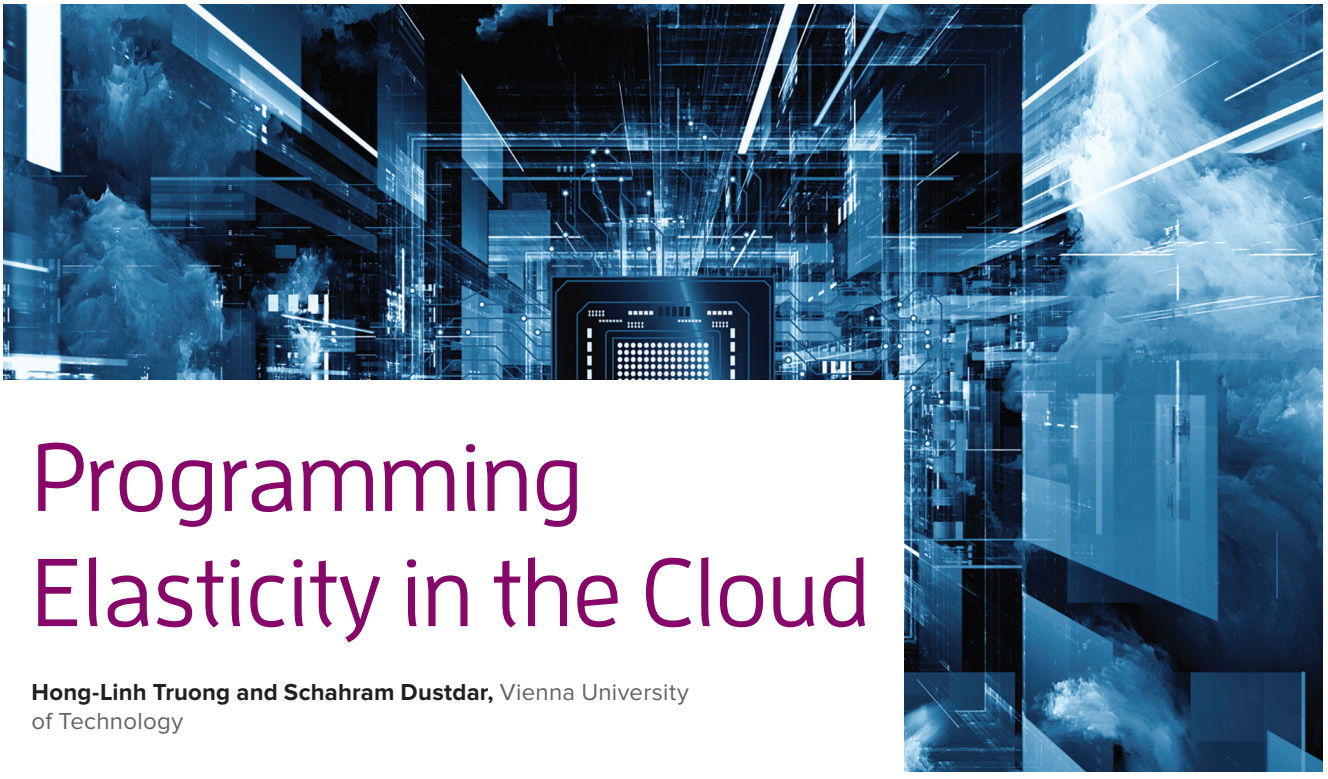
EDITOR **SAN MURUGESAN**
BRITE Professional Services, Sydney, Australia;
san@computer.org



Programming Elasticity in the Cloud

Hong-Linh Truong and Schahram Dustdar, Vienna University of Technology

As a virtualized environment, the cloud offers considerable flexibility. Maximizing its benefits requires incorporating elasticity as a "first-class" property throughout the cloud software development life cycle.

With its attractive pay-per-use model and the opportunities it provides for flexible resource scaling, cloud computing is increasingly the go-to choice for developing and deploying software. In recent years, various multitiered applications and data analytics workflows have been successfully ported to the cloud, proving that the cloud can greatly simplify the software development life cycle when dynamic functionality is a primary goal.

NEW PROGRAMMING POSSIBILITIES

Cloud software infrastructures offer a new virtualization environment that enables developers to "program elasticity": that is, to create and execute multiple cloud objects—including sensors, virtual machines, OS and application containers, network functions, and data streams—so that their operation, quality, and cost to end users can vary elastically at runtime.[1] Ideally, then, cloud software should be constructed as a cloud-native application,[2] intrinsically incorporating elasticity.

Consider, for example, a cloud software for predictive maintenance, as illustrated in Figure 1. Data collected by sensors and processed locally is relayed via gateways to the cloud, where various Web services store and analyze it depending on the end purpose. Cloud software like this can involve several different cloud objects—for example, load balancers, message-oriented middleware (MOM), and data analytics in addition to sensors, gateways, and so forth. At runtime, depending on predictive maintenance requirements, it's possible to deploy and activate additional sensors or alter the data-sensing rate to obtain more reliable, accurate analytics data.

Because the data—and the data flow—can vary according to these changing data requirements, the connectivity between gateways and cloud infrastructures also changes,

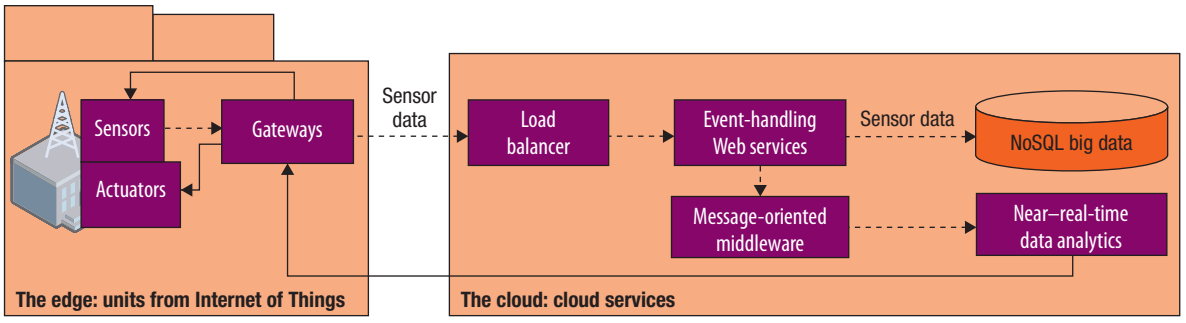


Figure 1. Cloud software for predictive maintenance. Sensor data can vary according to predictive maintenance requirements, creating a need for elasticity throughout the process.

altering load-balancing strategies, MOM resource allocation, algorithm execution for data analytics, and so on. Such a cloud software requires novel techniques to account for and deal in a coordinated manner with this elastic behavior at different stages in the process and for various parts of the cloud software.

LITTLE SUPPORT FOR ELASTICITY THROUGH THE CLOUD SOFTWARE LIFE CYCLE

Ideally, elasticity should be programmed natively into cloud-deployed software applications. Currently, however, doing so is quite complex. We don't yet have tools to program elasticity within cloud objects during the design stage, primarily because we still consider elasticity a nonfunctional feature to be "added" after cloud objects are programmed, using separate control mechanisms to support elasticity.

Various tools and techniques based on well-known application models have been introduced, in both industry and academia, to support programming cloud software. However, most don't support elasticity throughout the software development process.

Programming models and techniques

Conventional programming languages and models allow us to easily define cloud-object functions and interfaces, and approaches have been developed for cloud model–driven architectures[3] that support modeling cloud-object properties in terms of cost and quality. But these are designed mainly for selecting components, and are inadequate for several reasons:

- they don't include elasticity, dynamicity, and pay-per-use as primary elements in the programming;
- they require separate components to achieve elasticity features, which are still limited; and
- they lack APIs for managing elasticity properties.

Elasticity control

Rules-based approaches (http://aws.amazon.com/autoscaling) allow us to specify if-then-else rules for controlling elasticity. However, these are low level and mainly serve infrastructure-as-a-service (IaaS) models. In addition, they're decoupled from programming activities and from high-level application object behavior.

Elasticity monitoring

System monitoring tools allow resource monitoring and provide detailed system information with traditional metrics, while application monitoring tools provide coarse-grained information about performance and costs. But it's difficult to correlate these to high-level application objects, and they lack elasticity metrics and state monitoring.

Deployment

Cloud software deployment tools such as Chef[4] and declarative languages such as TOSCA[5] and Google's deployment template (https://developers.google.com/deployment-manager/overview) are available. However, current deployment tools like these operate at a low or single level, while multiple objects exist in cloud software, so they don't integrate well with elastic object models.

Testing

Although approaches are available that support generating and executing test cases in the cloud,[6] we lack test cases geared specifically toward understanding elasticity behaviors or toward evaluating elasticity control strategies for cloud operations.

PROGRAMMING ELASTICITY IN CLOUD SOFTWARE

While most cloud software enables limited elasticity, it isn't achieved in an integrated way throughout the programming, deployment, control, and testing process. Instead, separate components are used to take care of elasticity. And while we can use appropriate programming languages to create objects that represent cloud service functions, we can't easily associate elasticity with these objects—for example, we can't describe the elastic relationship between cost and quality as an object's intrinsic property.

To incorporate elasticity as a central part of cloud programming, we

need a new conceptual model for elastic cloud objects and programming languages that will allow us to construct cloud software from objects whose elasticity can be programmed via software-defined APIs and incorporated into cloud software runtime systems for deployment, control, monitoring, and testing.

The model should also allow us to hide this complexity so users don't have to deal with multiple tools when programming for elasticity. Ideally, the user would be able to compose software from many elastic objects, deploy the software based on its requirements and the best available cloud information, and then continuously test and control the software's elasticity to meet these requirements while optimizing its running costs and quality.

CONCEPTUALIZING ELASTIC CLOUD OBJECTS

As noted earlier, cloud providers already offer multiple service units, including virtual machines, OS and Web service containers, databases, and virtual network connections. A new model could allow runtime objects in cloud software to represent execution environments as well as middleware and application components. When developing elastic cloud software, we must conceptually take these elastic objects into account along with the interactions among them, which are also elastic, and consider them instances of service units that can be deployed, controlled, monitored, and tested on the fly. From this standpoint, we can describe elastic objects (such as a Web service) and their required objects (such as a Web container) as *elastic service units*.[7]

Conceptual models for an elastic service unit should

- function with a well-defined interface;
- offer different service provisioning, consumption, quality-management, and pricing models;
- have dependent service units; and
- have elastic capabilities.

DEVELOPING ELASTIC CLOUD SOFTWARE

To develop elastic software, cloud service providers should offer fundamental services—virtual machines, MOM, network functions, and the like—as elastic service units, and programmers should use these services in implementing cloud software. Furthermore, any developer should be able to bring his or her software's elastic service units into the cloud software ecosystem. To achieve these goals, we need to focus on elements in various phases of elastic cloud software development and operation.

Programming languages and frameworks

Programming relies on a range of tools, but, as we observed, most current tools follow a tradition in which objects are developed in existing programming languages and different components are created and used to control and test the elasticity. This doesn't enable a good correlation between programming for elasticity and runtime support for elasticity. Changing this will require developing new programming models and languages. Current approaches using directives and annotations for dynamic configuration, life-cycle management, and control offer a potential starting point.

Controlling elastic objects

On one hand, we need high-level elasticity specifications suitable for users but that an elasticity controller can support as well. Users should also be able to specify elasticity when programming the software[8]—that is, elasticity should be considered a "first-class" entity in cloud programs. On the other hand, the elasticity capabilities offered by service units must be clearly defined through primitive operations available to users during programming. Clearly, this requires that cloud software developers and providers explicitly build elasticity into APIs to control elastic cloud objects.

Monitoring and analyzing elasticity

We must be able to define what constitutes "elasticity behavior" so that we can monitor and analyze cloud software elasticity. Another important perspective is the "elasticity dependency" among elastic objects. Although several software benchmarks and models have been proposed,[9,10] we currently lack metrics representing the elasticity behavior of cloud software. Since we must view elasticity multidimensionally, concepts of elasticity space, pathway, and boundaries[11] could be useful for investigating elasticity dependencies and metrics.

Deploying and configuring for elasticity

Deployment and configuration techniques must deal with different elastic objects at different levels of abstraction. Chef and other popular techniques for deploying the whole software stack (generally before objects are instantiated) aren't enough. Unless elastic objects are equipped with runtime APIs to control their elasticity capabilities, deployment and configuration will be limited. We also need to integrate elasticity monitoring and analysis features for deployment and configuration, and to offer interfaces for elasticity controllers.

Testing elasticity

Testing now relies on traditional metrics;[6] as yet, we lack real elasticity metrics. Therefore, we need to determine the properties we want to test and develop the appropriate metrics—that is, metrics bound specifically to elasticity behavior, which currently don't exist—to complement existing techniques for generating test cases, all of which can be executed in the cloud.

Another question we face is whether to continue following a traditional test-deploy-execute approach or rather to consider an interwoven

CALL FOR CLOUD COVER COLUMN CONTRIBUTIONS

We welcome short articles (1,200 to 1,500 words) for publication in this column that discuss your ideas for advancing cloud computing or share your experiences in harnessing the cloud. We also solicit articles on topics such as fog computing, cloudlets, cloud forensics, cloud aggregation and integration, service level agreements, and legal issues. Send proposals or submissions to San Murugesan at cloudcover@computer.org.

For a list of previous Cloud Cover columns, visit http://tinyurl.com/computer-cloudcover.

deploy-control-execute-test approach. This question is particularly important because it could fundamentally change current testing methods, not just the metrics used in testing. If elastic objects can change and be changed over time, why should we test and then deploy, rather than deploy and then test?

RECOMMENDATIONS

A first step for developers is to improve current cloud objects in ways that associate elasticity properties and relationships with the objects directly. Compilers, interpreters, and code generators should map these properties to specific underlying elasticity primitives available in shared libraries. Such elasticity primitives should be generic and so apply to different types of cloud objects, but also be independent from cloud-specific technologies and infrastructures.

Cloud providers should offer mappings from their specific APIs that help developers implement elasticity capabilities for shared elasticity primitives. Runtime for cloud programming frameworks should focus on executing elasticity primitives—based on monitoring information about elasticity properties and relationships—instead of dealing with cloud vendor–specific APIs.

As a virtualized environment, the cloud allows us to manipulate and invoke objects dynamically and economically. Therefore, we need to rethink how best to program elasticity within the cloud. This requires novel techniques and tools that interweave design, deployment, testing, execution, and control activities for elastic software. Future work should focus on studying and creating languages, techniques, and frameworks that allow us to develop intrinsic elasticity properties as first-class entities in the cloud software development life cycle.

HONG-LINH TRUONG is an assistant professor in the Distributed Systems Group at the Vienna University of Technology, where his work focuses on service. Contact him at truong@dsg.tuwien.ac.at.

SCHAHRAM DUSTDAR is a full professor of computer science and heads the Distributed Systems Group at the Vienna University of Technology. Contact him at dustdar@dsg.tuwien.ac.at.

ACKNOWLEDGMENTS

This work was partially supported by the European Commission in accordance with terms of the CELAR FP7 project (FP7-ICT-2011-8 #317790).

REFERENCES

1. S. Dustdar et al., "Principles of Elastic Processes," *IEEE Internet Computing*, vol. 15, no. 5, 2011, pp. 66–71.
2. C. Fehling et al., *Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications*, Springer, 2014.
3. N. Ferry et al., "Towards Model-Driven Provisioning, Deployment, Monitoring, and Adaptation of Multi-cloud Systems," *Proc. IEEE 6th Int'l Conf. Cloud Computing* (CLOUD 13), 2013, pp. 887–894.
4. G. Katsaros et al., "Cloud Application Portability with TOSCA, Chef, and Openstack," *Proc. 2014 IEEE Int'l Conf. Cloud Eng.* (IC2E 14), 2014, pp. 295–302.
5. T. Binz et al., "Portable Cloud Services Using TOSCA," *IEEE Internet Computing*, vol. 16, no. 3, 2012, pp. 80–85.
6. L. Riungu-Kalliosaari, O. Taipale, and K. Smolander, "Testing in the Cloud: Exploring the Practice," *IEEE Software*, vol. 29, no. 2, 2012, pp. 46–51.
7. H.-L. Truong et al., "CoMoT—A Platform-as-a-Service for Elasticity in the Cloud," *Proc. 2014 IEEE Int'l Conf. Cloud Eng.* (IC2E 14), 2014, pp. 619–622.
8. G. Copil et al., "SYBL: An Extensible Language for Controlling Elasticity in Cloud Applications," *Proc. 13th IEEE/ACM Int'l Symp. Cluster, Cloud, and Grid Computing* (CCGRID 13), 2013, pp. 112–119.
9. S. Islam et al., "How a Consumer Can Measure Elasticity for Cloud Platforms," *Proc. 3rd ACM/SPEC Int'l Conf. Performance Eng.* (ICPE 12), 2012, pp. 85–96.
10. B.F. Cooper et al., "Benchmarking Cloud Serving Systems with YCSB," *Proc. 1st ACM Symp. Cloud Computing* (SoCC 10), 2010, pp. 143–154.
11. D. Moldovan et al., "MELA: Monitoring and Analyzing Elasticity of Cloud Services," *Proc. 5th IEEE Int'l Conf. Cloud Computing Technology and Science* (CloudCom 13), vol. 1, 2013, pp. 80–87.