A Generic Framework for Service-based Business Process Elasticity in the Cloud

Mourad Amziani[1,2], Tarek Melliti[2], and Samir Tata[1]

[1] Institut Mines-Telecom, TELECOM SudParis, UMR CNRS Samovar, Evry, France
[2] University of Evry Val d'Essonne, IBISC, Evry, France

Abstract. Cloud computing is a new model for the provisioning of dynamically elastic and often virtualized resources at the levels of infrastructures, platforms and software. Cloud platforms are being increasingly used for the deployment and execution of service-based business processes (SBPs). Nevertheless, the provisioning of elastic infrastructures and/or platforms is not sufficient to provide users with elasticity at the level of SBPs. Therefore, there is a need to provide SBPs with mechanisms to scale their resource requirements up and down whenever possible. This can be achieved using mechanisms for duplicating and consolidating business services that compose the SBPs. In this paper, we propose a formal model and a generic framework for elasticity of SBPs.

1 Introduction

Cloud computing is a new delivery model for IT services based on Internet protocols. It typically involves provisioning of dynamically scalable and often virtualized resources at the infrastructure, platform and software levels. Cloud environments are being increasingly used for deploying and executing business processes and particularly service-based business processes (SBPs) that are made up of components that provide business services. One of the expected facilities of cloud environments is elasticity at different levels.

At the platform as a service (PaaS) level, the deployed processes should be provided with platform mechanisms that can scale up and down whenever needed. In this context, we have conducted studies of existing application servers and SBP engines. These classical platforms are not elastic [6]. For that reason, we have developed a new model for service deployment called micro-container [8]. Our approach was based on a simple idea that consists in dedicating a micro-container with minimal and personalized functionalities to manage the life cycle of each deployed services. With this idea we have shown the elasticity of services deployed at the PaaS level can be ensured [8]. In addition, we have shown that elastic micro-containers can be used to host service-based application.

Nonetheless, provisioning of elastic platforms, *e.g.* based on micro-containers, is not sufficient to provide users with elasticity of the deployed business process (at the Software as a Service (SaaS) level). Therefore, SBPs should be provided with elasticity so that they would be able to adapt to the workload changes while ensuring the desired functional and non-functional properties. In this paper we address elasticity at the level of SBPs that mainly raises the following questions.

– What mechanisms should be developed to perform elasticity of SBPs?
– How to evaluate elasticity strategies of SBPs?

Performing elasticity consists in providing cloud environments with mechanisms that allow deployed SBPs to scale up or down. To scale up a SBP, elasticity mechanisms have to create, as many copies as necessary, of some business services (part of the considered SBP). To scale down a SBP, elasticity mechanisms have to remove unnecessary copies of some services.

Many strategies that decide on when SBP elasticity is performed can be proposed. It would be useful for a Cloud provider to have an evaluation framework in order to make a better decision on the elasticity strategy to adopt.

In this paper, we propose a formal model for SBP elasticity and a framework to evaluate elasticity strategies.

2 Model for SBPs elasticity

We are interested in this paper in modelling elasticity of SBPs. A SBP is a business process that consists in assembling a set of elementary IT-enabled services. These services realise the business activities of the considered SBP. Assembling services into a SBP can be ensured using any appropriate service composition specifications (*e.g.* BPEL). Elasticity of a SBP is the ability to duplicate or consolidate as many instances of the process or some of its services as needed to handle the dynamic of received requests. Indeed, we believe that handling elasticity does not only operate at the process level but it should operate at the level of services too. It is not necessary to duplicate all the services of a considered SBP while the bottleneck comes from some services of the SBP.

2.1 SBP modeling

We model the SBP using Petri nets. Many approaches model SBPs using petri nets, but instead of focusing on the execution model of processes and their services, we focus on the dynamic (evolution) of loads on each basic service participating in the SBP's composition. In the proposed model, each service is represented by a place. The transitions represent calls transfers between services.

Definition 1. *A SBP load model is a petri net $N =< P, T, Pre, Post >$:*

– *P: a set of places (represents the set of services involving in a SBP).*
– *T: a set of transitions (represents the call transfers between services according to the SBP behavioural specification).*
– $Pre : P \times T \rightarrow \{0, 1\}$
– $Post : T \times P \rightarrow \{0, 1\}$

For a place p and a transition t we give the following notations:

- $p^\bullet = \{t \in T | Pre(p,t) = 1\}$. ${}^\bullet p = \{t \in T | Post(t,p) = 1\}$
- $t^\bullet = \{p \in P | Post(t,p) = 1\}$. ${}^\bullet t = \{p \in P | Pre(p,t) = 1\}$

Definition 2. *Let N be a Petri net, we define a net system $S = \langle N, M \rangle$ with $M : P \rightarrow \mathbb{N}$ a marking that associates to each place an amount of tokens.*

The marking of a Petri net represents a distribution of calls over the set of services that compose the SBP. A Petri net system models a particular distribution of calls over the services of a deployed SBP.

Definition 3. *Given a net system $S = \langle N, M \rangle$ we say that a transition t is fireable in the marking M, noted by $M[t\rangle$ iff $\forall p \in^\bullet t : M(p) \geq 1$.*

Definition 4. *The firing of a transition t in marking M changes the marking of the net system to M' s.t. $\forall p : M'(p) = M(p) + (Post(t,p) - Pre(p,t))$, we note the transition by $M[t\rangle M'$.*

The transition firing represents the evolution of the load distribution after calls transfer. The way that calls are transferred between services depends on the behaviour specification (workflow operators) of the SBP.

2.2 Elasticity operations

Place duplication

As noted before places represent services deployed on containers. The marking of a place denote the number of the current instances (or requests) of the service. Each service has a maximal capacity over what the QoS of the service decrease and can leads to the stuck of the container and by the same way the crash of the service. Giving to the container more memory and/or more CPU time will not change the issue of the problem [8]. A solution to this problem is to duplicate the service without changing underlying SBP.

Definition 5. *Let $S = \langle N, M \rangle$ be a net system and let $p \in P$, the duplication of p in S by a new place p^c ($\notin P$), noted as $D(S, p, p^c)$, is a new net system $S' = \langle N', M' \rangle$ s.t*

- $P' = P \cup \{p^c\}$
- $T' = T \cup T''$ *with* $T'' = \{t^c | t \in ({}^\bullet p \cup p^\bullet)\}$
- $Pre' : P' \times T' \rightarrow \{0, 1\}$
- $Post' : T' \times P' \rightarrow \{0, 1\}$
- $M' : P' \rightarrow \mathbb{N}$ *with* $M'(p') = M(p')$ *if* $p' \neq p^c$ *and* 0 *otherwise.*

The Pre' (respectively $Post'$) functions are defined as follow:

$$Pre'(p',t') = \begin{cases} Pre(p',t') & p' \in P \wedge t' \in T \\ Pre(p',t) & t \in T \wedge t' \in (T' \setminus T) \wedge p' \in (P \setminus \{p\}) \\ Pre(p,t) & t \in T \wedge t' \in (T' \setminus T) \wedge p' = p^c \\ 0 & otherwise. \end{cases}$$

$$Post'(t',p') = \begin{cases} Post(t',p') & p' \in P \wedge t' \in T \\ Post(t,p') & t \in T \wedge t' \in (T' \setminus T) \wedge p' \in (P \setminus \{p\}) \\ Post(t,p) & t \in T \wedge t' \in (T' \setminus T) \wedge p' = p^c \\ 0 & otherwise. \end{cases}$$

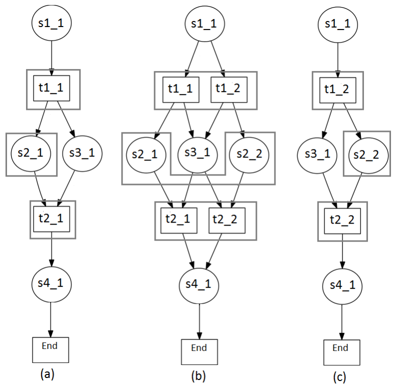

Fig. 1. Example of the elasticity of a SBP

Place consolidation

When a service has few calls, the containers that host its instances use more resources than required for the same QoS. As a dual operator to duplication we define the consolidation or merging operator that removes a copy of a service.

Definition 6. *Let $S = \langle N, M \rangle$ be a net system and let p, p^c be two places in N with $p \neq p^c$, the consolidation of p^c in p, noted as $C(S, p, p^c)$, is a new net system $S' = \langle N', M' \rangle$ s.t*

- *N': is the net N after removing the place p^c and the transitions $(p^c)^{\bullet} \cup {}^{\bullet} p^c$*
- *$M' : P' \rightarrow \mathbb{N}$ with $M'(p) = M(p) + M(p^c)$ and $M'(p') = M(p')$ if $p' \neq p$.*

Example 1. Figure 1-(a) shows an example of nets system that represents an SBP. Figure 1-(b) is the resulted system from the duplication of $s2\_1$ in (a). Figure 1(c) is the consolidation of the place $s2\_1$ in its copy $s2\_2$.

3 A generic framework for SBPs elasticity

Usually in the Cloud, a set of policies is implemented to guarantee some SLA properties to the deployed applications. These policies are implemented in what is usually called controller. In our case, we are interested in elasticity policies of services that compose a SBP. In order to achieve this, we want to develop a controller to provide an optimal ratio QoS and allocated resources of a SBP. The proposed controller (Figure 2)-(a) is able to perform three actions:

- Routing: Routing decision is about the way a load of services is routed over the set of their copies. It determines under which condition a given transition (call transfer) is fired. One can think of routing as a way to define a strategy to control the flow of load according to some rules *e.g.* a call is transferred iff the resulted marking does not violate the capacity of the services.

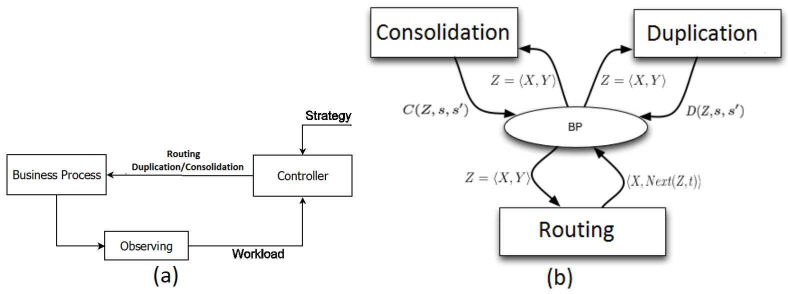
Strategy
Routing
Duplication/Consolidation
Business Process
Controller
Observing
Workload
(a)
Consolidation
Duplication
$Z = \langle X, Y \rangle$
$Z = \langle X, Y \rangle$
$C(Z, s, s')$
$D(Z, s, s')$
BP
$Z = \langle X, Y \rangle$
$\langle X, Next(Z, t) \rangle$
Routing
(b)

Fig. 2. General architecture of the controller

- Duplication: Duplication decision is about the creation of a new copy of a service in order to meet its increased workload.
- Consolidation: Consolidation decision is about the removing of an unnecessary copy of a service in order to meet its workload decrease.

If we consider the three actions that can be performed by an elasticity controller, any combination of conditions associated with a decision of routing, duplication and consolidation is an elasticity strategy. The strategy is responsible of making decisions on the execution of elasticity mechanisms *i.e.* deciding when and how to use these mechanisms. Several strategies can be used to manage the SBP elasticity [4]. The abundance of possible strategies requires evaluating these strategies before implementing them. Our goal here is not to propose an additional elasticity strategy, but a framework, called generic controller that allows the implementation and evaluation of SBPs elasticity strategies.

We model the controller as a high level Petri net (HLPN). The structure of the controller is shown in Figure 2-(b). The controller HLPN contains one place (BP) of type net systems (SBPs). The marking of this place is modified by three transitions that represent the three elasticity mechanisms (Routing, Duplication and Consolidation). Each of these transitions is guarded by a generic condition. Implementing a strategy consists then in instanciating the three generic conditions. The reachability graph resulted from the instanciated controller represents the different evolutions of the SBP according to the strategy. Using HLPN analysis tools, the evaluation of the strategy can be processed by model-checking its reachability graph.

4 Related Work

The elasticity in the Cloud has been studied in the past. Proposed approaches use generally sets of rules to make decisions about the elasticity of the infrastructure. In this kind of approaches, several techniques have been used. In [3] the authors propose to add or remove VMs according to demands. In [5, 1] the authors propose to calculate the optimal number of VMs to be deployed according to variations of demands.

The use of duplication/consolidation mechanisms to provide elasticity have been considered in the area of dynamic service deployment [2, 7]. The proposed mechanisms allow the duplication/consolidation of the entire SBP (and so, of all its services) while the bottleneck may come from some services of the SBP.

At the best of our knowledge the approaches for elasticity are interested in the infrastructure level of cloud environments (IaaS). As stated before, ensuring elasticity at the IaaS level is not sufficient to provide users with elasticity of deployed SBPs. Similarly, ensuring elasticity at the PaaS level is not enough to ensure elasticity of deployed SBPs. We believe that elasticity should be handled and tuned at different levels of cloud environments. The work we present in this paper is novel in the sense that it (1) tackles the problem of elasticity at the SaaS level and (2) proposes a generic framework for evaluating SBPs elasticity.

5 Conclusion

This paper addresses the problem of elasticity of service-based business processes (SBPs) deployed in cloud environments. Unlike existing work, the proposed approach tackles the elasticity at the level of SBPs. To perform elasticity we proposed using Petri nets tow operations: duplication and consolidation. In addition, we have proposed a framework to evaluate SBPs elasticity. As perspectives of this work, we are working on the implementation of the elasticity operations into CloudServ (a PaaS under development within the French FUI CompatibleOne project).

References

1. J. Bi, Z. Zhu, R. Tian, and Q. Wang. Dynamic provisioning modeling for virtualized multi-tier applications in cloud data center. *IEEE International Conference on Cloud Computing*, 0:370–377, 2010.
2. C. Chrysoulas, G. Kostopoulos, E. Haleplidis, R. Haas, S. Denazis, and O. Koufopavlou. A decision making framework for dynamic service deployment.
3. T. N. B. Duong, X. Li, and R. S. M. Goh. A framework for dynamic resource provisioning and adaptation in iaas clouds. *IEEE International Conference on Cloud Computing Technology and Science*, 0:312–319, 2011.
4. H. Ghanbari, B. Simmons, M. Litoiu, and G. Iszlai. Exploring alternative approaches to implement an elasticity policy. In *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing*, CLOUD '11, pages 716–723, 2011.
5. R. Han, L. Guo, Y. Guo, and S. He. A deployment platform for dynamically scaling applications in the cloud. *IEEE International Conference on Cloud Computing Technology and Science*, 0:506–510, 2011.
6. M. Mohamed, S. Yangui, S. Moalla, and S. Tata. Web service micro-container for service-based applications in cloud environments. In *WETICE*, pages 61–66, 2011.
7. J. B. Weissman, S. Kim, and D. England. A framework for dynamic service adaptation in the grid: Next generation software program progress report. *International Parallel and Distributed Processing Symposium*, 2005.
8. S. Yangui, M. Mohamed, S. Tata, and S. Moalla. Scalable service containers. In *CloudCom*, pages 348–356, 2011.