2017 IEEE 24th International Conference on Web Services

Genetic-based Configurable Cloud Resource Allocation in QoS-aware Business Process Development

Emna Hachicha, Karn Yongsiriwit
Telecom SudParis, Université Paris-Saclay
UMR 5157 Samovar, France

Mohamed Sellami
ISEP Paris
LISITE Laboratory, France

Walid Gaaloul
Telecom SudParis, Université Paris-Saclay
UMR 5157 Samovar, France

***Abstract*—Organizations are nowadays more and more adopting Cloud computing to execute their multi-tenant business processes in order to quickly adapt to changes of requirements at lower costs. In such environment, using configurable process models allows for various tenants to share a reference process which can be customized depending on their needs. However, there is a lack of support for cloud-specific resource configuration with considering quality of service requirements. In this paper, we cope with this gap by proposing a genetic-based approach that enables to optimally extract cloud resource configuration allocation w.r.t Cloud resource features (i.e., elasticity and shareability), and process non functional properties (i.e. QoS properties). Keywords. Business process management; configurable resource allocation; Cloud computing; QoS properties**

I. Introduction

In past years, Cloud computing has become an important paradigm for organizations to outsource their service-based business processes (BPs) in a multi-tenant way. This is due to the capacity of the cloud of offering a pool of configurable cloud resources (e.g., networks, and storage) and enabling tenants to use them in an on-demand manner with a minimal management effort [1]. Moreover, organizations are implementing their BPs w.r.t QoS properties to better satisfy Service Level Agreement (SLA). In such environment, using configurable process models permits cloud-based BP providers to deliver flexible and customizable processes that can adapt to different requirements provided by tenants [2,3].

Resource orchestration represents a key challenge that leads to correctly exploit cloud potential [4]. More specifically, we are interested in resource by the process provider, which is a tedious task. This task should be allocation for BPs that consists in selecting the most suitable resources required well managed for ensuring a correct and optimal BP completion.

While optimizing resource allocation has been studied in the last years, the existing proposals [5–10] have not tackled the problem considering both Cloud and QoS perspectives. On the one hand, some works [11–13] consider only cloud resource scheduling and neglect the QoS impact. On the other hand, other researches [14,14–16] consider QoS properties to reduce costs while omitting Cloud features such as elasticity and shareability that should be properly handled to ensure a correct resource allocation.

Therefore, we build upon our previous work [10,17], where the configurable BP has been enriched with configurable resource operators, to reach the next step which is to generate correct and optimal process variants. Since this phase is complicated task, we choose to use genetic algorithms as a suitable technique to deal with such combinatorial problem. Hence, we propose a novel approach that, using a genetic algorithm, aims at selecting the optimal cloud resource configuration that best fits to the tenant requirements in terms of elasticity and shareability and optimizes the QoS properties.

The rest of the paper is organized as follows. Section II describes background for this work. Section III describes a cloud resource allocation optimization model and identifies its requirements. Section IV formulates the problem and present our genetic-based approach. Section V describes the experimentation in order to validate our approach. Finally section VI concludes the paper and provides some future work.

II. Preliminaries

We briefly present in this section a general view about our previous approach [17] for supporting configurable resource allocation in configurable BPs on which this paper is based.

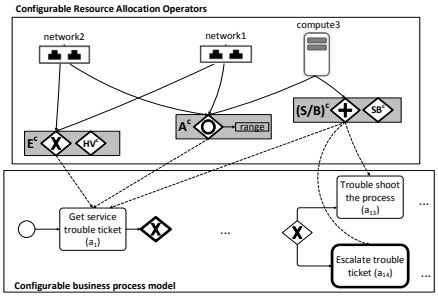

Fig. 1: Configurable resource operators

A configurable process model includes configurable elements whose configuration decisions are made at design-time [3]. While concepts of the control-flow perspective have been widely addressed [3,18,19], there is a lack of support for the resource perspective configuration.

Therefore, we have extended configurable BPs with three configurable resource operators to allow tenants to customize their needed resources w.r.t cloud features: elasticity and shareability. These resource operators are: (*i*) configurable resource Assignment operator (A^C): allows the modeling of a variable number of resources allocated to a specific activity, (*ii*) configurable resource Elasticity operator (E^C): allows to model the variability at the elasticity level that an organization may require regarding the anticipation of its activities workload, and (*iii*) configurable resource Sharing/Batching operator (S/B^C): permits to model the variability regarding

978-1-5386-0752-7/17 $31.00 © 2017 IEEE
DOI 10.1109/ICWS.2017.101

the shareability of resources. Fig. 1 represents a configurable process model, at both control-flow and resource levels. It is modeled using the appropriate icons of the proposed resource operators. Hence, to allocate the needed cloud resources to activities, Orange affiliates configure these operators.

Our work is motivated by a real business case study of the French Orange operator which is an industrial partner. Our example refers to two process examples that are variants of a configurable service supervision business process which is shared between Orange affiliates. This configurable process presents how a customer's complaint is addressed after a service quality drop. Details are given on line [1]. Once an affiliate decides to use this process, he can fine-tune the process according to his requirements in terms of resources. Thus, for each activity in the process variant, the affiliate expresses its needs in terms of cloud resource allocation and according to them he configures the configurable resource operators (i.e., A^c, E^c, and S/B^c).

III. Cloud-specific Resource Properties (CRP) Problem

In this section, we define cloud-specific resource properties (CRP) to describe our optimization problem. In a cloud context, resources are characterized by their dynamic and flexible behavior. We are interested in two major cloud features: elasticity and shareability. To ensure a correct resource configuration, we identify process variants that fit to a tenant's requirements in terms of elasticity and shareability referred to as CRP requirements. To do so, we formally define such requirements (Definition 2) based on the new definition of configurable BP (Definition 1).

Definition 1 (Configurable BP): A configurable BP denoted as CBP is a tuple $(id, A, E, CR_p, RO_p, I, K)$ where:

- id is its unique identifier;
- A is the set of activities;
- $E \subseteq A \times A$ is the set of edges connecting two activities;
- CR_p is the set of cloud resources;
- RO_p is the set of configurable resource operators;
- $I \subseteq CR_p \times RO_p$ is the set of edges connecting cloud resources with configurable operators;
- $K \subseteq RO_p \times A$ is the set of edges connecting activities with configurable cloud resource operators;

Next, we specify the CRP requirements that are expressed by process tenant. They are defined per activity and refer to the accepted states of the consumed cloud resources (Definition 2).

Definition 2 (CRP requirements): The CRP requirement is a tuple $(id_A, RRT, AES, ASS, Cr)$ where:

- id_A is the identifier of the activity;
- RRT (Requirement Resource Type) is the resource type where $RRT = \{compute, network, storage\}$;
- $AES \in CR.RES$ (Acceptable Elastic State) is the required state regarding the elasticity;
- $ASS \in CR.RSS$ (Acceptable Sharability State) is the required state regarding the shareability;
- Cr is the requested capacity.

[1]http://www-inf.it-sudparis.eu/SIMBAD/tools/EE-CRA/

The tenant defines its CRP requirements per activity. Each activity may have three requirements per resource type i.e., CRP^c for compute, CRP^s for storage, and CRP^n for network resources. For instance, the tenant requires for a_1 vertical elastic and shared compute resources, and vertical elastic and not shared network resources.

Algorithm 1 Generation of resource process variant

Require: CBP, CRP
Ensure: RAV
1: $O \leftarrow GetresourceOperators(RO)$ ▷ Extract the configurable resource operators.
2: **for all** o **in** O **do** ▷ Iterate for each operator in O.
3: $OR \leftarrow GetAllocatedResourceSet(o)$ ▷ Get the allocated resources
4: **if** $CanConfiguredTo(o, \wedge)$ **then** ▷ Check if the operator can be configured to $\wedge$.
5: $R \leftarrow NewResourceSet()$ ▷ Initial an empty resource set.
6: **for each** r **in** OR **add** r **to** R ▷ Add all allocated resources to R.
7: **add** R **to** RAV ▷ Add a set of allocated resources to RAV.
8: **end if**
9: **if** $CanConfiguredTo(o, \times)$ **then** ▷ Check the configuration to $\times$.
10: **for each** r **in** OR **add** r **to** RAV ▷ Add individual resource to RAV.
11: **end if**
12: **if** $CanConfiguredTo(o, \vee)$ **then** ▷ Check the configuration to $\vee$.
13: $i = 0$
14: **while** $i < SizeOf(OR)$ **do**
15: $i = i + 1$
16: $R \leftarrow combine(OR, i)$ ▷ $combine$ is a function that returns a set of allocated resources following the logic of $\vee$ operator
17: **add** R **to** RAV
18: **end while**
19: **end if**
20: **end for**

Algorithm 2 Acceptable resource process variant

Require: RAV
Ensure: RAV
1: **for all** rav **in** RAV **do** ▷ Iterate for each resource allocation variant in RAV.
2: **for all** r **in** rav **do** ▷ Iterate for each resource in rav.
3: $RES_r \leftarrow GetRES(r)$ ▷ Get resource RES.
4: $RSS_r \leftarrow GetRSS(r)$ ▷ Get resource RSS.
5: $O_r \leftarrow GetAssignedOperators(r)$ ▷ Get a set of operators assigned with resource r.
6: **for all** o **in** O_r **do** ▷ Iterate for each operator in O_r.
7: $a \leftarrow GetAssignedActivity(o)$ ▷ Get an activity assigned with operator o
8: $AES_a \leftarrow GetAES(a)$ ▷ Get activity AES.
9: $ASS_a \leftarrow GetASS(a)$ ▷ Get activity ASS.
10: **if** $RES_r \neq AES_a \vee RSS_r \neq ASS_a$ **then** ▷ Verify whether or not CRP properties (RES_r, RSS_r) of a resource is compliance with CRP requirements (AES_a, ASS_a) of its assigned activity.
11: **remove** rav **from** RAV ▷ Remove the non-compliace resource variant rav.
12: **break**
13: **end if**
14: **end for**
15: **end for**
16: **end for**
17: **return** RAV ▷ Return accepted resource allocation variant.

By configuring the resource operators, we may obtain several possible variants for resource allocation for each activity. These variants may fit to the CRP requirements (i.e., acceptable) or not (i.e., not acceptable). Since our aim is to look for the best variant, we only select at this stage the acceptable variants. Algorithm 1 allows to generate all the possible resource variants from a configurable BP according to the behavior of the resource operators. Then, algorithm 2 aims at selecting, from these variants, the acceptable ones based on CRP requirements. More details can be found on line [2].

[2]http://www-inf.it-sudparis.eu/SIMBAD/tools/EE-CRA/

IV. Problem Formalization using Genetic-based Approach

In this section, we use a genetic-based algorithm to solve the cloud resource allocation problem in configurable BPs w.r.t QoS properties. Genetic algorithm [20] is a powerful tool to deal with such combinatorial problems. To this end, we firstly encode our problem with a genome depicted at the left in Fig. 2. The genome represents an individual in the form of an integer array where the number of its elements is equal to the number of activities that consume at least one resource. Each element refers to an activity a_i that contains an index I_{ai} to the array of its cloud resource configurations $R^c a_i$.

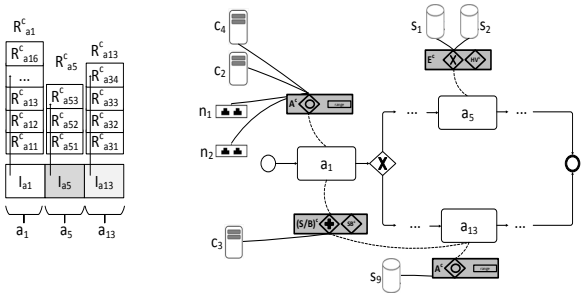

Fig. 2: Configurable business process example

Then, we apply the standard two-point crossover operator to generate the population of genome arrays. Such operator randomly selects an activity a_i (i.e., an index in the genome array) and replaces the index of resource configuration I_{ai} corresponding to its cloud resource configuration array $R^c a_i$ by another configuration. In our example, the maximum number of resource configurations for one activity is 6. Thus, three bits are enough to express every cloud resource configuration. One of the possible generated population is $I_a 1 = 010$, $I_{a5} = 000$, and $I_{a13} = 001$ which represents cloud resource configurations $R^c_{a13} = \{c_2, n_1\}$ of the activity a_1, $R^c_{a51} = \{s_1\}$ of a_5, and $R^c_{a32} = \{c_2\}$ of a_{13}, respectively. For example, the activity a_1 (see Fig. 2) can be allocated to the following resource configurations $R^c a_1 = \{\{c_2, c_3\}, \{c_2, n_1\}, \{c_1, n_2\}, \{c_2, n_1\}, \{c_2, n_2\}, \{c_1, c_2, n_1\}, etc\}$.

Fitness Function

To define a fitness function (*FF*) for our genetic algorithm, we specify a set of QoS properties: cost (*CT*), response time (*RT*) and availability (*AT*). The according constraints denote that the *CT* and *RT* of a given BP should be less than maximal thresholds $cmax$ and $rmax$, and *AT* should be greater than minimal threshold referred to as $amax$ (Equation1).

$$\begin{aligned} CT(BP) &\geq cmax \\ RT(BP) &\geq rmax \\ AT(BP) &\leq amax \end{aligned} \tag{1}$$

Thereafter, our (*FF*) is defined as follows to enable the assessment of these QoS properties for a given BP. The higher FF the better the solution is.

$$\begin{aligned} FF(BP) = \alpha_1 * (cmax - CT(BP)) + \\ \alpha_2 * (rmax - RT(BP)) + \alpha_3 * (AT(BP) - amax) \end{aligned} \tag{2}$$

Where α_1, α_2 and α_3 are defined as the weighting factors that belong to the interval [0,1]. They represent the user preferences w.r.t the QoS properties ($\alpha_1 + \alpha_2 + \alpha_3 = 1$). It is necessary to define a stop condition for a genetic algorithm. Hence, we consider the following condition:

$$FF(g) \geq \delta \tag{3}$$

where δ is the weighting quality factor which is determined by experimentations.

Algorithm 3 Optimal resource allocation variant construction

Require: RAV, δ
Ensure: bit

1: $bit \leftarrow GenerateAnInitialBit(RAV)$ ▷ Generate an initial bit representation
2: $bit \leftarrow$ GENOME(bit, δ) ▷ Start genetic algorithm.
3: **return** bit ▷ Return an optimal cloud resource allocation variant.
4: **procedure** GENOME(bit, δ) ▷ Check if the stop condition is respected
5: **while** $FF(bit) < \delta$ **do**
6: $bit \leftarrow Crossover()$ ▷ Apply the crossover operation to the bit
7: $bit \leftarrow Mutation()$ ▷ Apply the mutation operation to the bit
8: GENOME(bit, δ) ▷ Recursive call for the next population.
9: **end while**
10: **return** bit ▷ Return an optimal cloud resource allocation variant.
11: **end procedure**

Algorithm 3 shows the steps of our genetic-based algorithm. Firstly, we generate an initial bit representation bit based on the maximum number of cloud resources in RAV (line 1). Then, we execute the genetic algorithm function (line 3). In our genetic algorithm (line 4-11), we compute the fitness function FF from bit (line 5). If the result is less then the δ value, we apply the crossover and mutation operators for obtaining new population taking into account previous populations (line 6-7). Thus, we recursively call the genetic algorithm function for the next population (line 8). If the result satisfy the δ value, the algorithm return the resource variant of the bit representation bit (line 9).

V. Experimentation

To validate our approach, we conduct experiments using a real dataset of BPs from Orange labs. The experiments were performed on a computer with Intel, 3.0 GHz, 8 Go of RAM, Windows 7 Entreprise edition. Our genetic-based approach was implemented in Java using a library[3].

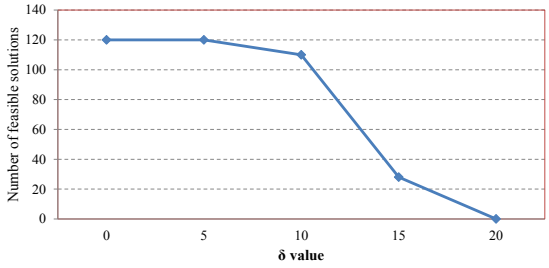

Fig. 3: The impact of the of δ on the quality of the solution

First, we assess the quality of our genetic algorithm by varying the factor δ from 0 to 20. The weight of the fitness function, α_1, α_2, and α_3 are 0.33 for simplicity sake. Fig. 3 shows the impact of δ on the number of feasible solutions.

[3]http://jenetics.io/

The results show that the number of feasible variants decreases when δ increases. From the δ value is 20, there is a risk to reject feasible resource allocation variant. To conclude, the quality of the solution is influenced by δ.

Second, we present a comparison with LIP which is a broadly used mathematical optimization program [21]. We study the growth of the computation time comparing to the number of consumed resources per process variant. Concretely, we generate 5 variants from our orange configurable BP having from 4 to 20 allocated resources per activity.

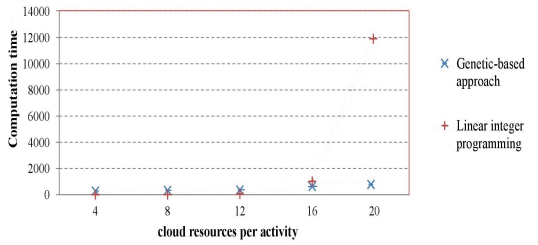

Fig. 4: Comparing Genetic-based approach with LIP

Fig. 4 depicts the results of our comparison. It shows that when the number of assigned cloud resources per activity is small (4-12), LIP outperforms our genetic-based approach. However, when the number of cloud resources becomes higher (16-20), our approach is able to keep its timing performance almost constant while LIP performs an important exponential growth. Hence, the LIP algorithm can be efficient only when the problem is not complex. We can conclude that our genetic-based approach should be preferred than LIP in the scenario where we have a large number of cloud resources that can be assigned to process activities. We refer herein to the case of large-scale service-based business processes.

VI. Related Work

The general benefits of the blending of Cloud and BPM have been stressed by different authors [22,23]. Thus, organizations have lately been investigated in the deployment of service-based business processes in Cloud environments. Different works on Cloud resource orchestration in such context have been proposed [8,11,12,24]. However, some works [8,24] have mainly focused on the management of human resources while other researches [4,12] working on Cloud resource management have neglected the variability in terms of Cloud resource allocation.

From QoS vision, organizations are constantly seeking to improve the non functional properties of their BPs in many aspects [14–16]. There are domain-dependent aspects (e.g., resolution for image processing BPs) and domain-independent e.g., response time, or cost. Despite the great support of QoS properties, they lack optimize at the same time Cloud resource allocation w.r.t requirements that BP providers define regarding the elasticity ans shareability properties and improving QoS properties.

VII. Conclusion

We proposed an approach that achieves a QoS-aware configurable resource allocation in cloud-based BPs. It guarantees the selection of the process variant which best fits to the process tenant requirements, and optimizes the QoS properties. This approach has been validated through experiments that we have conducted.

Some threats to validity exist in our work. First, our approach requests a larger dataset to further evaluate its effectiveness. Second, there are other metrics, which are of a high importance, that we do not take into account such as environmental metrics included in Green Computing. Third, our approach can be compared to other algorithms, in addition to LIP, such as the Ant Colony Optimization.

References

[1] P. M. Mell and T. Grance, "Sp 800-145. the nist definition of cloud computing," tech. rep., Gaithersburg, MD, United States, 2011.

[2] M. Rosemann and W. M. P. van der Aalst, "A configurable reference modelling language," *Inf. Syst.*, vol. 32, pp. 1–23, Mar. 2007.

[3] W. M. P. van der Aalst, "Business process configuration in the cloud: How to support and analyze multi-tenant processes?," in *ECOWS 2011, Switzerland, September 14-16*, pp. 3–10, 2011.

[4] R. Ranjan and et al., "Cloud resource orchestration programming: Overview, issues, and directions," *IEEE IC*, vol. 19, pp. 46–56, 2015.

[5] M. L. Rosa and et al., "Configurable multi-perspective business process models," *Inf. Syst.*, vol. 36, no. 2, pp. 313–340, 2011.

[6] A. Kumar and W. Yao, "Design and management of flexible process variants using templates and rules," *CiI*, pp. 112–130, 2012.

[7] A. Hallerbach, T. Bauer, and M. Reichert, "Capturing variability in business process models: the provop approach," *Journal of Software Maintenance*, vol. 22, no. 6-7, pp. 519–546, 2010.

[8] C. Cabanillas and et al., "Ralph: A graphical notation for resource assignments in business processes," in *CAiSE, Sweden*, pp. 53–68, 2015.

[9] A. Nowak, T. Binz, C. Fehling, O. Kopp, F. Leymann, and S. Wagner, "Pattern-driven green adaptation of process-based applications and their runtime infrastructure," *Computing*, vol. 94, no. 6, pp. 463–487, 2012.

[10] E. Hachicha, K. Yongsiriwit, and W. Gaaloul, "Energy efficient configurable resource allocation in cloud-based business processes (short paper)," in *OTM 2016 Conferences, Rhodes, Greece*, pp. 437–444, 2016.

[11] P. H. et al., "Cost-efficient scheduling of elastic processes in hybrid clouds," in *CLOUD, NY, USA*, pp. 17–24, 2015.

[12] P. Hoenisch and al., "Self-adaptive resource allocation for elastic process execution," in *Cloud, CA, USA*, pp. 220–227, 2013.

[13] W. S. et al., "Adaptive resource provisioning for the cloud using online bin packing," *IEEE Trans. Computers*, no. 11, pp. 2647–2660, 2014.

[14] W. Z. Low and et al., "Revising history for cost-informed process improvement," *Computing*, vol. 98, no. 9, pp. 895–921, 2016.

[15] W. Z. Low and et al., "Perturbing event logs to identify cost reduction opportunities: A genetic algorithm-based approach," in *IEEE CEC 2014, Beijing, China, July 6-11*, pp. 2428–2435, 2014.

[16] M. T. Wynn and et al., "Cost-informed operational process support," in *ER 2013, Hong-Kong, China, November 11-13*, pp. 174–181, 2013.

[17] E. Hachicha, N. Assy, W. Gaaloul, and J. Mendling, "A configurable resource allocation for multi-tenant process development in the cloud," in *CAiSE 2016, Ljubljana, Slovenia, June 13-17*, pp. 558–574, 2016.

[18] M. L. Rosa, W. M. van der Aalst, M. Dumas, and F. P. Milani, "Business process variability modeling : A survey," 2013.

[19] R. Mietzner and F. Leymann, "Generation of BPEL customization processes for saas applications from variability descriptors," in *SCC 2008, 8-11 July, Hawaii, USA*, pp. 359–366, 2008.

[20] S. N. Sivanandam and S. N. Deepa, *Introduction to genetic algorithms*. Springer, 2008.

[21] S. Talluri and R. C. Baker, "A multi-phase mathematical programming approach for effective supply chain design," *European Journal of Operational Research*, vol. 141, no. 3, pp. 544–558, 2002.

[22] E. F. Duipmans and et al., "Towards a BPM cloud architecture with data and activity distribution," in *EDOC Workshops, Beijing, China, September 10-14*, pp. 165–171, 2012.

[23] M. Wang, K. Y. Bandara, and C. Pahl, "Process as a service," in *IEEE IC on Services Computing, SCC,Miami USA 2010*, pp. 578–585, 2010.

[24] C. Cabanillas and et al., "Towards process-aware cross-organizational human resource management," in *EMMSAD, Greece*, pp. 79–93, 2014.