International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622
NATIONAL CONFERENCE on Developments, Advances & Trends in Engineering Sciences
(NCDATES- 09th & 10th January 2015)

RESEARCH ARTICLE OPEN ACCESS

Cost-Based Optimization of Service Compositions

*Turkar Vipul Prakash
M.Tech schlor,
CMR Engineering College,Hyderabad,India

Mrs.Rajeswari **,Asst.Professor
CMR Engineering College,Hydearbad,India
E-Mail:rajeswariimaturi@gmail.com

Abstract—
For providers of composite services, preventing cases of SLA violations is crucial. Previous work has established runtime adaptation of compositions as a promising tool to achieve SLA conformance. However, to get a realistic and complete view of the decision process of service providers, the costs of adaptation need to be taken into account. In this paper, we formalize the problem of finding the optimal set of adaptations, which minimizes the total costs arising from SLA violations and the adaptations to prevent them. We present possible algorithms to solve this complex optimization problem, and detail an end-to-end system based on our earlier work on the PREvent (prediction and prevention based on event monitoring) framework, which clearly indicates the usefulness of our model. We discuss experimental results that show how the application of our approach leads to reduced costs for the service provider, and explain the circumstances in which different algorithms lead to more or less satisfactory results.

Index Terms—Service composition, service-level agreements, adaptation, optimization

I. INTRODUCTION

SERVICE BASED applications have seen tremendous research activity in the last years, with many important results being generated around the world [1]. This global interest is justified by the ever increasing services industry, which is still only starting to explore the potential that new paradigms like Everything-as-a-Service (XaaS) or cloud computing provide [2]. However, to fully realize this potential,research and industry alike need to focus more strongly on nonfunctional properties and quality issue of services (generally referred to as QoS). In the business world, QoS promises are typically defined within legally binding service-level agreements (SLAs) between clients and service providers, represented, e.g., using WSLA [3]. SLAs contain service-level objectives (SLOs), i.e., concrete numerical QoS objectives, which the service needs to fulfill. If SLOs are violated, agreed upon monetary consequences go into effect. For this reason, providers generally have a strong interest in monitoring SLAs and preventing viola- tions, either by using post mortem analysis and optimiza- tion [4], [5], or by runtime prediction of performance problems [6], [7]. We argue that the latter is more powerful, allowing to prevent violations before they have happened by timely application of runtime adaptation actions [8], [9], [10]. However, preventing SLA violations is, in general, not for free. For instance, some alternative services usable in a composition may provide faster response times (thereby improving the end-to-end runtime of the composite service, SLOs), but those services are often more expensive than slower ones. Therefore, there is an apparent tradeoff between preventing SLA violations and the inherent costs of doing so. We argue that this tradeoff is currently not covered sufficiently in research. Instead, researchers as- sume that the ultimate goal of service providers is to minimize SLA violations, completely ignoring the often significant costs of doing so (e.g., [9], [10]). In this paper, we contribute to the state of the art by formalizing this tradeoff as an optimization problem, with the goal of minimizing the total costs (of violations and applied adaptations) for the service provider. We argue that this formulation better captures the real goals of service providers. Additionally, we present possible algorithms to solve this optimization problem efficiently enough to be applied at composition runtime. We evaluate these algo- rithms within our PREVENT (prediction and prevention based on event monitoring) framework [8]. The remainder of this paper is structured as follows: In Section 2, we motivate our work and present an illustrative example, which will guide us through the rest of the paper. Following in Section 3, we present our earlier work on prevention of SLA violations. In Section 4, we formalize the problem of cost-based optimization of service compositions. We explain possible algorithms to solve this problem efficiently in Section 5, which are experimentally evaluated in Section 6. Finally, we compare our work with the most important related scientific approaches in Section 7, and conclude the paper in Section 8.

II. EXISTING SYSTEM:

In this paper, we contribute to the state of the art by formalizing this tradeoff as an optimization problem, with the goal of minimizing the total costs (of violations and applied adaptations) for the service provider. We argue that this formulation better captures the real goals of service providers.

III. DISADVANTAGES:

There is an apparent tradeoff between preventing SLA violations and the inherent costs of doing so. We argue that this tradeoff is currently not covered sufficiently in research. Instead, researchers assume that the ultimate goal of service providers is to minimize SLA violations, completely ignoring the oftensignificant costs of doing so.

IV. PROPOSED SYSTEM:

we present possible algorithms to solve this optimization problem efficiently enough to be applied at composition runtime. We evaluate these algorithms within our PREVENT (prediction and prevention based on event monitoring) framework

3.1ALGORITHMS:

We will now discuss different approaches for finding solutions to this problem. These algorithms are implemented in the Cost-Based Optimizer component. Optimization is always triggered by a predicted violation of at least one SLO, and receives as input a list of monitored facts and estimates of the current instance.

3.2 Branch-and-Bound

Branch-and-bound is a very general deterministic algorithm for solving optimization problems. The high-level idea of this approach is to enumerate the solution space in a "smart" way so that at least some suboptimal solutions can be identified and discarded prematurely, i.e., before they have been fully constructed and evaluated. We use a binary encoding to represent solutions, i.e., every solution is represented as a binary vector, and an adaptation action with index j is applied iff the solution vector is 1 at index j. For example, the solution vector 00110100 encodes that the third, fourth, and sixth adaptation action should be applied. Evidently, 2jAj different solutions exist for each optimization problem, where jAj is the number of possible adaptation actions (but not all combinations need to be legal). For solutions that are still being constructed we allow a third symbol, "_", representing an action that is still undecided (alive). We refer to solutions, which contain at least one alive action, as partial, and solutions, which do not contain any alive actions, as complete. Therefore, the vector 001101 _ 0 is a partial solution, where the last-but-one action is alive. We describe our general Branch-and-Bound algorithm is easy to understand. What is the most important is the implementation of Line 13, the rules for pruning the search tree (i.e., for prematurely discarding solutions). In our Branch-and-Bound approach, we prune a partial solution in two cases: 1) if the partial solution already contains at least one conflict, or 2) if the partial solution already prevents all SLA violations (the penalty function ps is 0 for all s 2 S) without applying any more actions. Case 1 is trivial, because the target function value for all solutions in such a subtree will always be 1. Case 2 lends itself to more discussion. Remember the assumption that every action has nonnegative costs, and that we described SLA penalty functions as nonnegative functions. Therefore, we can assure that for any solution where all penalty functions are 0, the additional application of more actions can never improve the target function valueremaining solution subtree can be pruned. In Listing 6, we simply iterated over all actions in the order they appeared in the solution vector (in every step, we always just investigate the next action, see Lines 18 and 22). In general, this approach is suboptimal. Even though the order in which we investigate actions has no impact on the quality of our solution (the algorithm is deterministic, i.e., we will always find the global optimum eventually), the order may have an impact on the number of solutions we are able to prune. Assume the Fig. 6. Branch-and-Bound algorithm. Fig. 7. Pruning of solution trees. following simple scenario: There is only one SLO and three possible adaptations. Only adaptation 3 is able to preventthe violation of the SLO. Actions 1 and 2 have costs but no relevant influence. There are no conflicts between actions. Hence, the optimal solution vector is 001. In Fig. 7a, we strictly followed the algorithm in Listing 6 and investigated the actions in the order they appear in the solution vector. Since the only "useful" action is investigated last, we extend the whole solution tree without any pruning (the worst case, equivalent to full enumeration). Now, in Fig. 7b, we investigate the actions in reverse order (from back to front). Now, the "useful" action is investigated first, and a large part of this solution tree can be pruned according to pruning Case 2. Therefore, we can conclude that it is beneficial to investigate actions in a specific order that maximizes the number of solutions that can be pruned. We specify two possible criteria for this ordering: 1) the impact of an action on the SLOs (actions with higher total impact should be investigated first), and 2) the utility of an action (actions with higher utility should be investigated first. Based on the set of historical process instances that we have already used to train the Violation Predictors, we can calculate an estimation of impact and utility of each action as follows: We define the set of available historical process instances as H ¼ fh1; h2; . . . hqg, with H I. We refer to the number

of historical instances as q ¼ jHj. Now, we are able to calculate an estimation of the overall impact of an adaptation action a on a SLO s as _a;s (Equation 8). Simply put, the impact is the arithmetic mean of the difference between SLO value with and without applying the adaptation to each historical instance. X msðhÞ À msðh 1 fagÞ the algorithm increases exponentially with the number of available actions. Even though we can reduce the runtime using impact- or utility-based sorting of actions, the complexity still remains exponential. Hence, there is an evident need to find strong heuristics, i.e., nondeterministic algorithms that find "good" (even if not necessarily optimal) solutions in polynomial time.

A simple heuristic that is often used to very good ends is

Local Search. Local Search is a metaheuristic, i.e., final solutions are constructed by iteratively improving a start solution. The general idea is that in each iteration the algorithm searches a specified neighborhood for better solutions than the current one. If at least one such solution is found, the algorithm progresses to the next iteration with one of the better solutions (typically, the best one in the neighborhood, equivalent to steepest descent). If no better solution can be found in the neighborhood, the algorithm has converged to a local optimum and is terminated. Usually, this algorithm is repeated multiple times with different starting solutions (because different starting

3.3 Local Search

While the Branch-and-Bound algorithm discussed above has the advantage of always finding the optimal set of actions for any composition instance, the execution time of the local search algorithm as discussed above to each solution in the generation, basically reducing the population to a set of locally optimal solutions. Second, we remove the mutation operator from the algorithm (technically speaking, we set the mutation probability parameter to 0). The main reason is that given that all solutions in the population are already locally optimal, randomly mutating one bit in a solution can only lead to a worse solution. In theory, it is possible that multiple bits in a single solution are mutated at the same time, and that these mutations lead to an improvement, but this corner case is very unlikely in practice. Furthermore, the main motivation for having mutation in the first place was that it is the only way of introducing new actions in the canonical GA. This is no longer the case, because local search can do the same thing. the algorithm increases exponentially with the number of available actions. Even though we can reduce the runtime using impact- or utility-based sorting of actions, the complexity still remains exponential. Hence, there is an evident need to find strong heuristics, i.e., nondeterministic algorithms that find "good" (even if not necessarily optimal) solutions in polynomial time.

3.4 Genetic Algorithm

As an alternative to locality-based heuristics (local search, GRASP) we also present a solution based on the concept of evolutionary computation. More precisely, we use genetic algorithms (GAs) [23] as a more complex, but potentially also more powerful heuristic to generate good solutions to the cost-based optimization problem. The overall idea of GA is to mimic the processes of evolution in biology, specifically natural selection of the fittest individuals, crossover, and mutation. Therefore, in GA, we prefer to work on a population of solutions instead of a single one. We use the term "fit" to describe solutions with a good (low) target function value. First, we generate a random start population. For this, we use the same primitive construction scheme as discussed above for local search: We randomly apply m actions in every solution. Every following iteration of the algorithm (referred to as generations) essentially follows a three-step pattern.

```
# name: grasp_init

# input: number of start solutions n,

RCS max size r

# output: set of start solutions



Grasp_init(n,r);

G={ }//empty set of start solutions

Repeat n times

Pa=empty_partial_solution

While(vc(pa)>0);

rcs=construct_rcs(pa,r)
```

V. Conclusion

For providers of composite web services, it is essential to be able to minimize cases of SLA violations. One possible route to achieve this is to predict at runtime, which instances are in danger of violating SLAs, and to apply various adaptation actions to these instances only. However, it is not

trivial to identify which adaptations are the most cost-effective way to prevent any violation, or if it is at all possible to prevent a violation in a cost-effective way. In this paper, we have modeled this problem as a one-dimensional discrete optimization problem. Furthermore, we have presented both, deterministic and heuristic solution algorithms. We have evaluated these algorithms based on a manufacturing case study and have shown which types of algorithms are better suited for which scenarios.

References:

[1] M.P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-Oriented Computing: State of the Art and Research Challenges," Computer, vol. 40, no. 11, pp. 38-45, Nov. 2007.

[2] A. Lenk, M. Klems, J. Nimis, S. Tai, and T. Sandholm, "What's Inside the Cloud? An Architectural Map of the Cloud Landscape," Proc. ICSE Workshop Software Eng. Challenges of Cloud Computing (CLOUD '09) pp. 23-31, 2009.

[3] A. Dan, D. Davis, R. Kearney, A. Keller, R. King, D. Kuebler, H. Ludwig, M. Polan, M. Spreitzer, and A. Youssef, "Web Services on Demand: WSLA-Driven Automated Management," IBM Systems J., vol. 43, no. 1, pp. 136-158, Jan. 2004.

[4] L. Bodenstaff, A. Wombacher, M. Reichert, and M.C. Jaeger, "Analyzing Impact Factors on Composite Services," Proc. IEEE Int'l Conf. Services Computing (SCC '09), pp. 218-226, 2009.

[5] B. Wetzstein, P. Leitner, F. Rosenberg, S. Dustdar, and F. Leymann, "Identifying Influential Factors of Business Process Performance Using Dependency Analysis," Enterprise Information Systems, vol. 4, no. 3, pp. 1-8, July 2010.

[6] P. Leitner, B. Wetzstein, F. Rosenberg, A. Michlmayr, S. Dustdar, and F. Leymann, "Runtime Prediction of Service Level Agreement Violations for Composite Services," Proc. Third Workshop Non-Functional Properties and SLA Management in Service-Oriented Computing (NFPSLAM-SOC '09), pp. 176-186, 2009.

[7] L. Zeng, C. Lingenfelder, H. Lei, and H. Chang, "Event-Driven Quality of Service Prediction," Proc. Sixth Int'l Conf. Service-Oriented Computing (ICSOC '08). pp. 147-161, 2008.

[8] P. Leitner, A. Michlmayr, F. Rosenberg, and S. Dustdar, "Monitoring, Prediction and Prevention of SLA Violations in Composite Services," Proc. IEEE Int'l Conf. Web Services (ICWS '10), pp. 369-376, 2010.

[9] S. Haykin, Neural Networks and Learning Machines: A Comprehensive Foundation, third ed. Prentice Hall, 2008.

[10] J.R. Quinlan, "Induction of Decision Trees," Machine Learning, vol. 1, pp. 81-106, Mar. 1986.

[11] D. Ivanovic, M. Carro, and M. Hermenegildo, "Towards Data- Aware QoS-Driven Adaptation for Service Orchestrations," Proc. IEEE Int'l Conf. Web Services (ICWS '10), pp. 107-114, 2010.

[12] L. Juszczyk and S. Dustdar, "Script-Based Generation of Dynamic Testbeds for SOA," Proc. IEEE Int'l Conf. Web Services (ICWS '10), pp. 195-202, 2010.

[13] M.C. Jaeger, G. Rojec-Goldmann, and G. Muhl, "QoS Aggregation for Web Service Composition Using Workflow Patterns," Proc. Eighth Int'l Enterprise Distributed Object Computing Conference (EDOC '04), pp. 149-159, 2004.

[14] L. Xu and B. Jennings, "A Cost-Minimizing Service Composition Selection Algorithm Supporting Time-Sensitive Discounts," Proc. IEEE Int'l Conf. Services Computing (SCC '10), pp. 402-408, 2010.

[15] T. Feo and M. Resende, "Greedy Randomized Adaptive Search Procedures," J. Global Optimization, vol. 6, pp. 109-133, 1995.

[16] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Professional, 1989.