A Control-Theoretic Approach to Auto-Tuning Dynamic Analysis for Distributed Services

Chandan Dhal
Washington State University, USA
chandan.dhal@wsu.edu

Xiaoqin Fu
Washington State University, USA
xiaoqin.fu@wsu.edu

Haipeng Cai
Washington State University, USA
haipeng.cai@wsu.edu

***Abstract*—Traditional dynamic dependence analysis approaches have limited utilities for continuously running distributed systems (i.e., *distributed services*) because of their low cost-effectiveness. A recent technique, SEADS, was developed to improve the cost-effectiveness by adjusting analysis configurations on the fly using a general Q-learning algorithm. However, SEADS is unable to utilize the user budget as far as needed for pushing up precision. To overcome this problem, we propose CADAS, an adaptive dynamic dependency analysis framework for distributed services. To realize the adaptation, we are exploring a control-theoretical method which uses a feedback mechanism to predict optimal analysis configurations. Then, we evaluated CADAS against six real-world Java distributed services. We compared CADAS against SEADS as the baseline and show that CADAS outperforms the baseline in both precision and budget utilization. Our results suggest a new door opening for future research on adaptive dynamic program analysis.**

***Index Terms*—Distributed system, dynamic analysis, control theory, dependence analysis, cost-effectiveness, auto-tuning**

I. INTRODUCTION

Continuously running distributed systems (i.e., *distributed services*) play crucial roles in our society and daily lives. Thus, the quality (e.g., maintainability, stability, and security) of distributed services is important. Dependence modeling and analysis support different code-based quality-assurance tools, such as system measurement [1] and security defense [2]. Different from static approaches, a dynamic dependence analysis approach focuses on specific program executions; thus, it generally has higher effectiveness (e.g., precision) in nature.

However, developing a practical dynamic dependence analysis often faces scalability and cost-effectiveness challenges [3], even for centralized programs [4], [5]. These challenges become greater for most real-world distributed services because of their typically great complexities and large scales [6]. Other factors (e.g., the non-determinism of long-term even infinite executions and uncertainties in runtime environments of distributed services) further aggravate such challenges [7].

In this context, we recently developed SEADS [8], a framework that overcomes the above challenges by adjusting analysis configurations at runtime according to the dependence computation time costs of specific configurations through a general Q-learning (QL) algorithm. With the better cost-effectiveness it achieved, SEADS made an important first step in scaling dynamic dependence analysis to large distributed services. However, SEADS suffers low precision and then low cost-effectiveness for not being able to utilize as much of the user budget as possible. The reason is that SEADS uses a general QL reward function for all distributed services under analysis (*SUAs*) rather than adapting for a particular SUA.

Thus, we are developing CADAS, a control-based adaptive distributed online dynamic dependency analysis for distributed services, to address the limitations of SEADS hence reach the goal of utilizing as much analysis time budget as possible to gain as much in precision as possible. As surveyed in [9], control methods have been widely used and shown to be effective for adapting software systems. Since a program analysis is also a kind of software, intuitively the analysis should be beneficial from those methods too, especially when adaption is a design goal as in CADAS. Yet so far control-theoretical approaches are rarely seen as applied to adapt a program analysis. Specifically, CADAS uses the time costs of various analysis configurations as well as the knowledge about the precision of the analysis at different configurations as *control objectives*. Then, leveraging the relationships between those objectives and the analysis configuration items (as *control parameters*) as observed in sample executions of the SUA (as training samples), CADAS builds a feedback controller to achieve adaptation of its analysis algorithm to optimized balance between analysis cost and effectiveness (precision).

We implemented CADAS and applied it to six real-world industry-scale distributed SUAs. Our results revealed its cost-effectiveness and scalability advantages over a conventional dynamic analysis (with fixed configuration chosen), at least for the dynamic dependence computation at method level [10]. With 10 randomly chosen queries for each execution, CADAS responded with dependency results within acceptable time and under the budget. Compared to SEADS, CADAS achieved a higher budget utilization rate (87% versus 56.37%) and higher precision (87% versus 81.5%) on overall average.

II. APPROACH

CADAS works in four phases for its dependence analysis: pre-training, instrumentation, adaptation, and user interaction. There are three **inputs** from the user: the distributed (executable) SUA D with tests, a user budget B (i.e., a response time constraint for the analysis), and a dependence query Q.

Firstly, CADAS generates the training data (the relative precision and time costs) for refining its control-based model in the first phase (**pre-training**). In the second phase (**instrumentation**), CADAS creates an instrumented version D' of D, by inserting probes to monitor the *entry* (i.e., program control entering a method) and *returned-into* (i.e., program control

returning from a callee into a caller) events of each executed method as in [11]. Then, in the third phase (**adaptation**), D' continuously runs and CADAS continually performs adaptation for the dependence computation by tuning the analysis configuration via a control-based method. After the computation has finished, dependencies related to Q are delivered to a *querying_interface* attached to each process of the SUA. In the last phase (**user interaction**), relevant dependencies are received from the respective querying interface as the final result to answer Q. Among these phases, **instrumentation** and **user interaction** are similar to the original design of respective modules in SEADS. CADAS also considers the same analysis configuration design as SEADS does. Next, we elaborate the other two phases that differentiate our approach from SEADS.

Pre-training. This phase creates the precision and time costs for each of the analysis configurations that, when switching among them, the analysis cost-effectiveness would have the greatest variations. CADAS will use those sampled precision and time cost values as control conditions to compute new configurations via a feedback adaptive controller as follows.

Adaptation. The intuition behind our control-theoretical approach is that at any given time we have access to the user budget, hence via feedback we can leverage this information with insights from the training data to determine the optimal configuration. In particular, we design a simple feedback controller to predict a set of optimal configurations for the requested user budget. The novelty in our approach is that it incorporates human knowledge to design such a controller.

During the adaptation process, we analyzed the *linearity and time-invariability* (LTI) properties of the dependence analysis and found that the dependence computation time costs are constant for purely-dynamic configurations (i.e., those with which no static SUA information is utilized). This LTI nature of the dependence analysis is then exploited to a hierarchical model. More specifically, for each SUA, we first derive a hierarchy of time cost of the analysis under each configuration, which is used to realize a feedback control algorithm to predict the optimal cost-effectiveness for any specified user budget. Currently, this feedback control is simply a greedy algorithm searching on the hierarchy for configuration under which the analysis cost is closest to the budget (but not surpassing it) while analysis precision is the highest possible.

III. RESULTS

We have successfully applied CADAS against six industry-scale distributed Java services, including xSocket, Thrift, OpenChord, Voldemort, ZooKeeper, and Netty, along with the test inputs representing their typical operations (as detailed in [8]). These services cover different architectures, application domains, and scales. In each test, at least two server/-client instances were involved. Beyond SEADS, we considered DODA [12], a traditional dynamic dependence analysis without automatic configuration tuning, as another baseline.

Effectiveness. With the acceptably longer query response time (197 seconds) by average, CADAS was able to scale to real-world enterprise-scale distributed services with higher precision and user budget utilization rates (both more than 87%) than the better baseline (81.5% and 56.4% respectively). Learning from the pre-training data through statistical methods for specific executions and configurations, CADAS gained an advantage over the baseline SEADS for the cost-effectiveness. CADAS also had the scalability advantage over the baseline DODA which is too heavy to scale oftentimes.

Efficiency. CADAS had generally longer query response time than SEADS. For the SUAs and their executions that DODA can scale to, CADAS incurred about 4.6x runtime slowdown, compared to DODA causing 2–6x slowdown, with greater advantages on larger-scale systems. The reason is that CADAS immediately triggers the dependence computation when receiving a query. Overall, CADAS achieved significantly higher cost-effectiveness than both baselines.

IV. CONCLUSION

We gave a recap about CADAS, a self-adaptive dynamic dependency analysis for continuously running distributed services with concurrent, decoupled, and distributed processes. When compared to a state-of-the-art peer technique, CADAS achieved higher precision and budget utilization rate through a simple control-based method refined from pre-training samples (i.e., the relative precision and analysis time costs for some configurations). Also, CADAS has effectiveness, efficiency, and scalability advantages over a conventional dynamic analysis without automatic analysis configuration tuning, at least for method-level dynamic dependence computation. Through the results, we demonstrated how control-theoretical methods could be leveraged to automatically tune dynamic analysis.

ACKNOWLEDGMENT

This research was supported in part by Office of Naval Research (ONR) under Grant N000142212111.

REFERENCES

[1] X. Fu and H. Cai, "Measuring interprocess communications in distributed systems," in *ICPC*, 2019, pp. 323–334.

[2] ——, "Flowdist: Multi-staged refinement-based dynamic information flow analysis for distributed software systems." in *USENIX Security Symposium*, 2021, pp. 2093–2110.

[3] H. Cai and R. Santelices, "A framework for cost-effective dependence-based dynamic impact analysis," in *SANER*, 2015, pp. 231–240.

[4] X. Zhang *et al.*, "Dynamic Slicing Long Running Programs Through Execution Fast Forwarding," in *FSE*, 2006, pp. 81–91.

[5] H. Cai, R. Santelices, and D. Thain, "DiaPro: Unifying dynamic impact analyses for improved and variable cost-effectiveness," *TOSEM*, 2016.

[6] X. Fu and H. Cai, "Scaling application-level dynamic taint analysis to enterprise-scale distributed systems," in *ICSE-Companion*, 2020, pp. 270–271.

[7] X. Fu, H. Cai, and L. Li, "Dads: Dynamic slicing continuously-running distributed programs with budget constraints," in *ESEC/FSE*, 2020, pp. 1566–1570.

[8] X. Fu, H. Cai, W. Li, and L. Li, "Seads: Scalable and cost-effective dynamic dependence analysis of distributed systems via reinforcement learning," *TOSEM*, vol. 30, no. 1, pp. 1–45, 2020.

[9] S. Shevtsov, M. Berekmeri, D. Weyns, and M. Maggio, "Control-theoretical software adaptation: A systematic literature review," *TSE*, vol. 44, no. 8, pp. 784–810, 2018.

[10] H. Cai and R. Santelices, "Diver: Precise dynamic impact analysis using dependence-based trace pruning," in *ASE*, 2014, pp. 343–348.

[11] H. Cai and D. Thain, "DistIA: a cost-effective dynamic impact analysis for distributed programs," in *ASE*, 2016, pp. 344–355.

[12] H. Cai and X. Fu, "D2Abs: A framework for dynamic dependence analysis of distributed programs," *TSE*, vol. 48, no. 12, pp. 4733–4761, 2021.