Title TBD

Rodrigo Araújo
University of British Columbia
Vancouver, British Columbia
rodarauj@cs.ubc.ca

Reid Holmes
University of British Columbia
Vancouver, British Columbia
rtholmes@cs.ubc.ca

ABSTRACT

Due to the advancement in distributed systems and the increasing industrial demands, software systems contain multiple components with complex interactions, e.g databases and their replication, caching components, proxies and load balancers, application instances and their complex configuration parameters. The engineers in a project must thinker with many configuration parameters that change the behavior and/or structure of the system, this can cause many problems that affect the quality of the service. In other words, dealing with high dimensionality is both cognitively demanding and risky for the project.

In this work we show the design and analysis of a pragmatic machine learning based tool that aims to assist the engineering of systems that can: 1) monitor themselves, 2) Forecast workloads and performance metrics and 3) Change themselves in run-time by self-configuring and adapting for a specific scenario. After the integration of this tool with a system, it should be able to answer the question: given that we have many configuration parameters, how can we change them in order to optimize a certain metric for a given predicted workload?

We show that it can decrease the risk of changing systems' configurations in run-time and decrease the engineering effort that otherwise would be spent manually optimizing parameters, usually following a trial-and-error approach.

CCS CONCEPTS

• **TBD** → **TBD**;

KEYWORDS

ACM proceedings, LaTeX, text tagging

ACM Reference Format:
Rodrigo Araújo and Reid Holmes. 1997. Title TBD. In *Proceedings of ACM Woodstock conference (WOODSTOCK'97)*, Rodrigo Araújo and Reid Holmes (Eds.). ACM, New York, NY, USA, Article 4, 2 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

The industrial adoption of microservices has led to increasingly complex configuration schemes that are commonly fine-tuned manually by engineers. Ganek and Corbi (2003) discussed the need for autonomic computing to handle the complexity of managing software systems, it had been noticed that managing complex systems has grown too costly, prone to error and labor-intensive, because people under such pressure make mistakes, increasing the potential of system outages with a concurrent impact on business [cite The dawning of the autonomic computing era].

Paul Horn in his IBM's autonomic computing initiative (2001) described software self-adaptation in terms of a few characteristics of an autonomic computing system, which he called the "grand challenge", here we reorganize these characteristics into the following terminology:

- **Self-Awareness**: an autonomic system needs to know itself;
- **Self-configuring**: an autonomic system must configure and reconfigure itself under varying and unpredictable conditions;
- **Self-optimization**: an autonomic system never settles for the status quo, it always looks for ways to optimize its workings, it will anticipate the optimized resources needed to meet a user's information needs while keeping its complexity hidden;
- **Self-healing**: An autonomic system must perform something akin to healing, it must be able to recover from routine and extraordinary events that might cause some parts to malfunction.

Although this has driven many researchers to study self-adaptive systems over the years [cite them], the software industry still lacks practical tools to provide self-adaptation mechanisms to their systems, and thus, most of the configuring and tuning of the systems happens manually, often in run-time, which is known to be a very time consuming and risky practice.

In this work we focus on the practical realization of self-configuring and self-optimization, based on ideas already discussed, but following alternatives approaches to provide an accessible tool to build self-adaptive systems, such approaches are:

- Use of system observability to collect a sufficiently big time series dataset that represents the state of a system and its components in relation to time;
- Use of Machine Learning applied to time series data to analyze and predict workload and optimal configuration in order to provide adaptation plans;
- Use of Online Learning to provide constant relearning of the model to adapt to different scenarios and requirements;
- Use of Service Level Objectives as a way to define the system's performance goals that will be used by the tool as optimization objectives;
- Use of the concepts in control theory as the central component of this tool, in which it implements a variation of the well known and studies MAPE loop (Monitor, Analyze, Plan, Execute);

WOODSTOCK'97, July 1997, El Paso, Texas USA

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

- Abstraction of all the concepts above into a tool that can be integrated to compatible systems, including systems that were built without self-adaptation in mind.

The rest of the paper is structure as following:

2 RELATED WORK

Maybe related work would be better now instead of putting it in the end, since there are a lot of related work and they are very relevant to this work, it would be nice to upfront point out what has been done and what would be different in this work.

3 APPROACH

3.1 Control theory and self-adaptive systems

3.2 System's configuration as an optimization problem

3.3 Providing system adaptation with machine learning

3.4 Workload simulation

3.5 System instrumentation

3.6 Machine learning architecture

3.6.1 *Features and models.*

3.6.2 *Online training.*

3.6.3 *Achieving self-adapation.*

4 IMPLEMENTATION

5 EVALUATION AND DISCUSSION

6 FUTURE WORK

7 CONCLUSIONS

ACKNOWLEDGMENTS

REFERENCES

[1] [n. d.]. short term performance forecasting in enterprise systems. ([n. d.]). http://www.hpl.hp.com/techreports/2005/HPL-2005-50.pdf

[2] [n. d.]. Using probabilistic reasoning to automate software tuning. ([n. d.]). http://ftp.deas.harvard.edu/techreports/tr-08-04.pdf

[3] Virgilio AF Almeida. 2002. Capacity Planning for Web Services.. In *Performance*. Springer, 142–157. http://link.springer.com/content/pdf/10.1007/3-540-45798-4.pdf#page=151

[4] Joy Arulraj Haibin Lin Jiexi Lin Lin Ma Prashanth Menon Todd Mowry Matthew Perron Ian Quah Siddharth Santurkar Anthony Tomasic Skye Toor Dana Van Aken Ziqi Wang Yingjun Wu Ran Xian Tieying Zhang Andrew Pavlo, Gustavo Angulo. 2017. *Self-Driving Database Management Systems.*

[5] Ira Cohen, Jeffrey S. Chase, Moises Goldszmidt, Terence Kelly, and Julie Symons. 2004. Correlating Instrumentation Data to System States: A Building Block for Automated Diagnosis and Control.. In *OSDI*, Vol. 4. 16–16. http://sailhome.cs.queensu.ca/~corpaul/ciscxxx/papers/HPL-2004-183.pdf

[6] Rogério de Lemos, Holger Giese, Hausi A. Müller, and Mary Shaw (Eds.). 2013. *Software Engineering for Self-Adaptive Systems II.* Lecture Notes in Computer Science, Vol. 7475. Springer Berlin Heidelberg, Berlin, Heidelberg. http://link.springer.com/10.1007/978-3-642-35813-5 DOI: 10.1007/978-3-642-35813-5.

[7] Serge Demeyer. 2011. Research methods in computer science.. In *ICSM*. 600. http://ansymore.uantwerpen.be/system/files/uploads/courses/CapitaSelecta/ResearchMethds00_Contents.pdf

[8] Dirk Draheim, John Grundy, John Hosking, Christof Lutteroth, and Gerald Weber. 2006. Realistic load testing of web applications. In *Software Maintenance and Reengineering, 2006. CSMR 2006. Proceedings of the 10th European Conference on*. IEEE, 11–pp. http://ieeexplore.ieee.org/abstract/document/1602358/

[9] F. Faniyi, P. R. Lewis, R. Bahsoon, and X. Yao. 2014. Architecting Self-Aware Software Systems. In *2014 IEEE/IFIP Conference on Software Architecture*. 91–94. https://doi.org/10.1109/WICSA.2014.18

[10] Archana Sulochana Ganapathi. 2009. *Predicting and Optimizing System Utilization and Performance via Statistical Machine Learning.* Ph.D. Dissertation. EECS Department, University of California, Berkeley. http://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-181.html

[11] Alan G. Ganek and Thomas A. Corbi. 2003. The dawning of the autonomic computing era. *IBM systems Journal* 42, 1 (2003), 5–18.

[12] Martin Georgiev, Subodh Iyengar, Suman Jana, Rishita Anubhai, Dan Boneh, and Vitaly Shmatikov. 2012. The most dangerous code in the world: validating SSL certificates in non-browser software. In *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 38–49. http://dl.acm.org/citation.cfm?id=2382204

[13] Nikolas Roman Herbst, Nikolaus Huber, Samuel Kounev, and Erich Amrehn. 2014. Self-adaptive workload classification and forecasting for proactive resource provisioning. *Concurrency and Computation: Practice and Experience* 26, 12 (Aug. 2014), 2053–2078. https://doi.org/10.1002/cpe.3224

[14] P Horn. 2001. Autonomic Computing: IBM's Perspective on the State of Information Technology. 2007 (10 2001).

[15] Shijun Liu, Zeyu Di, Lei Wu, Li Pan, and Yuliang Shi. 2016. Probabilistic-based workload forecasting and service redeployment for multi-tenant services. *International Journal of High Performance Computing and Networking* 9, 1/2 (2016), 134. https://doi.org/10.1504/IJHPCN.2016.074666

[16] Theophano Mitsa. 2010. *Temporal data mining.* Chapman & Hall/CRC, Boca Raton, FL. OCLC: ocn474869534.

[17] Jeho Oh, Don Batory, Margaret Myers, and Norbert Siegmund. 2017. Finding near-optimal configurations in product lines by random sampling. ACM Press, 61–71. https://doi.org/10.1145/3106237.3106273

[18] Barry Porter, Matthew Grieves, Roberto Rodrigues Filho, and David Leslie. 2016. REX: A Development Platform and Online Learning Approach for Runtime Emergent Software Systems. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. USENIX Association, GA, 333–348. https://www.usenix.org/conference/osdi16/technical-sessions/presentation/porter

[19] M. Salehie and L. Tahvildari. 2009. Self-adaptive software: Landscape and research challenges. ACM Transactions on Autonomous and Adaptive Systems (TAAS).

[20] Stepan Shevtsov, Mihaly Berekmeri, Danny Weyns, and Martina Maggio. 2017. Control-Theoretical Software Adaptation: A Systematic Literature Review. *IEEE Transactions on Software Engineering* (2017), 1–1. https://doi.org/10.1109/TSE.2017.2704579