A Generalizable Framework for Automated Cloud Configuration Selection

Supervisors: Adam Barker & Yuhui Lin

Jack Briggs - 140011358

MSc Data-Intensive Analysis

2019-06-06

Abstract

Outline of the project using at most 250 words

Declaration

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated. The main text of this project report is NN,NNN* words long, including project specification and plan. In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker, and to be made available on the World Wide Web. I retain the copyright in this work.

Contents

List of Figures

1 Introduction

Describe the problem you set out to solve and the extent of your success in solving it. You should include the aims and objectives of the project in order of importance and try to outline key aspects of your project for the reader to look for in the rest of your report.

2 Context Summary

Surveying the context, the background literature and any recent work with similar aims. The context survey describes the work already done in this area, either as described in textbooks, research papers, or in publicly available software. You may also describe potentially useful tools and technologies here but do not go into project-specific decisions.

Cloud computing, specifically Infrastructure-as-a-Service (IaaS), has opened the doors to allow organizations and developers to utilize a diverse range of computer resources on demand without any up-front commitment or cost [1]. These resources are generally provided in the form of Virtual Machines (VMs) with a diverse range of options, and correctly selecting the most appropriate cloud configuration for a given task or service can reduce costs for the user and free up underutilized resources for the cloud provider. Depending on the task, choosing a sub-optimal configuration can lead to up to 12 times higher costs in the worst case, but choosing randomly can lead, on average, to around twice the cost. [2].

Selecting the 'optimal' cloud configuration for a given job is not trivial, however. This is due to the large diversity in service, the difficulty in predicting performance, and the flexibility of the term 'optimal.' The diversity in services provided by different cloud providers creates a large search space. At the time of writing, Amazon EC2[1] alone offers over 200 predefined instance models[2], while Google Compute Engine[3] allows users to define their own machine types, ranging from small VMs with 1 vCPU and 1 GB of RAM to 96 vCPUs with 624 GB of RAM, and includes options for specifying the CPU platform or adding GPUs.

Once given a specific configuration, predicting its performance for a given job is difficult. A number of recent studies have shown that previously reported variability [3] in performance for different instances of a given instance type has dramatically improved [4–6], particularly for AWS, leading to much more consistent performance on a given instance type. However, even when stable, predicting performance is still far from trivial.

[1]`https://aws.amazon.com/ec2/`
[2]`https://www.ec2instances.info`
[3]`https://cloud.google.com/compute/`

'Larger' VM types may not provide improved performance [7], or may do so at a far less cost-efficient rate. Optimizing for a good mean performance may fail to optimize for the tail when variance or high stress leads to lower than expected performance. In addition, workload resource requirements are hard to model, with jobs using different resources at different times, and performance may have non-linear relationships with these resources [2]. A method to predict performance should account for these hurdles. A costly option is to simply run the job or service itself on the instance type and measure the performance directly. For batch jobs, the cost of the initial performance testing can be offset by later repeats. In 2012, it was reported that up to 40% of Microsoft's analytics jobs are recurring [8, 9]. Another, faster, option is to run benchmarks or micro-benchmarks on the target machine, and use these to estimate performance. Benchmarking suites such as Cloud workbench [10] and CloudSuite [11] allow application-specific performance prediction, and micro-benchmarking has shown success in rapidly estimating performance of appropriate applications with errors between 10-20% [4, 12, 13]. However, using benchmarks to predict performance requires knowing how the benchmarked resources will relate to job performance, which limits the flexibility of an automated solution.
Finally, what is meant by the 'optimal' configuration may differ. For a batch job, it may simply be the cheapest option to complete the job, or it may be the cheapest option that can complete the job within a given time-constraint. For a service, there will likely be a trade-off between consistency, latency, and cost, and where the optimal balance lies will depend on the user's preferences or requirements.

In order to discover the optimal cloud configuration for a job, a user must first decide what 'optimal' means, find a way to evaluate the performance of a configuration given that metric, and search or model the diverse range of available configurations using this evaluation until they are confident they have found the best available configuration. Unsurprisingly, given the challenges described above, this has been difficult to automate. PARIS [7] uses Random Forests to model how benchmarks will relate to an application's performance, in order to improve on performance evaluation. CherryPick [2] instead focuses on reducing search space, by using Bayesian optimization [14] to minimize the number of samples needed to confidently estimate the optimal configuration. Bayesian Optimisation is perfectly suited to the problem, as it is non-parametric, typically needs only few samples to find a near-optimal solution, and can tolerate uncertainty in both the sampling and performance modelling [15]. Okta [5], built specifically for web-serving applications, leaves it to the user to input which cloud configurations to test and what 'benchmark driver' to use to test their performance, giving it good generalizability for any service that takes end-user requests. However, none of these examples provide usable implementations for their solutions.

3 Requirements Specification

Capturing the properties the software solution must have in the form of requirements specification. You may wish to specify different types of requirements and given them priorities if applicable.

4 Software Engineering Process

The development approach taken and justification for its adoption.

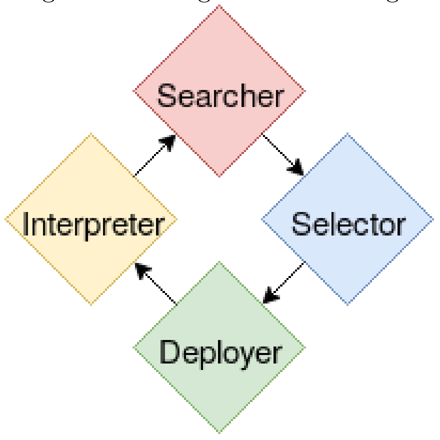
5 Ethics

Any ethical considerations for the project. You should scan the signed ethical approval document, and include it as an appendix.

6 Design

Indicating the structure of the system, with particular focus on main ideas of the design, unusual design features, etc.

Figure 1: A diagram of the design.



7 Implementation

How the implementation was done and tested, with particular focus on important / novel algorithms and/or data structures, unusual implementation decisions, novel user interface features, etc.

8 Evaluation and Critical Appraisal

You should evaluate your own work with respect to your original objectives. You should also critically evaluate your work with respect to related work done by others. You should compare and contrast the project to similar work in the public domain, for example as written about in published papers, or as distributed in software available to you.

9 Conclusions

You should summarise your project, emphasising your key achievements and significant drawbacks to your work, and discuss future directions your work could be taken in.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," *EECS Department, University of California, Berkeley*, pp. 1–25, feb 2009.

[2] O. Alipourfard, H. H. Liu, J. Chen, S. Venkataraman, M. Yu, M. Zhang, Y. University, H. Harry Liu, J. Chen, S. Venkataraman, M. Yu, and M. Zhang, "CherryPick: Adaptively Unearthing the Best Cloud Configurations for Big Data Analytics," in *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, pp. 469–482, 2017.

[3] P. Leitner and J. Cito, "Patterns in the Chaos - a Study of Performance Variation and Predictability in Public IaaS Clouds," *ACM Transactions on Internet Technology*, vol. 16, pp. 1–23, apr 2014.

[4] J. Scheuner and P. Leitner, "Estimating Cloud Application Performance Based on Micro-Benchmark Profiling," in *IEEE International Conference on Cloud Computing, CLOUD*, vol. 2018-July, pp. 90–97, IEEE, jul 2018.

[5] C. Davatz, C. Inzinger, J. Scheuner, and P. Leitner, "An Approach and Case Study of Cloud Instance Type Selection for Multi-Tier Web Applications," in *Proceedings - 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID 2017*, pp. 534–543, IEEE, may 2017.

[6] C. Laaber, J. Scheuner, and P. Leitner, "Software microbenchmarking in the cloud. How bad is it really?," *Empirical Software Engineering*, pp. 1–40, apr 2019.

[7] N. J. Yadwadkar, B. Hariharan, J. E. Gonzalez, B. Smith, and R. H. Katz, "Selecting the best VM across multiple public clouds," in *Proceedings of the 2017 Symposium on Cloud Computing - SoCC '17*, (Santa Clara, CA, USA), pp. 452–465, ACM Press, 2017.

[8] S. Agarwal, S. Kandula, N. Bruno, M.-C. Wu, I. Stoica, and J. Zhou, "Re-optimizing data-parallel computing," in *NSDI'12 Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation*, pp. 281–294, USENIX, 2012.

[9] A. D. Ferguson, P. Bodik, S. Kandula, E. Boutin, and R. Fonseca, "Jockey: Guaranteed Job Latency in Data Parallel Clusters," in *EuroSys '12 (Proceedings of the 7th ACM European Conference on Computer Systems)*, pp. 99–112, 2012.

[10] J. Scheuner and P. Leitner, “A Cloud Benchmark Suite Combining Micro and Applications Benchmarks,” pp. 161–166, 2018.

[11] T. Palit, Y. Shen, and M. Ferdman, “Demystifying cloud benchmarking,” in *ISPASS 2016 - International Symposium on Performance Analysis of Systems and Software*, pp. 122–132, IEEE, apr 2016.

[12] B. Varghese, L. T. Subba, L. Thai, and A. Barker, “DocLite: A Docker-Based Lightweight Cloud Benchmarking Tool,” in *Proceedings - 2016 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2016*, pp. 213–222, IEEE, may 2016.

[13] B. Varghese, L. T. Subba, L. Thai, and A. Barker, “Container-based cloud virtual machine benchmarking,” in *Proceedings - 2016 IEEE International Conference on Cloud Engineering, IC2E 2016: Co-located with the 1st IEEE International Conference on Internet-of-Things Design and Implementation, IoTDI 2016*, pp. 192–201, IEEE, apr 2016.

[14] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, “Taking the human out of the loop: A review of Bayesian optimization,” *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[15] D. R. Jones, M. Schonlau, and W. J. Welch, “Efficient Global Optimization of Expensive Black-Box Functions,” *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.

Appendices

Testing Summary

User Manual