PATAT 2014, 26-29 August 2014, York, United Kingdom

Predictive scheduling for optimal cloud configuration

Michael G. Epitropakis · Andrea Bracciali · Marco Aldinucci · Emily Potts · Edmund K. Burke

1 Introduction

Cloud Computing provides availability of a large amount of resources in a seamless and tailored way to a vast public of users, following the paradigm of computation as a utility, which, like all the more traditional utilities, comes with an associated cost. Users can access resources through facilitated interfaces supported by the vast majority of service providers. However, non-trivial Cloud Computing usually requires advanced skills to configure and manage the computation in an optimal way. Configuration and management of the available resources might include either selection of "optimal" parameters for the utilized software in order to efficiently use the available resources, or selection of the "optimal" hardware resources in order to minimize renting cost and maximize efficiency.

Very relevant is the trade-off between efficiency and costs, which is also typically quite difficult to manage, because of intrinsic unpredictability of computation and the consequent lack of accurate cost models for these architectures. Even if the unit cost of different architecture components, e.g. bandwidth, different classes of virtual machines and storage, is known, determining beforehand the exact cost of running a complex parallel or distributed application over different, possibly dynamic, configurations is currently an open problem.

Several approaches address the problem as a multi-objective optimisation problem on (very) simplified *"theoretical models"* of the architecture. These models are general and typically unrelated to the specific performances of the

Michael G. Epitropakis · Andrea Bracciali · Emily Potts · Edmund K. Burke
Computing Science and Mathematics,
University of Stirling, UK.
E-mail: {mge,abb,epo}@cs.stir.ac.uk, e.k.burke@stir.ac.uk

Marco Aldinucci
Computer Science Department,
University of Torino, Italy.
E-mail: aldinuc@di.unito.it

architecture for the specific *problem at hand*. For instance, in [8] different virtual machines are abstracted in terms of their brute Gb-per-second theoretical processing capability.

Another class of approaches is based on *"experimental models"*, which build on top of knowledge about past executions. For instance, in [6] past executions are clustered in classes of problems, whose optimal time and cost configuration can be inferred from past experience. Then, mapping the *problem at hand* to one of such classes gives indications about how to configure Cloud Computing architectures for best time and cost performances.

A critical aspect of the above modelling approaches is the difficulty, i.e. the cost ultimately, of collecting suitable information for devising accurate and specific cost models for the *problem at hand*. Theoretical approaches by-pass such costs, losing accuracy. Experimental approaches increase accuracy, but at the cost of accumulating enough experience and hence potentially undermining the cost-optimisation goals. Furthermore, properly mapping the specific problem under consideration to the so-far available experience base might be difficult and introduces back a loss of accuracy.

Bridging the two approaches, we investigate the possibility of building a model from *limited experimental information* on a specific problem, on top of which we build a "theoretical" model. The model, then, reflects some specific information about the *problem at hand*, hence improving domain knowledge with respect to the theoretical approaches, but the price needed to accumulate large experimental knowledge is waived.

On top of our model, we perform state of the art evolutionary multi-objective optimisation to discover optimal configurations. Multi-objective optimization formulations have been well established and widely adapted in various fields of Software Engineering [5], such as Requirements Engineering [10], Software Testing [3, 9] and Cloud Computing [4].

2 Combining models and experimental data for optimised cost predictions

We focus on (simplistic) variants of the *MapReduce* programming model [1], as provided by the Amazon EC2 Cloud. MapReduce is very relevant in Cloud Computing as it provides a high-level programming interface easing the development of efficient, scalable and robust applications and services.

We initially consider three benchmark problems, one CPU bounded, one I/O bounded and one making a large use of bandwidth. These broad categories characterise several aspects of Cloud Computing, other choices are possible. For each benchmark, a minimal configuration of the problem, a single task, is executed in one instance, i.e. a virtual machine, provided by EC2. Small, medium and large instances are available with different performances and costs. We hence use *averaged* minimal tasks, i.e. uniform representatives of tasks computational requirements, to *experimentally* gauge instances performances on the specific problem at hand. Then, we develop a *theoretical*

model that gives an approximation of how performances scale up when many tasks are executed on combinations of several instances of different kinds. The models are based on theoretical limits of performance scaling, and may consider, where appropriate, the cost of distributing data and collecting results, again inferred from simple (simplistic) experimental measurements, the costs associated to storage, and specific costing policies, such as pricing per temporal intervals of a given duration, and other problem- and architecture-specific information.

On top of our model, we formulate a multi-objective optimisation to tackle resource allocation and task scheduling. We use multi-objective optimisation analysis to determine optimal configurations of resources (number and kind of instances) and schedule of tasks. More precisely, we aim at minimising both *execution costs* for the MapReduce execution, and its *completion time* (makespan). Although, we focus on these two objectives, the formulation is general and capable of handling more objectives, based on the optimisation needs of the user.

In the current study, we utilise two well known evolutionary multi-objective algorithms that have demonstrated efficient behaviour in Search-based Software Engineering optimisation problems, namely the Non-dominated Sorting Genetic Algorithm (NSGA-II) [2], and the Two-Archive algorithm (TAEA) [7]. NSGA-II uses a fast non-dominated sorting procedure to enhance the diversity and the quality of the resulting solutions, while TAEA incorporates two archives with different properties to save potentially good solutions. The former aims at enhancing the diversity of the solutions found, while the latter aims at enhancing the convergence of the algorithm to the real Pareto front. In the current setting the representation of a solution encodes both the resource allocation and task scheduling goals, while both algorithms have been used with their default search and parameter settings.

An extensive empirical verification of the proposed methodology has been designed. The algorithmic part of the methodology can be enriched by developing an adaptive hyper-heuristic version of the algorithms mentioned above to enhance their searching abilities and their performance in terms of locating optimal and diverse solutions.

A key challenge for the proposed methodology is to enable the user to make fast and cost/performance optimal decisions when configuring Cloud Computing services for novel MapReduce based problems. User decisions will be less dependent on costly and hard-to-generalise past experimental information than the decisions based on traditional “experimental approaches”, and a bit more informed on the problem at hand than the decisions based on “theoretical approaches”.

Acknowledgements E.K. Burke and M.G. Epitropakis would like to thank EPSRC for their support for this work through grant EP/J017515/1.

References

1. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. In: Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6, OSDI'04, pp. 10–10. USENIX Association, Berkeley, CA, USA (2004)
2. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)
3. Harman, M.: Making the case for morto: Multi objective regression test optimization. In: Fourth International IEEE Conference on Software Testing, Verification and Validation, ICST 2012, Berlin, Germany, 21-25 March, 2011, Workshop Proceedings, pp. 111–114. IEEE Computer Society (2011)
4. Harman, M., Lakhotia, K., Singer, J., White, D.R., Yoo, S.: Cloud engineering is search based software engineering too. Journal of Systems and Software **86**(9), 2225–2241 (2013)
5. Harman, M., Mansouri, S.A., Zhang, Y.: Search-based software engineering: Trends, techniques and applications. ACM Computing Surveys **45**(1), 11:1–11:61 (2012)
6. Lama, P., Zhou, X.: Aroma: Automated resource allocation and configuration of mapreduce environment in the cloud. In: Proceedings of the 9th International Conference on Autonomic Computing, ICAC '12, pp. 63–72. ACM, New York, NY, USA (2012)
7. Praditwong, K., Yao, X.: A new multi-objective evolutionary optimisation algorithm: The two-archive algorithm. In: Y. Wang, Y.m. Cheung, H. Liu (eds.) Computational Intelligence and Security, *Lecture Notes in Computer Science*, vol. 4456, pp. 95–104. Springer Berlin Heidelberg (2007)
8. Tsai, J.T., Fang, J.C., Chou, J.H.: Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm. Computers & Operations Research **40**(12), 3045–3055 (2013)
9. Yoo, S., Harman, M.: Regression testing minimization, selection and prioritization: A survey. Software Testing, Verification & Reliability **22**(2), 67–120 (2012)
10. Zhang, Y., Finkelstein, A., Harman, M.: Search based requirements optimisation: Existing work and challenges. In: B. Paech, C. Rolland (eds.) Requirements Engineering: Foundation for Software Quality, *Lecture Notes in Computer Science*, vol. 5025, pp. 88–94. Springer Berlin Heidelberg (2008)