Runtime Infrastructure Optimisation in Cloud IaaS Structures

Kleopatra Chatziprimou
Department of Informatics
King's College London, UK
Email: kleopatra.chatziprimou@kcl.ac.uk

Kevin Lano
Department of Informatics
King's College London, UK
Email: kevin.lano@kcl.ac.uk

Steffen Zschaler
Department of Informatics
King's College London, UK
Email: szschaler@acm.org

***Abstract*—The requirement for elasticity involves the ability of cloud datacenters to add or remove resources at a fine grain and with a lead time of seconds, closely matching resources to the actual demand conditions. Elasticity techniques can assist cloud stakeholders in regards to: (i) alleviating datacenter capital and operating costs, (ii) keeping cloud services continuously available, (iii) supporting market flexibility. To date, most infrastructure scaling methodologies can provide resource reconfiguration decisions to maintain quality properties under environment changes. However, issues related to the timeliness of reconfiguration decisions under dynamic changes are not adequately addressed. In this paper, we describe the PhD motivation, research questions and methodology towards developing cloud infrastructure and hosted services QoS optimisations under environment uncertainties.**

I. Introduction

The cloud paradigm is shifting computing from the traditional overprovisioned infrastructure model, to a model that treats software and hardware as rented commodities. Computing resources (e.g., remote storage) accessed via APIs, are provided by a cloud infrastructure provider (IP) to a client. Often a client is a cloud service provider (SP), i.e., a third party company who uses the computing facilities to provide a service to their end users [1].

Cloud services execute in virtual machines (VMs), which are hosted in physical machines (PMs) and located in geographically distributed datacenters. The allocation of services to VMs and VMs to PMs forms an infrastructure configuration. The precise characteristics of a configuration, are based on stakeholders (i.e., IP, SP) requirements and may have to combine software (e.g., performance, reliability, security) and business quality attributes (e.g., costs, time-to-market). We consider a configuration to be optimal when it satisfies all stakeholders non-functional goals. Often, software quality attributes conflict: e.g., security versus reliability. In general improving one quality property can deteriorate another thus quality properties cannot be improved in isolation. Consequently, configurations that exhibit a good trade-off between multiple quality criteria must be found.

Full exploration of all possible configuration options is not feasible for enterprise clouds comprised of thousands of computing nodes. Consider for example the case of a datacenter comprised of 5 active heterogeneous PMs and 10 VMs hosting the current services. Although the example is trivial it is not obvious how to change the configuration to improve performance in case of e.g., a load spike. The infrastructure architect might consider rebalancing the CPU and RAM capabilities of the VMs to increase the system's processing power. Assuming 9 CPU size ranks and 8 RAM size ranks available to each VM, there are $10 * 8 * 9 = 720$ VM configuration possibilities. The architect might also consider switching on 2 more PMs and rebalance the 10 VMs to the 7 now active PMs, resulting to $7^{10} = 282475249$ additional configuration options. Assuming that evaluation of each configuration candidate takes 1 second, full exploration of the design space will take ~ 9 years. The design space of this trivial example is overwhelmingly large to be neither manually nor fully explored.

Collecting good-enough candidates at reasonable time often suffices to solve architectural decision problems [2]. Still, early considerations of architectural trade-offs alone could not satisfy the whole system life-cycle, as cloud environments are highly volatile. For example the infrastructure usage will vary according to workload fluctuations. The deregulated energy market enables competitive energy pricing contracts, therefore energy price variations may also appear during datacenter operation.

Such changes affect the underlying optimisation problem instance or constraints and thus, the optimum of the problem might change as well. For example in a workload spike, it is critical to find configurations that activate an optimal number of servers to support the increased load achieving moderate increase of operating costs. When the demand decreases, it is critical to search for configurations that deactivate idle resources to save operating costs, without compromising performance. Therefore, to ensure satisfactory quality under changing conditions, dynamic management of the infrastructure via runtime adaptations is critical.

The inherent trade-offs between optimisation objectives and the cost of adaptation itself makes runtime optimisation decisions complex. Solving optimisation problems at runtime, is challenging due to the time and computational costs needed to calculate the solutions. When the environment is changing it may be better to make a suboptimal decision fast, rather than invest time searching for design alternatives that may never recoup this investment. A long search for alternatives will keep the datacenter in the current configuration state, which is not necessarily close to optimal after the environment changed. This means that the stakeholders might risk revenue losses, while the optimisation process wastes additional computational resources. Overall it is critical that the optimisation process will not consume more resources than it can save.

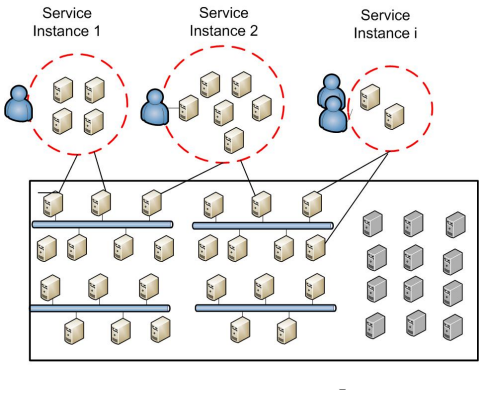


Fig. 1: Configuration Example.

The goal of our research is to determine how to automatically adapt cloud infrastructure configurations to dynamic changes at runtime. Our focus in this problem is achieving a useful balance between the feasible optimality of architecture candidates and timeliness of the optimisation procedure. Overall, our methodology proposes the integration of (i) formal models to describe the infrastructure architecture and how the architecture can change to affect quality attributes (ii) evolutionary optimization techniques and (iii) data-driven approximations, to estimate the quality of a configuration based on history observations. The rest of this paper is organized as follows. Section II presents our motivating scenario. Section III summarizes the state-of-the-art. Discussion on our research questions is given in Section IV while Section V details our methodology. Conclusions are presented in Section VI.

II. Motivation

To quickly convey our motivating concepts, we provide a simple use case example in the following. Consider an IP with geographically distributed datacenters. Each cloud is comprised of M PMs and N VMs. An SP will rent part of the infrastructure to provide web-hosting services (i.e., on-line newspaper) to its clients. Web hosting services are considered 3-tier (i.e., web, application and database). Replicas of each tier are hosted in each VM. The workload of the services is represented by the mean request rates for each tier. Let us further consider a Service level Agreement (SLA) which sets the average service response time threshold at 2 seconds.

To indicate the conflicts in priorities among stakeholders, we assume here that the best interest of the SP is maximum services performance measured as minimum mean response time. Therefore SP anticipates unlimited infrastructure scaling to support every possible incoming workload. On the other hand, the more workload is served, the more infrastructure resources must be activated to host it. Thus operating costs as energy consumption are increased. While the IP must be able to provide sufficient infrastructure resources, his budget is constrained. The challenge is to identify the best configuration options, which optimally balance trade-offs between performance and available budget. Figure 1 shows the selected configuration, where part of the infrastructure serves the requested service instances while another part is deactivated to regulate energy consumption.

Changes in the environment will affect the underlying optimisation problem's parameters and therefore the initially deployed best trade-off architecture. In case of a load surge in service instance i, new hardware servers must be activated to manage the increased load. A new supplier contract might result to reduced energy price. The infrastructure must react exploiting the possibility to activate more servers within budget to further improve performance. In any case, the new configuration has to be decided and deployed within the SLA threshold of 2 seconds. Otherwise, IP may risk revenue losses due to SLA violation penalties. Towards addressing these concerns, we must consider an automated and adaptive optimisation approach to enable fast convergence to good-quality design alternatives.

III. Related Work

In current literature the challenge of managing transient adaptation costs has been partially considered. A sophisticated approach is developed by in [3] that considers the time costs of adaptation actions. Yet, assumptions are made about static datacenter structures over time, which simplify the problem but limit the applicability of the solution in real world environments, which are highly dynamic. Similar static assumptions of e.g. a-priori known migration cost [4] fail to apply to dynamic systems as migration costs are dependent on memory use [5], which is dependent on time-varying workload requirements therefore cannot be static during datacenter operation. Other approaches targeted for online optimisation, use greedy heuristics and very few degrees of freedom. The corresponding search problem has small size and may be fast to solve [6]. However, the optimisation outputs can be low quality approximations of the true optima. The requirement for fast adaptation must not vastly degrade the quality of architecture candidates as they are related with financial risks.

In the literature, the extensive use of bin packing heuristics provides best-effort actions but neglect in depth considerations of conflicts between optimisation objectives [7]. Adaptation actions are applied based on greedy rules that do not allow simultaneous optimisation of multiple conflicting criteria and may cause traps to local optima. Additionally, greedy solutions can lead to instability inside the datacenter and oscillations between states. For example in [8] greedily switching on servers to host new services can lead to violations of allowed energy consumption. In a similar fashion, servers will be greedily switched off, possibly triggering performance degradation of hosted services, that will call again for activating new servers etc.

Another challenging direction is the requirement for automation during the search for optimal architecture candidates. A number of approaches require formulation of constraints to guarantee desired properties of the datacenter entities [9]. However, such constraints must be manually tuned when changes in the environment occur. Manual tasks are usually error-prone and time consuming therefore not suitable for enterprise clouds.

Uniform exploration of the possible architecture candidate space is not always assured as a number of approaches consider only a limited number of degrees of freedom towards tuning the quality of the current configuration [6]. For example if only

VM migrations are supported, energy-saving actions such as switching off unused VMs cannot be explored. Furthermore, VM migrations are computationally expensive and depending on the scenario cheaper adaptation actions could be also appropriate. On the other hand, exhaustive exploration of all possible configuration solutions within a prediction horizon as presented in [10] can be very time-consuming, therefore not suitable at runtime. Furthermore, the overwhelming size of the design space may restrict the applicability of the approach to trivial systems (e.g., authors in [10] experiment on a testbed of only 6 servers).

Finally, a number of approaches abstract useful details from the problem domain e.g., the hosted services behaviour. For example authors in [11] consider incoming workload in terms of requirements to host a particular number of VMs to PMs considering the problem a variation of bin-packing. While such approaches simplify the problem model, a knowledge about the services behaviour (i.e., arrival rates, component structure) may offer more precise management.

IV. Research Questions and Challenges

The goal of this PhD is to develop an automated framework for runtime infrastructure optimisation in dynamic cloud environments. The optimisation procedure considers multiple quality criteria as well as the cost of the adaptation itself, in terms of time needed to identify changed optima. The following are major identified research questions.

RQ1: How to Automatically Improve Configurations ?
Automation support is necessary as it can improve the speed and cost of the search of configuration candidates. Configurations must not change arbitrarily during optimisation. To the contrary each configuration must adhere to predefined design constraints. The goal of an automated improvement process is to always guarantee meaningful design alternatives, eliminating manual sanity checks. To this end, a formalisation of the cloud infrastructure core structures and degrees of freedom is required to enforce design constraints and specify which valid changes may be implemented towards tuning the architecture quality attributes without affecting the functionality of the system.

RQ2: How to Search for Good-Quality Configurations ?
In the space of all possible solutions, architectures that exhibit a good trade-off between multiple quality criteria must be tracked. As the design space to be explored can be vast, full exploration of all possible configurations is not realistic. At the same time, finding the exact globally-optimal solutions is not necessarily required; a good approximation found in reasonable time often suffices to solve a given design decision problem. To this end, we propose the use of population-based evolutionary meta-heuristic techniques to encode the challenge of optimising architectures as a search problem. They search for Pareto-optimal design candidates i.e., candidates that are superior to any other candidates in at least one quality criterion. The application of evolutionary meta-heuristics seems natural fit for our problem because they (i) treat the function to be optimised as a black box (ii) are simple, as one can make progress with only two ingredients: a choice for representation of the problem and a definition of the fitness function capturing the quality criteria to be optimised (iii) are robust and with appropriate parameter tuning their performance comfortably outperforms purely random search (iv) can demonstrate scalability though parallel execution of candidate fitness computations (v) are any-time algorithms (i.e., may trade deliberation time for quality of results) and therefore are appropriate for time critical domains.

RQ3: How to Accelerate the Search Process ?
To estimate the fitness of configuration candidates against the required quality criteria a full simulation or physical experiment is required, because of lack of an analytic mathematical formula to calculate the candidates quality. Therefore, the number of calls to the fitness function must be limited because each evaluation can be very time consuming and computationally expensive. In our use case for example, during the quality evaluation phase, the simulation candidates to measure their fitness is in the order of hours while the runtime SLAs are in the order of seconds. Towards reducing the cost of evaluations, fitness functions can be approximated based on data generated from experiments or simulations. These approximate models are orders of magnitude cheaper to run than the full analysis [12]. The approximated and original fitness model must be used together to achieve a balance between accuracy of the approximation and cost of calling the original fitness function [12]. Strategies towards fitness approximations are based on machine learning and may be classified into the following categories. (i) *Data-driven approximations/ Surrogates*: Provide predictions of the fitness using a training set of evaluated points. (ii) *Fitness imitation*: This type of approximation aims at saving function evaluations by estimating a candidate's fitness from similar candidates.

Popular techniques towards fitness imitation are clustering, and instance based learning. Clustering techniques divide candidates into several groups according to their similarity. Then only an individual represents the fitness of each cluster. A limitation of this technique is the the number of clusters must be known in advance. Furthermore the "curse of dimensionality" effects on a fuzzy notion of similarity when multiple variables are considered. Instance based learning methodologies suggest the inheritance of fitness among candidates. These approximation methods seem rather coarse grained and have been found not to work well for mutli-objective problems [12].

In this study, we will focus on surrogates, as the fitness function can be gradually learned and therefore they can provide a finer grained model of the objective space. Popular techniques under this category are neural networks and statistical learning. Neural networks can be trained to approximate every function, however the resulting weights of the trained network are difficult to interpret while finding an appropriate network topology/size is laborious. The most promising technique to devise surrogate models in evolutionary computation, appears to be statistical learning. The challenge in devising such models is fitting statistical functions to observed distributions of the fitness behaviour. In the category of statistical learning, the use of Gaussian Processes (GPs) is prominent [13]. GPs specify a probabilistic model, constructed such that the likelihood of the function value given a set of observed data is maximized. We gravitate towards the use of GPs for online fitness predictions because they are simple to construct, provide a measure of uncertainty for the predictions and support online addition of observed data without changing

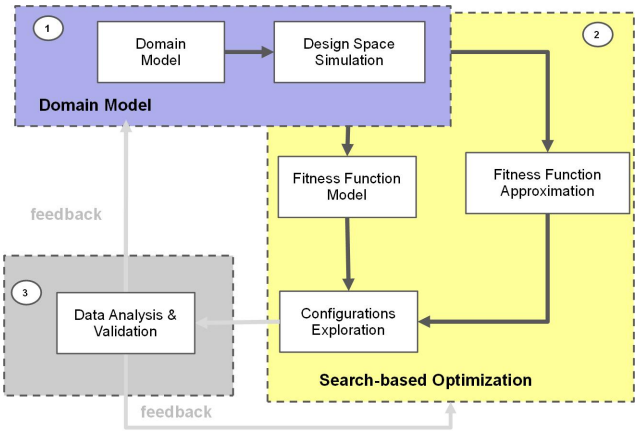


Fig. 2: Overall Methodology.

the model's parameters.

V. Methodology

Figure 5 shows our overall methodology. Each step is analysed in the following.

Assumptions

A prerequisite for our approach is that a software cloud monitoring tool exists to provide notifications of changes in the cloud environment and therefore triggers to call the optimisation framework. Furthermore, when new data points become available to the surrogate model (see Task 4) we will assume that the general form of the function described by the existing dataset is approximately the same as the one described by the expanded dataset.

Task 1. Domain Model

To solve RQ1 we have proposed a meta-model that formalises the achievable search space of all possible configurations [14]. The meta-model comprises information regarding the structure of a cloud datacenter configuration as well as its degrees of freedom. Conformance to the meta-model ensures that adaptations of architectures will lead to meaningful alternatives. In the following, we present the degrees of freedom we have identified so far and which can be exploited by our approach.

- ***PM Activate:*** The used hardware has an effect on many quality attributes. More active servers can make the datacenter perform better.
- ***PM Deactivate:*** On the other hand, there is a trade-off between quality attributes as speed, reliability and cost of the datacenter. Deactivation of low utilised servers provide a way to negotiate such trade-offs.
- ***VM Resize:*** The capabilities of a VM (e.g., virtual memory/CPU) can be resized. For example a virtual machine may receive a bigger CPU slice from its host to enforce its processing power.
- ***VM Create:*** New VMs can be instantiated to exploit more processing capacity and increase parallelism.
- ***VM Destroy:*** A VM may be destroyed to release its reserved resources towards saving costs.
- ***Live Migration:*** VM migration may be used to alleviate resource contention issues from overloaded servers or redistribute the VMs to release idle resources.

The different characteristics (e.g, severity, frequency) of possible environment changes leads us to believe that supporting adaptation actions of different costs and effects on the architecture is useful. For example a cheaper but moderate VM Resize can apply to rapid workload changes. An expensive but more aggressive VM migration could be more suitable in the case of e.g., a changed technical environment where new servers have become available and a new configuration is needed to cost-optimally balance the execution of services between the old and new infrastructure.

Having a formal description of the architecture design space (i.e., set of all feasible configurations), improved solutions can be automatically explored: First, stakeholders may define constraints for the design space. Second, the possible degrees of freedom of the infrastructure are identified. The third step runs an optimisation tool that reviews and varies the architecture along the most suitable degrees of freedom. The output is a set of improved solutions.

Task 2. Cloud Datacenter Simulation

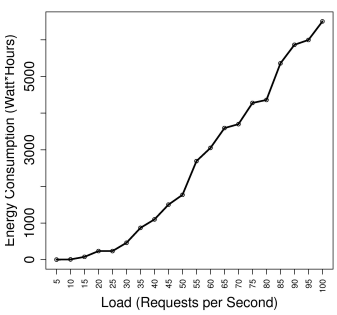
We simulate configurations using CloudSim[1] to represent our design space. Our current experimental setting comprises 150 PMs (HP ProLiant ML110 G4[2] and G5[3]) and 150 VMs. To simulate the web workload of our use case, requests for resources arrive per tier (application, business and database) according to a Poisson process, while requests size follows a Pareto distribution based on [15]. We vary the arrival rate with a step of 5 to generate $\{5 - 100\}$ requests per second. Each experiment runs for the time period $\{t_0, t_{finish}\}$, where $t_0 = 0$ and t_{finish} is the time needed for the datacenter to serve the incoming load. We repeat each experiment for 10 times and collect the average values. For each arrival rate we calculate the average (a) service's response time (b) datacenter energy consumption, and (c) datacenter CPU utilisation ratio.

Figure 3a shows that total energy consumption of the datacenter increases with workload. This can be explained from the CloudSim energy model, which is defined as a function of PM's power consumption model (for detail see footnotes 2 and 3) and CPU utilisation over time. Bigger workloads cause heavier CPU utilisation and require more time to complete execution, therefore energy consumption increases. Similarly, Figure 3b shows that the average service response time also increases with workload. VMs use a static resources (i.e, CPU, RAM, bandwidth) slice to execute the incoming requests. Therefore, as the workload size increases more requests are queued and response times increase. Finally, Figure 3c indicates that the average utilisation ratio of the datacenter remains $\sim 20\%$ of its total capacity regardless the workload size. Due to lack of infrastructure flexibility VMs and PMs cannot rebalance their capabilities to scale up or down to the actual demand.

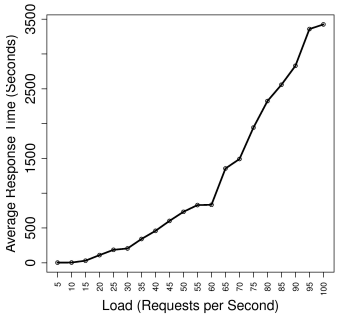
[1] http://www.cloudbus.org/cloudsim/

[2] http://www.spec.org/power_ssj2008/results/res2011q1/power_ssj2008-20110127-00342.html

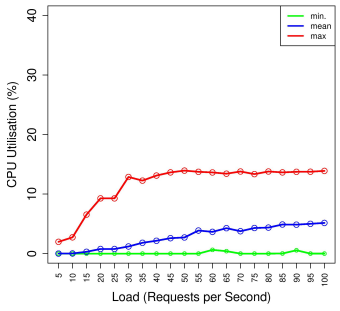
[3] http://www.spec.org/power_ssj2008/results/res2011q1/power_ssj2008-20110124-00339.html



(a) Energy Consumption



(b) Response Time



(c) Utilisation Rate

Fig. 3: Experimental results for the three metrics of interest.

Overall, our initial simulation experiments depict an over-provisioned datacenter where energy costs and response times sharply increase with load. Resources usage does not follow the actual demand; some servers are over-utilised while others remain idle wasting energy without improving performance. With our proposed methodology we envision to contribute to this picture by achieving runtime optimisation actions that follow workload variations towards limiting energy consumption within predefined cost thresholds and honouring runtime SLAs regardless the workload size.

Task 3. Multi-Objective Optimisation Problem

Let c be an architecture configuration in the space of all possible configurations D. Let q be a quality criterion and a fitness function $f_q(c)$ that denotes the quality of $c \in D$ for the criterion q. The optimisation problem can be formulated as follows for a set of quality criteria $\{q_1, ..., q_n\}$

$$min. \left[f_{q1}(c), ..., f_{qn}(c)\right], c \in D \quad (1)$$

Currently we study the quality criteria of services performance measured in average response time (*seconds*) and total datacenter energy consumption (*Watt-hours*). Figure 4 shows the optimisation process model of our method. The method presented is exemplary for our current realisation with NSGA-II evolutionary algorithm [16] as implemented in MOEA framework[4].

The optimisation process takes as input an initial configuration and generates a random set of candidates. In the

[4]http://www.moeaframework.org/

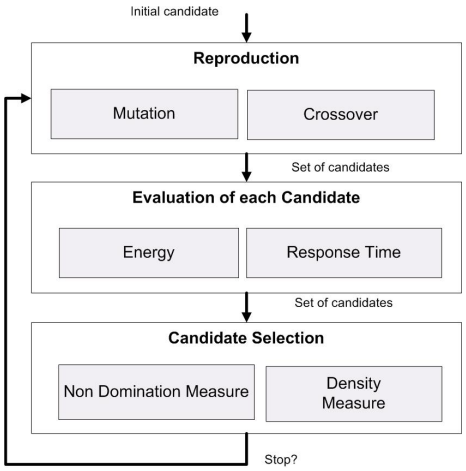


Fig. 4: Optimisation Algorithm.

second step, evolutionary algorithms iteratively search for better solutions, by applying the main steps of (a) ***Reproduction:*** "Mutation" and "crossover" operators modify the set of architecture candidates to tune their quality. With mutation one design option is varied along the formalised degrees of freedom. With crossover parts of two design options are merged into a new one. (b) ***Evaluation:*** The quality of each candidate is assessed against the achieved average response time and energy consumption. Here, we apply a full simulation of each perspective candidate to gather estimations for energy consumption and average services response time. (c) ***Selection:*** The most promising individuals are selected to form a new generation of candidates. Selection criteria include both a measure of non-domination indicating how close is the candidate to the true optimum and a measure of diversity indicating how uniform is the distribution of the solutions in the objective space. Over several iterations, the population will converge towards Pareto-optimal solutions, probably superior than the initial candidate.

Task 4. Surrogate Model

This task will devise a Gaussian processes (GP) surrogate model to learn online the fitness function of the optimisation process based on a set of sampled evaluated data points. GP modelling relates each observed value y with the unknown underlying fitness function $f(x)$ through a Gaussian noise model as follows:

$$y = f(x) + \mathcal{N}(0, \sigma^2) \quad (2)$$

Let us assume that evaluating f at M data points $X_M = [X_1, .., X_M]$ in the decision space, has yielded the fitness function values $Y_M = [Y_1, .., Y_M]$. The modelling task is to predict the unknown function value Y_{M+1} at a new data point X_{M+1} given by the conditional probability $p(Y_{M+1}|X_{M+1})$. The intended contribution of this step is to increase the speed of the optimisation convergence because we replace during evaluation, the highly time-consuming step of full candidates simulation with an estimation of conditional probability.

As shown in Figure 5, an environment change (e.g., workload spike) in the cloud datacenter will trigger the optimiser, to renegotiate the new best trade-off infrastructure architectures. Initially the optimiser will operate using the expensive fitness

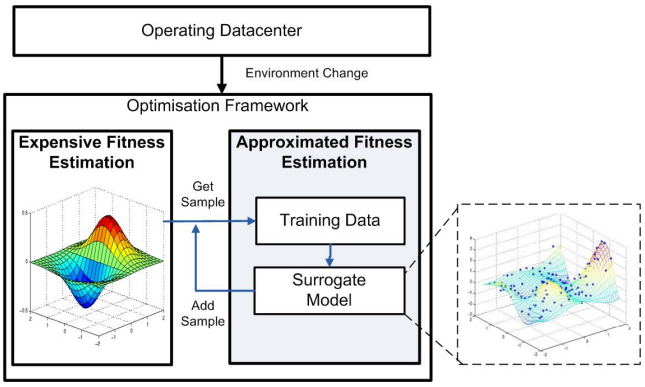


Fig. 5: Surrogate-based Optimisation.

model in order to gather an initial set of training data points. This dataset will calibrate our surrogate model. New calls to the optimiser will be able to use the surrogate model after its initial calibration. For each prediction of the fitness function values Y_{M+i} the surrogate model provides an error bar captured in the form of the standard deviation σ.

The quantity of the sampled training data is inverse analogous to the degree of uncertainty in the model's predictions. Based on this observation, a new call to the expensive fitness function can expand the initial training dataset towards reducing the error bar of each fitness prediction. A major challenge in this task is to establish a useful sampling methodology to minimise the risks of under-fitting and over-fitting the model. In other words, we must achieve a balance between calls to the expensive fitness function and to the approximate model, to ensure that our GP accesses a uniformly distributed set of training data that balances exploitation of already observed samples and exploration of so far unobserved values of the objective space.

Task 5. Validation

Our last target is to examine the efficiency of our proposed approach towards providing real-time infrastructure optimisations considering a set of configurable quality criteria $\{q_1, ..., q_n\}$ by simulation-based experiments. To this end, we will first compare the expensive optimisation procedure described in Task 2 with random search, to ensure that a possible success of the algorithm is not the result of an easy problem [17]. Then we will investigate the accuracy of the surrogate-based optimisation compared to the output of the expensive optimisation. The stochastic nature of evolutionary algorithms make comparisons between them challenging. If we apply the same algorithm several times to the same problem, each time a different Pareto set of solutions may be returned. Randomized algorithms can experience complex and high variance probability distributions. Therefore, to obtain reliable conclusions about the performance of the stochastic algorithms, we must use more powerful statistical testing methodologies (i.e., Mann-Whitney U) to ensure that there is enough empirical evidence to claim difference between the compared algorithms.

VI. Conclusion

Real time infrastructure adaptations is a key factor towards achieving elasticity in cloud datacenter structures. This PhD proposes a methodology for automated, runtime infrastructure optimisations in dynamic cloud environments. We expect our study to contribute to the academic community by providing analysis of dynamic cloud datacenter architectures and defining a compact methodology to negotiate at runtime inherent configuration design trade-offs such as performance versus cost and overhead of adaptations versus their benefit.

Acknowledgement

This work has been supported by the European FP7 Marie Curie Initial Training Network "RELATE" (Grant Agreement No. 264840).

References

[1] M. Harman, K. Lakhotia, J. Singer, D. White, and Y. Shin, "Cloud engineering is search based software engineering too," *In Journal of Systems and Software*, 2013.

[2] H. A. Simon, "Invariants of Human Behavior," *Autobiographies*, 2009.

[3] G. Jung, M. A. Hiltunen, K. R. Joshi, R. D. Schlichting, and C. Pu, "Mistral: Dynamically Managing Power, Performance, and Adaptation Cost in Cloud Infrastructures," *In International Conference on Distributed Computing Systems*, 2010.

[4] A. Verma, P. Ahuja, and A. Neogi, "pMapper: power and migration cost aware application placement in virtualized systems," in *Proceedings International Conference on Middleware*, 2008.

[5] C. C. Keir, C. Clark, K. Fraser, S. H, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live Migration of Virtual Machines," in *NSDI Proceedings*, 2005.

[6] G. Khanna, K. Beaty, G. Kar, and A. Kochut, "Application Performance Management in Virtualized Server Environments," in *the Network Operations and Management Symposium*, 2006.

[7] B. Li, J. Li, J. Huai, T. Wo, Q. Li, and L. Zhong, "EnaCloud: An Energy-Saving Application Live Placement Approach for Cloud Computing Environments," in *Proceedings of International Conference on Cloud Computing*, 2009.

[8] N. Huber, F. Brosig, and S. Kounev, "Model-based self-adaptive resource allocation in virtualized environments," in *Proceedings of Software Engineering for Adaptive and Self-Managing Systems*, 2011.

[9] B. Addis, D. Ardagna, B. Panicucci, and L. Zhang, "Autonomic Management of Cloud Service Centers with Availability Guarantees," in *Proceedings of International Conference on Cloud Computing*, 2010.

[10] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," *Cluster Computing*, 2009.

[11] F. Hermenier, X. Lorca, J.-M. Menaud, G. Muller, and J. Lawall, "Entropy: a consolidation manager for clusters," in *In the Proceedings of international conference on Virtual execution environments*, 2009.

[12] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments-a survey," *Transactions on Evolutionary Computation*, 2005.

[13] D. Buche, N. N. Schraudolph, and P. Koumoutsakos, "Accelerating Evolutionary Algorithms with Gaussian Process Fitness Function Models," *In Transactions on Systems, Man and Cybernetics*, 2004.

[14] K. Chatziprimou, K. Lano, and S. Zschaler, "Towards A Meta-Model of the Cloud Computing Resource Landscape," 2013.

[15] P. Barford and M. Crovella, "Generating representative Web workloads for network and server performance evaluation," in *Proceedings of ACM Sigmetrics*, 1998.

[16] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II," *In Transactions on Evolutionary Computation*, 2000.

[17] A. Arcuri and L. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," in *Proceedings International Conference on Software Engineering*, 2011.