Daleel: Simplifying Cloud Instance Selection Using Machine Learning

Faiza Samreen, Yehia Elkhatib, Matthew Rowe, and Gordon S. Blair
School of Computing and Communications, Lancaster University, United Kingdom
Email: {i.lastname}@lancaster.ac.uk

***Abstract*—Decision making in cloud environments is quite challenging due to the diversity in service offerings and pricing models, especially considering that the cloud market is an incredibly fast moving one. In addition, there are no hard and fast rules; each customer has a specific set of constraints (*e.g.* budget) and application requirements (*e.g.* minimum computational resources). Machine learning can help address some of these complicated decisions by carrying out customer-specific analytics to determine the most suitable instance type(s) and the most opportune time for starting and/or migrating instances. In this paper, we employ machine learning techniques to develop an adaptive deployment policy tailored for each customer, providing an optimal match between their demands and the available cloud service offerings. We provide an experimental study based on extensive set of job executions over a major public cloud infrastructure.**

1. Introduction

The Infrastructure-as-a-Service (IaaS) ecosystem is evolving so rapidly that it is becoming increasingly difficult to select the best resources to use. It is a composite decision that every customer is faced with:

- Which provider should she choose?
- What instance type(s) would provide her with the cost:performance ratio that suits her needs?
- Does the time or day at which she requests these resources affect how her application runs?

A customer has to choose between dozens of different instance types, as illustrated in Figure 1. Moreover, the answers to the above questions are highly subjective; each customer application needs careful consideration of its requirements against the various market offerings. Further complications are manifested due to the disparate pricing models adopted by different cloud service providers (CSPs). As such, a customer entering the cloud market is overwhelmed with a host of difficult questions without much of a support system for such decision making.

We argue that we can help answer many of the aforementioned questions about cloud infrastructure setup in a systematic and evidence-based manner. This is not only to assist customers entering the market, but also to provide guidance to those wishing to migrate deployments between CSPs to enhance Quality of Service (QoS), reduce cost, or to honour other non-functional requirements (*e.g.* legislation, disaster recovery, business continuity).

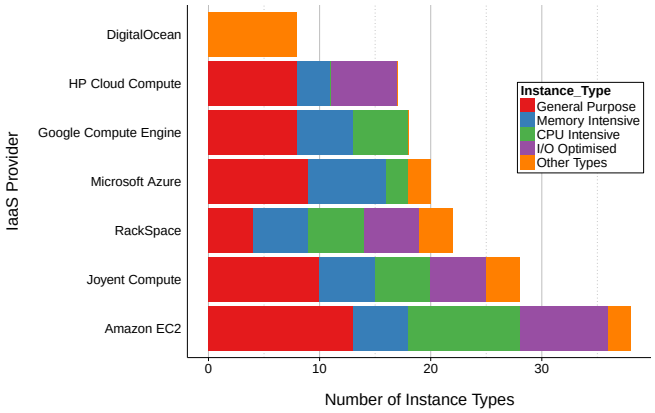

Figure 1. The number of instance types offered by the major IaaS vendors, as of 25th of August 2015.

In this paper we present *Daleel*, a multi-criteria adaptive decision making framework that is developed to find the optimal IaaS deployment strategy. In this paper, we take a first step by focusing on one CSP in order to answer the question: *Which instance type and what time(s) are best for a given customer application?*. After gathering the profiling evidence, we employ machine learning techniques to gain insight into the expected performance of an application on the calibrated CSP. Multivariate polynomial regression is used for predicting application behaviour on certain IaaS configuration.

Our contributions are as follows:

- The Daleel framework that supports adaptive decision making in IaaS environments. We consider two QoS attributes as criteria: instance price and application execution time.
- An extensive analysis of variability of Amazon EC2, a leading IaaS CSP. We use more than 5,000 application runs for this purpose.
- Utilisation of different machine learning techniques to evaluate Daleel's ability to predict application execution time on the EC2 cloud.

The rest of the paper is organised as follows. §2 reviews related work. §3 details the Daleel architecture. §4 presents the outcomes of a large-scale study into the variability of IaaS offerings. §5 evaluates our learning approach using data collected from EC2 deployments. §6 concludes and points out avenues of future work.

2. Related Work

2.1. Application Management Frameworks

A number of frameworks have been developed by industry and open source communities to act as an intermediary between cloud customers and providers. These, sometimes referred to as *cloud brokers*, carry out some task on behalf of the customer such as arbitrage, aggregation and integration. We classify such frameworks as either hosted or deployable. *Hosted* services are externally managed by third-party stakeholders and do not provide information about how the application is being provisioned across the cloud. Examples include RightScale cloud portfolio management[1], enStratus[2], xStream[3] and CliQr[4]. In contrast, *deployable* services rely on open source solutions that could be operated internally by a corporation or externally as a grey-box integrated service. Brooklyn[5], Scalr[6], Standing Cloud[7] and Aelous[8] are some examples.

The solutions mentioned thus far tackle interoperability to reduce application deployment, but do not support adaptive decision making. This feature is still largely lacking from cloud brokerage solutions, although some efforts have started to surface in the wider cloud computing community, *e.g.* STRATOS [1], CELAR[9], MODACloud[10], Cloud4SOA [2], mOSAIC[11], ARTIST[12], Broker@Cloud[13], and PlanForCloud[14]. We conjecture that there is still a long way to go in terms of providing dynamic decision making that can effectively optimise to the specific functional and non-functional requirements on a per-application basis.

2.2. Machine Learning

Machine learning can contribute immensely by taking appropriate decisions to cater to specific application requirements. Machine Learning has proved its potential for producing prediction and optimisation solutions in various fields [3], [4]. It has also been applied in cloud computing towards resource scaling [5], forecasting [6], and dynamic resource provisioning [7], [8], [9]. We aim to apply a similar methodology but for the benefit of cloud customers selecting between different IaaS resources.

1. http://www.rightscale.com/cloud-portfolio-management/benefits
2. http://www.enstratius.com/
3. http://www.virtustream.com/solutions
4. http://www.cliqr.com
5. https://brooklyn.incubator.apache.org/
6. http://www.scalr.com/
7. http://www.standingcloud.com/
8. http://www.aeolus-project.org/
9. http://www.celarcloud.eu/
10. http://www.modaclouds.eu/
11. http://www.mosaic-cloud.eu/
12. http://www.artist-project.eu/
13. http://www.broker-cloud.eu/
14. http://www.planforcloud.com/

3. Daleel

Daleel[15] is a multi-criteria adaptive decision making framework. It equips a cloud customer with evidence-based knowledge of the IaaS setup specification that is optimal for their particular application.

3.1. Architecture

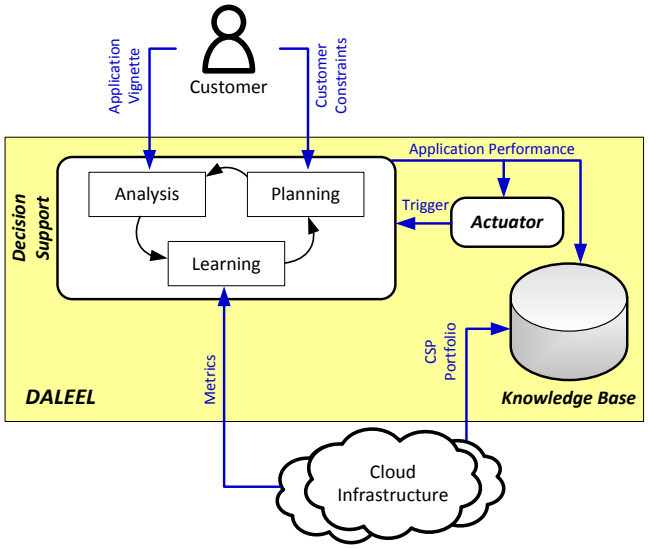

Figure 2. The Daleel Architecture.

Daleel's architecture (depicted in Figure 2) consists of three main modules: *Decision Support*, *Actuator*, and *Knowledge Base*. The **Decision Support** module is at the heart of the Daleel architecture and it relies on a three phase process that continuously operates throughout the application life cycle to predict application performance. These phases are: *Analysis*, *Learning*, and *Planning*. They carry out different but complimentary operations to acquire deep knowledge of the available cloud deployment options and how suitable they are to a given application. The **Actuator** triggers the Decision Support module into operation. The **Knowledge Base** holds data collected by the Decision Support module.

The application vignette is a short set of key-value pairs provided by the customer that serve as a high level description of the application requirements. The customer constraints include the customer's functional and non-functional requirements such as minimum QoS, availability, location, and budget. A CSP's portfolio contains data that we obtain (through APIs and web scraping) on their resource provisioning levels, resource metadata, and pricing models.

We now describe how the three Decision Support phases and the Actuator module work.

15. Daleel means 'guide' in Urdu.

3.2. Analysis Phase

The first stage comprises of a profiling procedure that is based on time series analysis. Application profiling is an effective way of tracking application behaviour under different deployment setups. This can be carried out *live* on shared cloud infrastructures (whether public or private), or *offline* in a completely controlled and isolated virtual environment. The obtained traces record different metrics such as CPU utilisation, memory utilisation, paging and caching information etc. Together these constitute the application profile that can be used to predict deployment options that can suit the application and customer requirements.

Aggregating different application profiles builds up the Knowledge Base with information about application descriptions and their behaviour on different deployment setups. This is used to infer performance of a not-yet-profiled application based on its 'vignette' (*i.e.* general description).

3.3. Learning Phase

The second phase in the Decision Support module is the Learning phase. It comprises of a prediction procedure that receives from the Analysis phase the following information: performance traces, cloud resources portfolio, and the application vignette. A critical task is to derive a prediction model for the cloud provider in question, which shows the QoS variations and offered services in order to help the following phase (*i.e.* Planning) in achieving optimal deployment that caters to the customer constraints (functional and non-functional requirements).

The Learning phase aims to accurately predict the cost of application execution, in terms of performance and virtual machine (VM) price. It also aims to achieve a better understanding of the correlation between the predictors and the response in order to infer some relationship for future prediction. These are quite difficult aims for which different machine learning techniques are explored as no one technique is considered to be the best for all data sets. In this study, different regression methods were employed as a prediction function. The response variable in our case study is application execution time that involves a continuous quantitative output value.

3.4. Planning Phase

The third phase takes input from the Learning phase in the form of a prediction model which can generate a vector output based on the input requirements of the customer. The Planning logic is designed to support a *multi-criteria decision making* problem where a set of vectors describing the performance is the Learning outcome. For the purposes of this study, we are targeting two QoS attributes as our criteria, namely VM price and application performance (execution time). Various methods are being used by multi-criteria decision making such as weighted sum [10], weighted product [11], VIKOR [12], PROMETHEE [13], and more. We intend for our decision making support to include such multi-criteria techniques while considering more than two QoS attributes.

3.5. The Actuator

The Actuator triggers the Decision Support module into operation at different times. This could be based on thresholds set according to the customer constraints on application QoS, application load, or Knowledge Base information (*e.g.* change in a provider's portfolio). Such triggers will launch new Analysis and Learning cycles, or will activate the Planning logic to begin migration to a new cloud infrastructure. Migration between different cloud infrastructures is a big challenge in its own right and is outside the boundaries of this work. However, the Planning logic could easily be extended to incorporate migration methods, *e.g.* [14].

4. Analysis of Variability in IaaS Offerings

We hypothesise that selecting specifications of a cloud-based infrastructure is not an easy or straight-forward task, especially due to the fact that there is considerable amount of performance variability at any service provisioning tier. Our initial step is to gather enough information to analyse such variability. We achieve this through extensive experiments over EC2's public IaaS offerings. In this section we explain the experimental details of our study along with profiling procedure and the performance variability analysis. §5 will detail about model development and learning evaluation based on profiled data.

4.1. Methodology

The overall objective of conducting this evaluation is to find the performance variations on different node configurations at different times of the day. This experiment is conducted on Amazon Elastic Cloud Compute (EC2). EC2 is the leading CSP with a 57% share of the IaaS market [15]. We run over different instance types, and throughout the seven days of the week to investigate temporal variations.

4.1.1. Infrastructure. All instances used were 64-bit Ubuntu Linux of different capacities as detailed in Table 1. Note that 'vCPU' indicates the number of virtual cores assigned to a VM. An 'ECU' refers to an *EC2 Compute Unit*; Amazon does not advise about how an ECU relates to physical processing speed; it only assures that it is a standard unit across its IaaS offerings[16]. 'Price' refers to the hourly charge for running a VM of the referenced instance type.

Amazon provides differentiated series of instance types, catering to different application needs (*e.g.* compute-intensive, memory intensive, I/O-intensive, etc.). Each series contains a number of instance types offering different setups of computational resources. We targeted the General Purpose series T2 and M3 as well as the Compute Optimised series C4 in order to evaluate varying combinations of

16. http://aws.amazon.com/ec2/faqs/

TABLE 1. THE COMPUTATIONAL SPECIFICATION OF EC2 INSTANCES.

Series	Node	vCPU	ECU	RAM (GB)	Storage (GB)	Price ($/h)
T2 (General Purpose)	t2.small	1	Var.	2	20	0.026
	t2.medium	2	Var.	4	20	0.052
M3 (General Purpose)	m3.medium	1	3	3.75	4(S)	0.070
	m3.large	2	6.5	7.5	32(S)	0.140
C4 (Compute Optimised)	c4.large	2	8	3.75	20	0.116
	c4.xlarge	4	16	7.5	20	0.232

resource capacity over a relatively wide price range. Only on-demand instances were used for this experiment. These have no long term commitment and are charged on a pay-as-you-go basis at an hourly rate. All instances were chosen to be located in eu-west-1 availability zone, hosted in Ireland.

We are not aware of how EC2 virtual cores are pinned to physical cores. Amazon EC2 uses the Xen hypervisor to host the VM instances but do not provide the details of scheduling algorithms used by the hypervisor. From running our experiments, we could not find any firm details for parallel workload and so are not aware of the interference effects. This, however, is not our focus.

4.1.2. Application & Execution. Our use case application was VARD [16], a tool designed to detect and tag spelling variations in historical text, particularly in Early Modern English. The output is aimed to improve the accuracy of other corpus analysis solutions. VARD is a single threaded application that is highly memory intensive. It holds in memory a representation of the full text, as well as various dictionaries that are used for normalising spelling variations. Experiments were continuously repeated using a fixed set of input texts over a period of seven days with a delay of ten minutes in between each pair of runs. The Linux tools `vmstat`, `glances` and `sysstat` were used to continuously monitor resource utilisation.

4.2. Variation Due to Instance Type

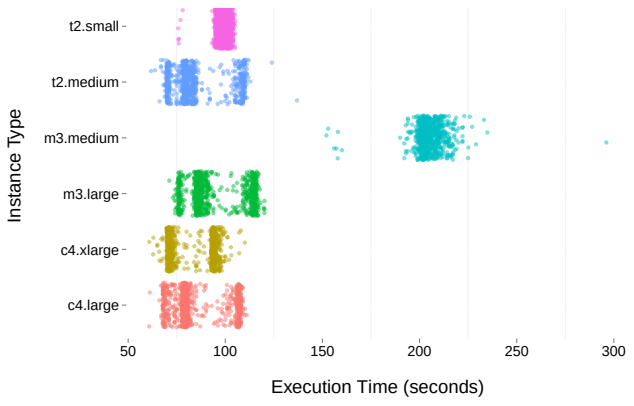

Figure 3. Application execution time over different cloud instance types.

We investigate performance of running VARD on VMs of different instance types. The results are summarised in Figure 3 where every dot represents the mean execution time of one run. Shorter execution times reflect a lower hourly rate over a full workload. There are a number of interesting observations from these results.

We observe that, contrary to intuition, m3.medium (a memory-rich instance) is of consistently poor performance. We also observe that c4.large surpasses both m3.medium and m3.large in performance. In fact it is on par with c4.xlarge, which has twice both in specification and cost.

Overall, the T2 series offers by far the best value for money. A possible explanation is the *CPU Credits* scheme which enables customers to collect credits for idle instances and later spend them when full CPU utilisation is needed. CPU Credits are offered only on T2 series. Hence, T2 instances are good for applications that do not consistently fully use the CPU. However, it also means that there is a degree of uncertainty associated with an application's performance that depends on its idle time.

4.3. Variation Due to Time

We now turn our attention to uncertainty in application performance due to the time at which they are executed. This is depicted by the box-plots in Figure 4.

The T2 series offer the least RAM, but exhibit the least variance in performance between the different days of the week. m3.medium VMs display a predictable, albeit quite high, application execution time. The median and quartiles show very little variation across the days of the week. m3.large also offers quite predictable performance across the week, with a narrow first quartile which is favourable.

The two C4 instance types portray contrasting performances. c4.large is rather predictable with a steady median and right skewness (*i.e.* a very narrow first quartile). On the other hand, dispersion in the c4.xlarge instances is more towards the high end of application execution time with a median that is less regular: less left skewness is observed on Wednesday, Saturday and Sunday.

This could be down to different reasons such as demand from other users, the provider's resource sharing algorithms, and the provider's energy efficiency policy. These are difficult attributes for us to ascertain from the outside. Nevertheless, we detect certain regularities that helps us determine the predictability of application performance at different time.

4.4. Lessons Learned

We investigated how variable the performance obtained from different IaaS settings could be, making the execution of a simple application rather uncertain. We demonstrated that public IaaS offerings are to a great extent black boxes. First, selecting instance types solely based on their advertised resource specifications is not necessarily optimal. Second, selecting which day of the week to run an application could result in significant variation in performance.

This variability serves as our motivation to equip users with some certainty when consuming IaaS resources.

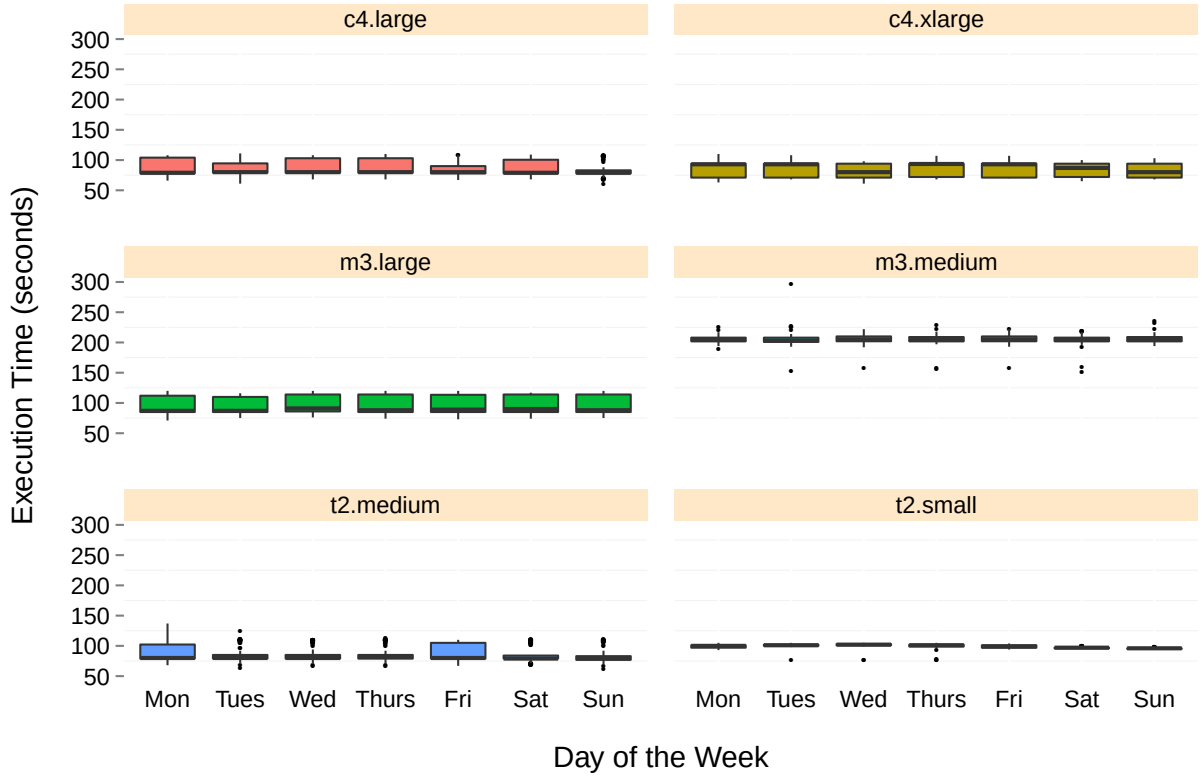

Figure 4. Dispersion of application execution time during all days of the week on different EC2 instance types. Notice that all graphs have the same y-axis range apart from m3medium.

5. Learning Evaluation

We now apply different machine learning algorithms to be able to predict the best IaaS deployment setup for a certain application. We again use the VARD application (see Section 4.1.2) as a use case, with the goal being to predict the optimal resources and most opportune time for starting an EC2 instance to execute VARD by predicting the execution time. We first describe how we look for and assess the best models, then we detail the outcomes of our learning investigation.

5.1. Model Development and Evaluation Method

The core technique of our methodology that can effectively predict execution time is based on polynomial regression. Polynomial regression is an approach of non-linear fit to data [17]. It extends the linear model by adding additional predictors that are obtained by raising each of the original predictors to a power. For example, a cubic regression has three variables: X, X^2, X^3. The basis functions for polynomial regression are $b_j(x_i) = x_{ij}$. We take application execution time as a response variable, whilst a list of other variables as candidate predictors: RAM, vCPU, processor speed, hypervisor, storage, day, time, application input parameters, etc.

Considering both prediction and inference based learning techniques, we follow the procedure outlined below in order to get a robust model that can accurately predict the response using the predictors.

1) Split the data into two sets: a training set to be used for learning, and a test sample for assessment and model evaluation. For current evaluation we split the data set into training and test set with a ratio of 57% and 43% respectively.
2) Train the model on the training set.
3) Assess the accuracy of the model using resampling methods (*e.g.* cross validation and bootstrapping [18]), these methods are employed on the training set for model assessment. Resampling methods repeatedly draw samples from the training set and refit the model on each sample to get additional information about the fitted model's performance such as variability estimates of regression fit. Cross validation is one of the widely used resampling methods for model selection. We used the k-fold cross validation method, computed by averaging the Mean Squared Error (MSE) for k-folds over the training sample using the formula:

$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^{k} MSE_i$$

where $k = 10$ in our case. The MSE serves as a risk function for an estimator to measure the average of the squares of the error that is basically a difference between the estimator and estimated value [4]. It is

calculated using below equation where y is actual response value:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

4) Check goodness of model fit using statistical testing methods like p-value, R^2, RSE, and F-statistics. R^2 measures the proportion of variability in the response variable that is explainable by the predictors. The Residual Standard Error (RSE) shows the actual deviation of the response from predicted, and measures the lack of fit for a model. F-statistics (also referred to as *fixation indices*) is a measure to reject a null hypothesis and to show the overall significance of a model.

5.1.1. Polynomial Fit. The multivariate polynomial model is a special case of a basis function approach that we used in our learning model. The idea of using a basis function is to have a transformation that can be applied to a variable X: $b_1(X), b_2(X), \ldots b_k(X)$. Instead of a linear model fit in X, we fit the model:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 {x_i}^2 + \beta_3 {x_i}^3 + \cdots + \beta_d {x_i}^d + \epsilon_i$$

Basis functions are fixed and known [18], hence the least square approach can be used to estimate the unknown regression coefficients in the model above. The least square fit approach for coefficient estimates indicates that all of the inference tools for linear regression, such as standard errors of the coefficient estimates and F-statistics to check overall significance of model, can be used in this setting as well. A robust polynomial model is build using profiling results. This model is an attempt to predict execution time using two significant predictors RAM and vCPU as well as an additional one: day of the week. This third predictor cannot describe the underlying distribution function on its own; instead it presents a meaningful outcome in a combinatorial way. This polynomial regression based formula takes the following form with one cubic and bivariate quadratic polynomial:

$$\begin{aligned} F(x) = \beta_{01} + \beta_{11}x + \beta_{21}x^2 + \\ \beta_{02} + \beta_{12}x + \beta_{22}x^2 + \beta_{32}x^3 + \\ \beta_{03} + \beta_{13}x + \beta_{23}x^2 + \beta_{33}x^3 \end{aligned}$$

This model is considered a successful attempt towards prediction at a fine grained level. It has the lowest MSE compared to other models evaluated in next section. The planning phase takes this model as an input along with substantial details of customer constrains and outputs the suitable configuration based on the metric calculated by the planner.

5.2. Model Accuracy Analysis

To evaluate the accuracy of our model we compared it with different learning models using the standard methods described in subsection 5.1. Due to the lack of previous models, we used other learning techniques as baseline for comparison, namely linear regression, ridge regression and Lasso. The same dataset and methodology were used to extract and evaluate the results.

5.2.1. Baseline Models. Linear models are relatively simple to implement and can provide good interpretation and inference. For accurate coefficient estimates, it uses the least square criteria [3]. The standard linear model is expressed as follows:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i \quad (1)$$

The linear least squares fitting procedure (Equation 1) estimates $\beta_0, \beta_1, \ldots, \beta_p$ using the value that minimises the residual sum of squares (RSS) as defined by:

$$RSS = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2$$

In extension to our assumption about linear regression we tried to fit the model containing all variables using a technique that shrinks the coefficient estimates towards zero. We used the two well-known regularisation techniques for shrinking regression coefficients: ridge regression (also known as *Tikhonov regularisation* [19]) and Least Absolute Shrinkage and Selection Operator (Lasso) [20].

Ridge regression is similar to least squares but minimises the coefficient estimates with a slightly different quantity of λ [18]. The ridge regression coefficient estimates $\hat{\beta}^R$ as the values that minimises

$$RSS + \lambda\sum_{j=1}^{p}\beta_j^2 = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}\beta_j^2$$

$\lambda \geq 0$ is a tuning parameter that controls the relative impact of the least square and shrinkage penalty on the regression coefficient estimates. When $\lambda = 0$, the penalty term has no effect and estimates are least square. However, as $\lambda \to \infty$ the shrinkage penalty grows and the coefficient estimates approaches zero.

Ridge regression includes all the variables as P predictors in the final model. Highest value of λ can reduce the coefficient value but cannot exclude any variable from the resulting model. On the other hand, Lasso overcomes this disadvantage by forcing some of the coefficient estimates to be equal to zero especially when the λ value is large enough [18], as such:

$$RSS + \lambda\sum_{j=1}^{p}|\beta_j| = \text{`}\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}|\beta_j|$$

In statistical terms, Lasso uses an l_1 penalty while ridge uses an l_2 penalty. The l_1 norm of a coefficient vector β is

$$||\beta||_1 = \sum|\beta_j|$$

For the ridge and Lasso regression model fits, we chose a range of λ values from $\lambda = 10^{10}$ to $\lambda = 10^{-2}$ in order to evaluate all scenarios starting from the null hypothesis (that contains only the intercept term) to the least square fit, respectively.

5.2.2. Diagnostic Assessment. We now asses the accuracy of the models, as summarised in Table 2.

The norm values assessment for Lasso and ridge regression models indicate that none of the λ values reduced the MSE. In fact, the best λ values (*i.e.* the ones that have minimum MSE, namely $\lambda = 2.30$ for ridge and $\lambda = 0.03$ for Lasso) have even higher MSE than that when the function is derived to the least square fit. The best λ value was figured out using cross validation technique.

Moving on to the other regression diagnostics (not suitable for ridge or Lasso), the R^2 statistic provides the proportion of variance explained using the predictor X and so it always takes a value between 0 and 1. The low R^2 value for linear regression indicates that this model did not explain much of the variability in the response; much less than half of it, in fact. On the other hand, the polynomial model captures more than 93% of data variability in terms of response prediction.

The high F-statistics value for the polynomial model indicates the significance of selected predictors and their relationship with the response variable. The validation set error rate is usually assessed using MSE especially in the case of quantitative response. The MSE values for ridge and Lasso are higher than that of the linear model. However, the same validation set MSE for polynomial fit is considerably smaller than the linear model. As with R^2, we observe a gross reduction for RSE in polynomial fit that estimates the standard deviation of error term which means there is less deviation of predicted response from the true regression line.

TABLE 2. ASSESSMENT OF THE MODELS OVER THE TRAINING DATASET.

Diagnostic	Model			
	Linear	Ridge	Lasso	Polynomial
MSE (10-fold CV)	1159.00	2312.69	2476.65	131.27
R^2	0.3741	–	–	0.9307
F-Statistics	298	–	–	5024
RSE	33.77	–	–	11.55

Furthermore, we can check the model visually by plotting the actual response from the test dataset against the predicted. If the model describes the structure of the data appropriately, then the estimated regression curve should be aligned with the data. This visualisation is shown in Figure 5. The red points denote the linear model fit, depicting considerably high deviation from the identity line. Ridge and Lasso display similar qualities and are not plotted for clarity. In contrast, the polynomial fit (blue points) is closely aligned to the identity line, proving a far superior prediction capability compared to the linear and other baseline models.

Finally, we assess the models on the test set (43% of the full EC2 data). The results confirm the above results, as summarised in Table 3.

5.3. Model Fitting Outcomes

The polynomial transformation has proved to be the best fit to the EC2 data as evaluated using different diagnostic

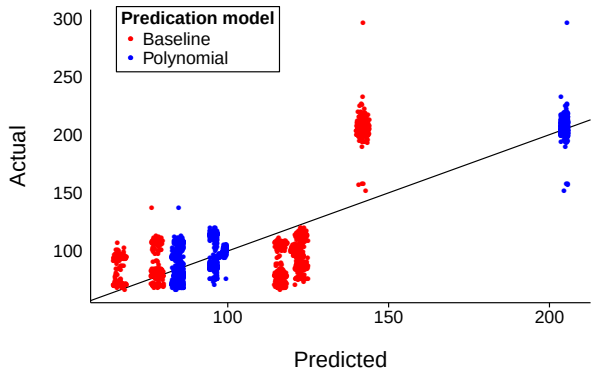

Figure 5. Predicted vs Actual values for the polynomial and the linear baseline model over the test dataset. Values closer to the identity line indicate better prediction performance.

TABLE 3. ASSESSMENT OF THE MODELS OVER THE TEST DATASET.

Diagnostic	Model			
	Linear	Ridge	Lasso	Polynomial
MSE	1183.00	1185.82	1162.80	129.84

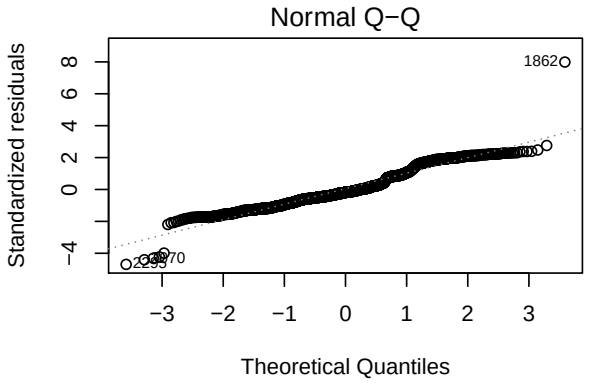

Figure 6. Q-Q plot of the polynomial model.

methods and plots. We now use residual plots to check for possible violations of our assumptions such as non-constant variance and non-normal distribution. If the two distributions are similar, then they would show a constant variance with normally distributed data. Q-Q plots are used for this purpose. In the Q-Q plots of our polynomial model shown in Figure 6, the points lie on the identity line indicating that the data indeed follows a normal distribution, validating our assumption. This also confirms that the non-linear transformation works well for our model fit.

An interesting finding is that the inclusion of an additional predictor (day of the week) has little but not significant improvement as reflected by R^2 in the polynomial model. Nonetheless, it does allow us to be able to predict the optimal deployment time at a day granularity level which is a good contribution to support decision making.

In summary, we explored the possibility of fitting the data with both linear and non-linear models. We found

that a non-linear transformation of the predictors is more suitable due to the non-linear association of data. Model assessment was done through cross-validation using the MSE which estimates the test errors associated with the learning method to evaluate its performance. We applied regression diagnostics to check the assumptions for linear regression with non-linear transformation of the predictors. Non-linear multivariate polynomial model outperformed the linear, ridge and Lasso models as indicated by different factors.

6. Conclusion

Customers are faced with a large array of choices for deploying their applications in the cloud. Supporting the customer decision making is an under-researched area. Furthermore, there is little understanding of how to implement adaptive decision making that can react to changes in context.

In this paper, we explored in depth the role of machine learning to provide such adaptive decision making. An overall architecture, called *Daleel*, was proposed (§3) and an empirical evaluation carried out to investigate how machine learning can support the key phases of analysis and learning in preparation for the subsequent planning. The analysis focuses on decision making around executing a particular application in a given cloud provider. Specifically, our empirical study focused on decision making around the selection of instance types for the execution of VARD, an application from the field of computational linguistics. The results show that:

a. The performance of instance types is often counter-intuitive and can vary significantly over time (§4).
b. Machine learning can be highly effective in making predictions based on performance data but care is required to select the right approach (§5).

Looking at this aspect in more detail, we explored the possibility of fitting the data with both linear and non-linear models. We found that a non-linear transformation of the predictors is more suitable. This is due to the inherent non-linear association of data; we found, through extensive experimentation, significant skewness in the performance of different EC2 instance types.

The results from this initial study are encouraging and we are convinced that machine earning has a central role to play in optimising cloud deployments. Our first avenue of future work is to consider other classes of application, including applications that vary in terms of being memory-intensive, processor-intensive or data-intensive and combinations thereof. The Knowledge Base in our Daleel architecture will be used for further investigation of application behaviour and resource utilisation patterns in order to predict based on different clusters of application types. Second, we plan to incorporate enhanced decision making methods, including consideration of multi-criteria decision making. Finally, extending the work to consider the management of cross-cloud environments including consideration of policies such as cloud-bursting in hybrid cloud infrastructures. Cloud brokerage is the key application area here, but it still is in its infancy especially in terms of practical experience.

References

[1] P. Pawluk, B. Simmons, M. Smit, M. Litoiu, and S. Mankovski, "Introducing STRATOS: A cloud broker service," in *Conference on Cloud Computing (CLOUD)*. IEEE, 2012, pp. 891–898.

[2] E. Kamateri, N. Loutas, D. Zeginis, J. Ahtes, F. D'Andria, S. Bocconi, P. Gouvas, G. Ledakis, F. Ravagli, O. Lobunets, and K. Tarabanis, *Cloud4SOA: A Semantic-Interoperability PaaS Solution for Multi-cloud Platform Management and Portability*, ser. LNCS. Springer, 2013, vol. 8135, ch. 6, pp. 64–78.

[3] T. Mitchell, *Machine Learning*. McGraw Hill, 1997.

[4] E. Alpaydin, *Introduction to Machine Learning*. MIT Press, 2014.

[5] J. Kupferman, J. Silverman, P. Jara, and J. Browne, "Scaling into the cloud," http://www.cs.ucsb.edu/~jbrowne/files/ScalingIntoTheClouds.pdf, University of California, Santa Barbara, Tech. Rep., 2009.

[6] E. Caron, F. Desprez, and A. Muresan, "Forecasting for grid and cloud computing on-demand resources based on pattern matching," in *Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2010, pp. 456–463.

[7] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 155–162, 2012.

[8] A. A. Bankole and S. A. Ajila, "Predicting cloud resource provisioning using machine learning techniques," in *Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE, May 2013, pp. 1–4.

[9] J. Gao, "Machine learning application for data center optimization," https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/42542.pdf, Google, Tech. Rep., 2013.

[10] R. T. Marler and J. Arora, "The weighted sum method for multi-objective optimization: new insights," *Structural and Multidisciplinary Optimization*, vol. 41, no. 6, pp. 853–862, 2010.

[11] J. R. S. C. Mateo, *Weighted Sum Method and Weighted Product Method*, ser. Green Energy and Technology. Springer London, 2012, ch. 4, pp. 19–22.

[12] H. Liao and Z. Xu, "A VIKOR-based method for hesitant fuzzy multi-criteria decision making," *Fuzzy Optimization and Decision Making*, vol. 12, no. 4, pp. 373–392, 2013.

[13] J.-P. Brans and B. Mareschal, "PROMETHEE methods," in *Multiple Criteria Decision Analysis: State of the Art Surveys*, ser. International Series in Operations Research & Management Science. Springer New York, 2005, vol. 78, ch. 5, pp. 163–186.

[14] J. Hadley, Y. Elkhatib, G. S. Blair, and U. Roedig, "Multibox: Lightweight containers for multi-cloud deployments," in *Embracing Global Computing in Emerging Economies (EGC2015)*, Feb 2015.

[15] RightScale, "2015 state of the cloud survey," http://assets.rightscale.com/uploads/pdfs/RightScale-2015-State-of-the-Cloud-Report.pdf, Tech. Rep., 2015.

[16] A. Baron and P. Rayson, "VARD2: A tool for dealing with spelling variation in historical corpora," in *Postgraduate conference in corpus linguistics*, May 2008.

[17] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning*, 7th ed. Springer, 2013.

[18] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: With Applications in R*, 4th ed. Springer, 2014.

[19] A. N. Tikhonov and V. Y. Arsenin, "Solutions of ill-posed problems," *WH Winston*, vol. 330, 1977.

[20] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.