Journal of Web Engineering, Vol. 17, No.3&4 (2018) 224-240

AN APPROACH FOR GUESSTIMATING THE DEPLOYMENT COST IN CLOUD INFRASTRUCTURES AT DESIGN PHASE IN WEB ENGINEERING

JUAN CARLOS PRECIADO, ROBERTO RODRIGUEZ-ECHEVERRIA, JOSÉ MARÍA CONEJERO, FERNANDO SÁNCHEZ-FIGUEROA AND ÁLVARO E. PRIETO

QUERCUS Software Engineering Group, School of Technology. University of Extremadura, Cáceres
{jcpreciado, rre, chemacm, fernando, aeprieto} @unex.es

Received July 27, 2017
Revised January 25, 2018

Nowadays, the total cost of cloud computing infrastructures for Web applications is calculated in deployment and production phases. Recently, the scientific community offers several methodologies to calculate the most suitable infrastructure at these stages to minimize its monetary costs while covering Service Level Agreement (SLA) constraints. On the other hand, Model Driven Web Engineering is taking advantages of code generation from Design level. With both concepts in the scene, in this work we show the first stage toward an approach to estimate the production costs in cloud computing infrastructures at Design phase, choosing the right infrastructure for the job. The process we have performed started defining the variables of the analysis, measuring the time needed in each different combination obtained, validating the confidence of results obtained and finally applying them to an illustrative example to exemplify the proposal in practical terms.

Key words: Web Engineering, IFML, CRUD

1 Introduction

Currently, cloud computing infrastructure has become the most widely used solution for deploying Web applications, mainly due to the flexibility, agility and availability capabilities that it offers [5]. Cloud computing enables Web developers to use remote hosting services to improve onsite infrastructure. In that sense, the organization systems can be specified at a deep level of detail according to the service and quality level agreements that determine the kind of servers, its arrangement and its scalability options. In that sense, there are many factors that impact the pricing structures, mainly related to the resources used in each moment. As an example, high availability, high data volume and high number of concurrent users are key factors to be considered in data intensive Web applications deployment [12]. Thus, since cost is of utmost importance for the infrastructure, getting an optimal cloud infrastructure is critical, especially when this kind of Web applications comes into this scenario.

Concretely, the definition of a deployment infrastructure for a Web application, previously implemented, is based on a general specification of a service level agreement (SLA) that is usually

defined in terms of [1]: (1) Network latency, (2) host CPU and throughput, (3) memory and (4) storage. Indeed, estimating the capacity of the infrastructure in a systematic and accurate manner at design level allows us to predict the behaviour of the infrastructure under critical load conditions. It is useful to identify the maximum performance expected and its better corresponding cloud infrastructure.

On the other hand, data intensive Web applications development has been widely studied by Model Driven Web Engineering community (MDWE). In the last few years, different methodologies and tools have been presented for supporting the Web development process. Model Driven approaches can also reduce the need for technical and human resources and improve the quality of the final product. Among the many benefits provided by MDWE approaches, such as IFML/WebRatio [3], XANUI [6], LIQUIDXML [14] or OOHDM [11], it is worth to mention the automatic generation of the final application code from the design so that this phase (design) becomes the most important step in the software development process. Other advantages provided by these approaches include productivity improvements, an important increment in software quality or a reduction in costs to adapt the system to changes in requirements. However, these approaches do not support the identification or estimation of production costs for the Web application in a selected cloud infrastructure yet. This identification needs to be made in post-design software development lifecycle stages, such as performance-testing phase [7] neglecting the benefits of an earlier estimation.

As an example, the identification at design level of design decisions that may have a relevant impact in the infrastructure costs may involve changes that would be accomplished before the final system has been generated. The identification of these decisions once the final system has been generated would imply higher cost changes [2].

Given this opportunity, we have formulated the following research question: *can we estimate at design phase the costs of production for a given Web System, developed with a specific MDWE approach and given a service quality requirements specification, to determine a Cloud deployment infrastructure?*

The main goal of this paper is to present an approach for the definition of a cloud computing cost estimation model for Web applications during the design phase. In other words, a design-time evaluation of the infrastructure needed at the next production stage to cover a certain quality of service for the application. This approach would reduce the impact of changes due to decisions about the capabilities and costs of the cloud infrastructure on the development process. As a first step to carry out this estimation, this work presents an analysis of the throughput times of an application, based on different design decisions, in order to have more information to anticipate the impact of the design over infrastructure costs.

This paper is structured as follows. In Section 2 we gather the steps and data used to carry out this first estimation as a function of throughput time. Section 3 analyses the data obtained and Section 4 introduces the production cost estimation method. Section 5 presents the related work and, finally, Section 6 collects the conclusions and future work.

2 Working Environment

In order to make concrete the conceptual framework that the work presented here relies on, we have analysed different MDWE approaches that were mature enough to be used by industry [13]. Among the current approaches, it is worth to mention IFML (Interactive Flow Modelling Language) ([3] that allows the design and development of data intensive Web applications. This standard has also become a reference for the industry in the data intensive web applications development area. WebRatio, the case tool that supports the development of applications by using IFML, allows managing and validating IFML models but it also provides engineers with the automatic generation of the final code of the application based on a particular J2EE target platform. WebRatio can be freely downloaded[a]. Furthermore, this study has been developed with the collaboration of a Software Development Company (Homeria Open Solutions, S.L.), which possesses a long experience in the usage of WebRatio for Web Application Development. This context led us to select IFML/WebRatio as the MDWE approach under study.

Although we have performed this study using a concrete MDWE approach (IFML/WebRatio), it has intentionally defined independently from such approach. Therefore, following the process depicted herein, we may obtain data operation execution average times for different MDWE approaches to provide cost estimates at design time, which remains the main goal of this work.

2.1 Study Design

As a first step to answer this question, in this work we focus on assessing how different design and production parameters (independent variables of our study) impact the response time (dependent variable) of a Web application.

To analyse the impact of the independent variables treatment in the study, we have defined a canonical design that will be used in all the assessments. This design consists of an IFML navigation model composed of a set of CRUD operations (Create, Read, Update, Delete). Note that CRUD operations represent the tasks that are most frequently repeated in IFML designs and, thus, the operations with a higher activity load in data intensive Web applications [10]. The model that has been designed follows the next pattern: first, a Create operation is executed; second, a Read is performed; next, an Update; and, finally, a Delete.

Once the core design has been specified, the independent variables related to the design aspects that may affect its response time have been defined. Concretely, the variables are the next: (a) number of attributes in the data entity that the operation is performed over (in this case, we have considered values of 1, 10, 20, 30, 40, 50); (b) persistence type considered for the operation (data stored into the data base, data as session variable or data as application variable at memory level). For the sake of simplicity, the type range for data entity attributes has been blocked to string in this work.

On the other hand, the production independent variables refer to the technical characteristics of the deployment infrastructure that may affect the response time of the application. In this case, the independent variables considered are: (c) computational capacity of the server instance were the tests are performed (two different machines are used: an EC2 t2.micro Amazon Web Services with a 2,5

[a] http://www.webratio.com

GHz Intel Xeon processor and 1 GB of RAM memory; an EC2 t2.small Amazon Web Services with a 2,5 GHz IntelXeon processor and 2 GB of RAM memory); (d) the number of concurrent users that may launch a particular operation in the Web application (in our case, this parameter may have the values 1, 5 and 10 to cover different scenarios). Furthermore, the deployment software stack has been blocked to a concrete version of Tomcat and PostgresSQL, so they do not introduce undesired alterations to the measures.

In order to provide a complete view of the problem domain, Table 1 presents a comprehensive list of design and production variables, which may have a clear impact on the dependent variable (response time), together with their domain range. Note that a significant subset of those independent variables has been fully considered in this study. The rest variables haven blocked or not considered here but their analysis is planned as a future work. Additionally, it is worth to mention that the variables considered are independent of any specific MDWE approach. Therefore, they remain valid in the case of performing the same study considering a different MDWE approach.

Category	Considered	Variable	Range of values
Design	Yes	CRUD operation	C, R, U & D
Design	Yes	Number of attributes of the entity	1, 10, 20, 30, 40 or 50
Design	Blocked	Attribute Type	float, integer, date, time, text, String, etc.
Design	No	Entity Relationships	One-to-many and many-to-many
Design	Yes	Persistence	Application, session or data base
Design	Yes	Persistence	Application, session or data base
Design	No	Transactions	CRUD combinations
Production	Yes	Computational capacity	EC2 t2.micro Amazon Web Services EC2 t2.small Amazon Web Services
Production	Blocked	Software stack	Apache Postgresql 9.5.4 (AWS db.t2.micro, or db.t2.small)
Production	Yes	Number of concurrent users	1, 5 or 10

Table 1 Independent variables for the study.

2.2 Set-Up

Based on the usage of IFML, our main research question, introduced in previous section, was refined as follows: *can we estimate at design phase the costs of production for a given Web System, developed with IFML/WebRatio and given a service quality requirements specification, to determine a Cloud deployment infrastructure*?

Given the great amount of data and the existence of different combinations for the testing groups, all the results have been represented by means of a 5-dimension ROLAP [8] cube. In order to represent that cube we have specified a Data Model (see Figure 1) that comprises those five dimensions: (1) operation type by means of the entity *Operation*; (2) persistence type by means of the entity Duration; (3) attribute number by means of the entity *NumAttributes*; (4) number of concurrent users by means of the entity *NumUsers*; and (5) deployment machine features by means of the entity

ComputationalCapacity. All those previous entities are connected to an entity named *CostRegistry*, which stores the measurements obtained for each combination of the values of the five dimensions considered in terms of time.

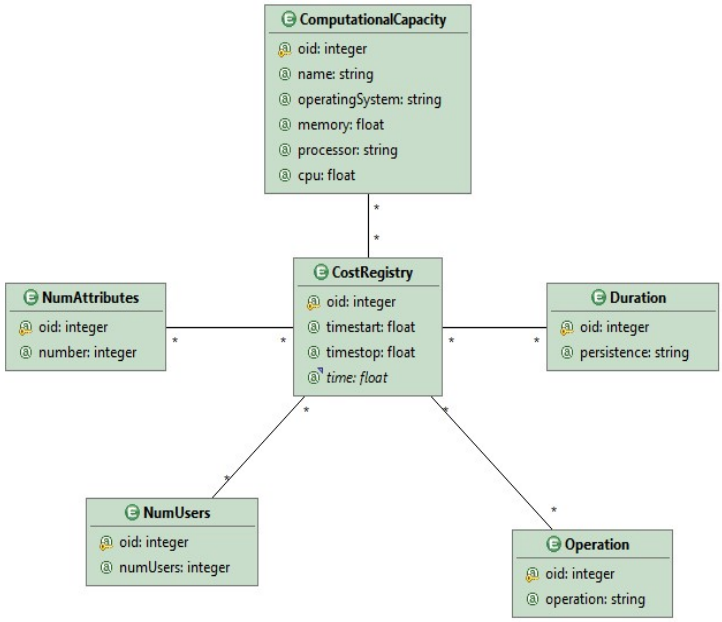

Figure 1 Data Model for managing the 5 dimensions used.

On the other hand, to perform every CRUD operation we have specified all those data operations according to the usual patterns in IFML. At this point, concerning combination packaging and deployment, we have two main alternatives:

(1) Creating four projects: one per each operation type. In practical terms, it means to create and deploy 864 war projects (=4 operations * 3 persistence types * 6 categories of attributes * 3 different sets of users * 2 deployments environments).

(2) Specifying a single operation chain to sequentially perform the four types of data operations (C, R, U, D). In this case, we only have to manage 216 war projects. Therefore, this was the alternative finally selected.

For the sake of simplicity, Figure 2 shows a reduced version of the IFML model specifying the second alternative approach. Note that every group of action units implementing every data operation is highlighted with grey squares. In general terms, the main idea is to measure the time needed in each operation by getting the time just before and after its execution. To this purpose, we have used a *time unit* placed before and after each operation unit. So, we can capture the exact time used for each operation. Additionally, in order to have a relevant set of data and to be able to dismiss abnormal results, each combination of values for the independent variables has been repeated 2.000 times. Thus, we have executed 2.000 create operations, then, 2.000 reads and so on. In this case, the *loop unit* is responsible of providing those iterations. The process starts at the point tagged as 1 in Figure 2,

concretely in the *No Operation unit* marked as Home. This unit is used to trigger the whole operations chain and to establish the number of iterations (2.000 in our study). Secondly, we show the units used to measure the Create in the grey square marked as 2 in the figure. The Read is placed in the square 3, the Update in 4 and Delete operation in 5. Finally, basic options to manage the results generated are placed in 6.

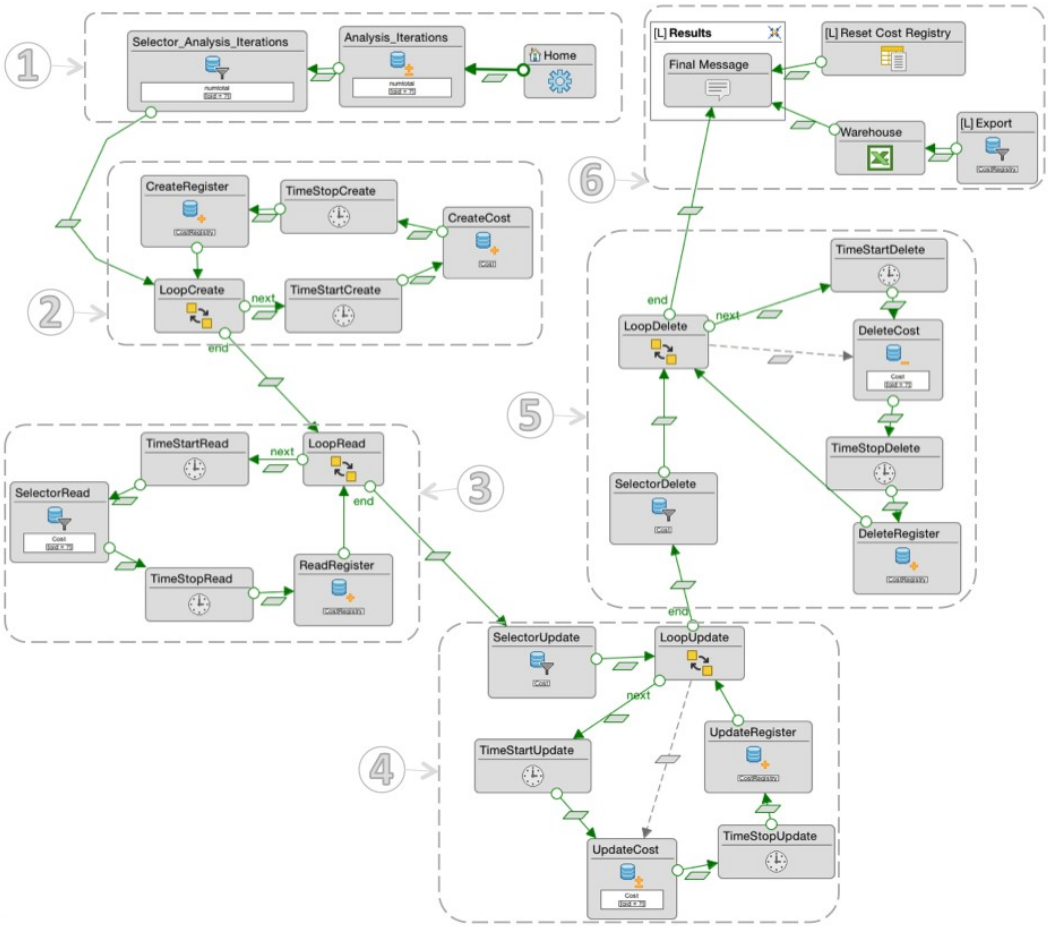

Figure 2 Operation chain to perform the analysis.

3 First Analysis

Once the different tests are finished, all the resulting data have been statistically processed to assess their reliability/confidence, i.e., how similar the execution times of the same test throughout its 2.000 repetitions are. We have applied the k-means method [9] to identify that the behavior was homogeneous in the great majority of the data (>86%) and to be able to discard not relevant outliers. The k-means process is useful for partitioning an N-dimensional population into k sets resulting partitions, which are relevant in the sense of our study. We have reached a value greater than 86%

using three (k=3) clusters. To facilitate this data analysis we have used the R[b] language by means of R-Environment. R is a language and environment for statistical computing and graphics and it provides an integrated suite of software facilities for data manipulation, calculation and graphical display. Figure 3 represents an excerpt of the resulting time plots for each CRUD operation for the case of 10 attributes using data entity at session level and the t2.micro cloud infrastructure. For the sake of better graphical understandability, we have placed the results along the iteration used, depicted in the X axis from 1 to 500 for the sake of understandability, and the time needed for each operation in the Y axis.

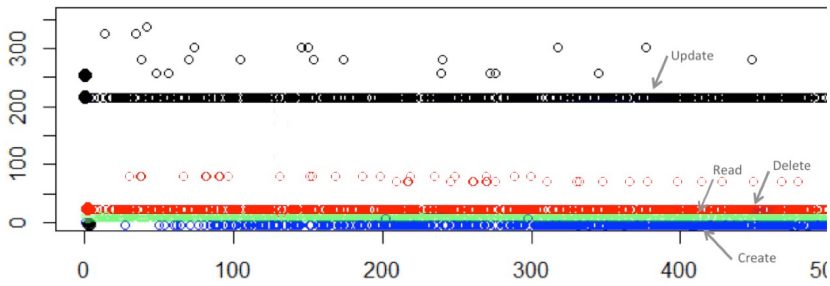

Figure 3 Graphic plot composed using R-Environment for the case 10 attributes/session/t2.micro.

Figure 3 shows that behaviors are highly stable. For the sake of better understanding, the Figure 3 depicts an excerpt of the whole behavior since it is homogeneous for the 2.000 repetitions. Therefore, given a homogeneity coefficient greater than 86%, we can derive a relevant mean execution reference time for every operation. Then, these mean times can be used in the design phase of an application to estimate its production costs. For example, Table 2 shows the mean times (in milliseconds) for the CRUD operations considered, the two amazon machines used, all attributes number cases and given 1 user.

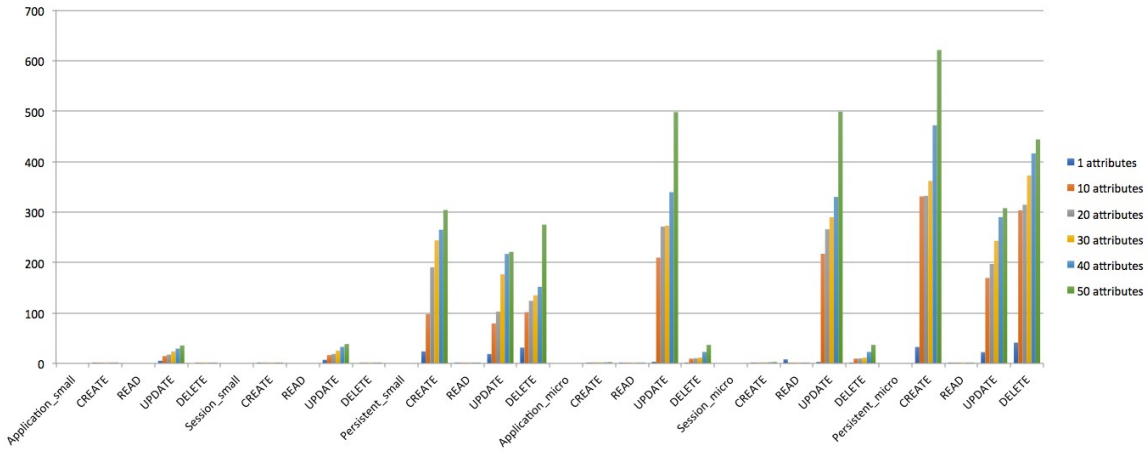

Figure 4 All-in-one mean execution times.

[b] https://www.r-project.org/

	1 attributes	10 attributes	20 attributes	30 attributes	40 attributes	50 attributes
Application_small						
CREATE	0,72	0,93	0,64	1,00	1,02	1,09
READ	0,00	0,00	0,00	0,00	0,00	0,00
UPDATE	5,21	14,60	17,52	23,83	29,28	35,43
DELETE	0,00	0,98	1,05	1,05	0,99	1,10
Session_small						
CREATE	0,82	0,94	1,26	0,82	0,89	1,00
READ	0,00	0,00	0,00	0,00	0,00	0,00
UPDATE	7,11	16,32	18,69	25,43	32,72	38,27
DELETE	0,00	1,03	1,08	1,08	1,10	1,11
Persistent_small						
CREATE	23,76	98	190,53	243,89	264,91	304,00
READ	0,89	1,20	1,22	1,30	1,32	1,36
UPDATE	18,62	79,26	102,54	176,55	216,88	220,98
DELETE	31,22	101,44	124,00	134,96	151,72	275,00
Application_micro						
CREATE	0,69	1,91	1,96	2,00	2,08	2,60
READ	0,59	1,37	1,45	1,42	1,43	1,49
UPDATE	3,22	209,58	271,14	272,82	339,31	498,04
DELETE	1,01	9,47	10,18	11,55	22,77	36,63
Session_micro						
CREATE	0,75	1,26	1,88	2,01	2,24	2,77
READ	8,00	1,32	1,41	1,49	1,50	1,60
UPDATE	2,92	217,25	265,79	289,68	329,91	498,75
DELETE	1,10	9,51	10,00	11,41	22,76	36,64
Persistent_micro						
CREATE	32,58	330,89	332,09	361,58	471,87	620,99
READ	1,10	1,23	1,29	1,38	1,44	1,45
UPDATE	22,37	169,28	197,04	242,97	289,94	307,57
DELETE	41,18	303,30	314,36	372,20	416,23	443,78

Table 2 Mean execution times for CRUD operations.

Figures 4, 5, 6, 7 and 8 visually present the results for the four CRUD operations, considering just 1 user, for all the different combinations of values from the dimensions: machine, persistence and attribute number.

Obviously, Figure 4 shows that, in general terms, working with a higher memory quantity reduces considerably the time needed, i.e., t2.small respect to t2.micro. Regarding the level of persistence, we can see how acting at database level needs always more time. The operations that consume more time are Create, Update and Delete, respectively.

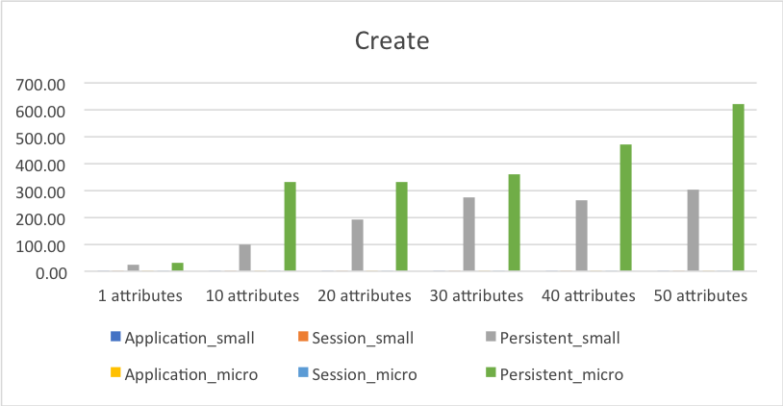

Figure 5 Mean time values plot for every case. CREATE.

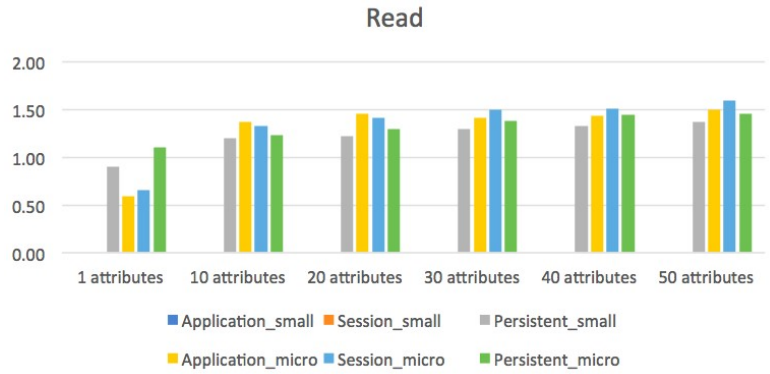

Figure 6 Mean time values plot for every case. READ.

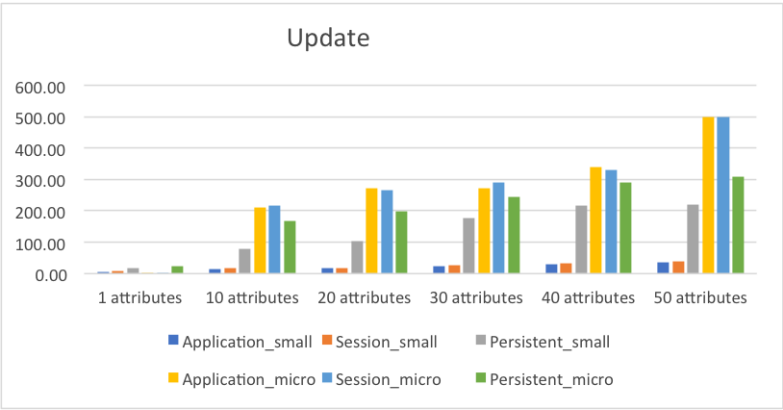

Figure 7 Mean time values plot for every case. UPDATE.

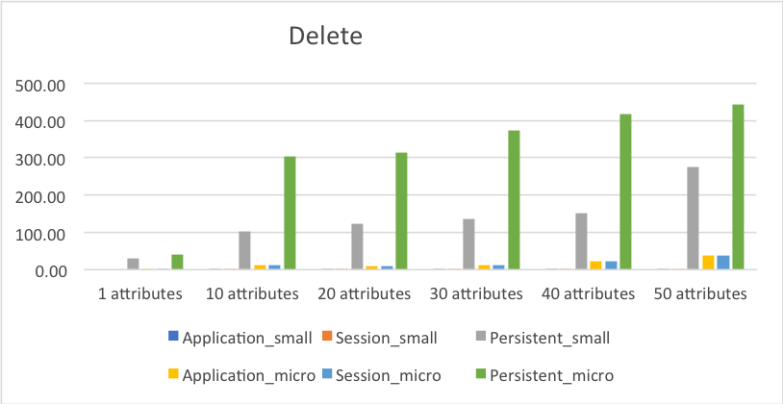

Figure 8 Mean time values plot for every case. DELETE.

From the data obtained, textually and visually given here, we can infer several interesting insights. On the one hand, CREATE and DELETE operations (Figures 5 and 8) behave properly with persistence at memory level (Session and Applications variables), i.e., both operations take really low portion of time for their execution, the memory size does not seem to be relevant to perform them.

On the other hand, for both CREATE and DELETE operations when working on persistence at database level the time became a key factor and we can also notice that the size of the memory plays an important role in these cases, being slightly higher in the AWS t2.micro/db.t2.micro case. In addition, when the number of attributes increases, it has a significant impact on performance.

Regarding READ operations (Figure 6), the scenario ranges from 0 to 1,6 milliseconds. We can observe that the execution of READ operations involves a really low portion of time but, in a particular way, the memory size is really relevant when reading from memory level. When the number of attributes ranges from 1 to 50, it does not significantly impact execution times since the time needed for the case of 50 attributes is close to the rest of cases. For all cases, but mainly in READ one, millisecond seems to be a measure with not enough level of detail to perform the study; at least nanoseconds level should be considered.

As also expected, better hardware features set (more RAM, in this case) for the deployment machine implies a relevant reduction on the mean execution time for every considered case of the UPDATE operation (Figure 7). Conversely, a greater number of attributes implies higher execution times. Additionally, it may be noted the high impact of the RAM availability when data entities are stored inside session or application scopes.

4 Production Cost Estimation

From the first data obtained here we can take decisions in order to optimize the execution times of the CRUD operations groups at design phase and we can give the first step to estimate the throughput time to convey the general services level specification. In other words, once we have estimated the operation times from this initial study for the different combinations, it is possible to assign a concrete value to each particular operation that conforms an operation chain in the business logic of an application at design phase. We define an Operation Chain as the set of all of the CRUD operations

that are launched sequentially and that must be executed from the first to the last, as if it was a transaction. We do it by adding up all the operation times of each of the operations involved in such operation chain.

$$\max_{j:C_j \epsilon OC} \sum_{i:O_i \epsilon C_j} time(O_i)$$

The formula above obtains the estimated execution time (indicated as time) of the longest operation chain (Cj) by selecting the maximum value from the set of estimated execution times of all the Operation Chains (OC) in the design. The function time returns the mean time of every operation (Oi) inside the Operation chain Cj given its type, the number of attributes in the involved data entity, and its type of persistence.

Following we present a specific example to illustrate the cost estimation process. Simplified versions of three typical operation chains defining common transactions are shown here. Those operation chains have been synthetized from samples extracted of more than 100 projects of Web applications developed by a Spanish software company, Homeria, S.L., which it is an official partner of WebRatio[c].

The first operation chain (OP1 – Figure 9) exemplifies a common transaction to consolidate a stream of incoming data into a new summary record in the database. In this case, we are just considering the following steps in such transaction: (1) a new summary instance is created (CREATE operation over an entity composed by 10 attributes and persisted into database); (2) new data item instances are created CREATE (CREATE operation over an entity with 30 attributes at session) within session representing the incoming data; (3) all those new data items are eventually read (READ operations over entities with 30 attributes at session) and processed; (4) results are consolidated into the summary record (UPDATE operation over an entity composed by 10 attributes and persisted into database); and (5), finally, all the data items at session level are deleted (DELETE operations over entities with 30 attributes at session).

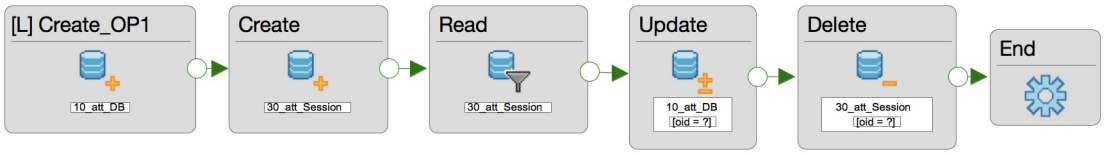

Figure 9 Operation chain OP1.

For the second one (OP2 – Figure 10) the whole chain is composed sequentially by a CREATE (20 attributes at data base) + an UPDATE (20 attributes at data base) + a DELETE (20 attributes at data base). In this case, we try to simulate common user transactions with a data entity persisted into database. They do not actually define a real operation chain, but a set of operations accessing the database so concurrency issues may be exposed.

[c] http://www.webratio.com/site/content/es/partners

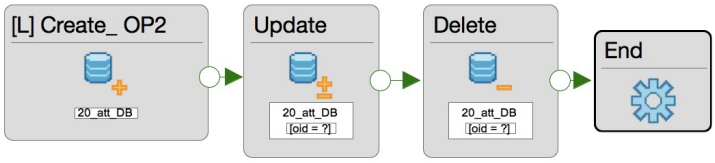

Figure 10 Operation chain OP2.

For the last one (OP3 – Figure 11) the whole chain is composed sequentially by a CREATE (30 attributes at data base) + a CREATE (10 attributes at application persistence level) + a READ (30 attributes at data base) + an UPDATE (10 attributes at application). This case is just an example of generating temporal data from persisted data for processing or presenting them conveniently.

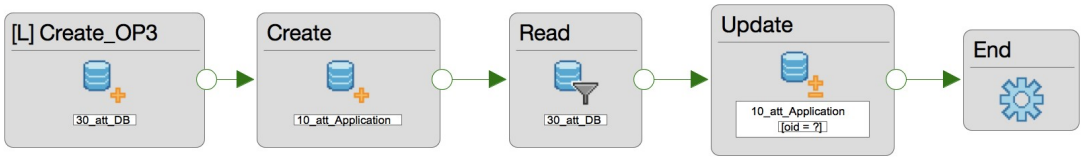

Figure 11 Operation chain OP3.

Let us suppose that the SLA establishes a maximum unitary user performance time of 420 milliseconds –*Time (ml)1u*– with a runtime growth, following a logarithmic scale depending on the number of concurrent users, with a maximum performance peak of 3.95 seconds for attending 1.500 users simultaneously –Time (ml)nu–. nu references the number of concurrent users. When reaching the Time (ml)nu the operating margin should be greater than 7% to establish the viability of each operation in the deployment infrastructure. Also as an example, we can assume that the economic viability of the project recommends a design that can be performed correctly in a t2.small deployment infrastructure when it is pos.

	Create	Create	Read	Update	Delete	Time (ml)1u	ET (ml)un	Viability un	Suitable	SLA (max ml) un
	10 attributes	30 attributes	30 attributes	10 attributes	30 attributes					
OP1	Data Base	Session	Session	Data Base	Session					
t2.small	98	0,89	0	79,26	1,08	179	3379,09	85,55%	Yes	3950
t2.micro	330,89	2,24	1,49	169,28	11,41	515,31	4068,10	102,99%	No	3950
	Create	**Update**	**Delete**			**Time (ml)1u**	**ET (ml)un**	**Viability un**		**SLA (max ml) un**
	20 attributes	20 attributes	20 attributes							
OP2	Data Base	Data Base	Data Base							
t2.small	190,53	102,54	124,00			417	3930,31	99,50%	No	3950
t2.micro	332,09	197,04	314,36			843,48	4389,11	111,12%	No	3950
	Create	**Create**	**Read**	**Update**		**Time (ml)1u**	**ET (ml)un**	**Viability un**		**SLA (max ml) un**
	30 attributes	10 attributes	30 attributes	10 attributes						
OP3	Data Base	Application	Data Base	Application						
t2.small	273,89	0,93	1,3	14,60		291	3695,21	93,55%	Borderline	3950
t2.micro	361,58	1,91	1,38	209,58		574,45	4138,88	104,78%	No	3950

Table 3 SLA coverage in OP1, OP2 and OP3.

Table 3 shows the operations for OP1, OP2 and OP3 and the values for each operation regarding persistence type and number of attributes. The column named as *Time (ml)1u* adds up the whole operation for each case and the column named *ET(ml)un* indicates the estimated time in milliseconds when working with the maximum number of concurrent users expected. Obviously, *ET(ml)un* value must be less than maximum performance peak value that SLA indicates. The column tagged like *Viability un* shows a green background colour when such operation could be performed in the deployment infrastructure attending the SLA operating margin, while it shows an orange background colour when the value is between 93%-100% and a red colour is used in those cases where the SLA is overpassed. The last column (Suitable) indicates if the deployment infrastructure evaluated is suitable to perform each operation according the SLA values.

If we analyze Table 3, we can observe that the level of 1 user in SLA is fulfilled for the three operations if we use a t2.small machine (179ml in OP1, 417 ml in OP2 and 291 in OP3, all of them marked in green colour). All of them are lower than 420ml.

However, when we plan the results up to 1,500 users we can appreciate that the three operation chains (OP1: 3.379,09 ml, OP2: 3.930,31 ml, OP3: 3.695,21 ml) could be executed in a t2.small deployment infrastructure within the required parameters (referred in green colour in the cells of the *ET(ml)un* column) but, unfortunately, none of them could be carried out in a t2.micro deployment infrastructure (referred in red colour in the cells of the *ET(ml)un column*). Conversely, only the OP1 fulfils the operating margin as indicated in the *Viability* column (85,55%).

At this point we may determine that although the three operation chains could be performed in a t2.small infrastructure we should choose a higher deployment infrastructure to not force the SLA. Notwithstanding, and trying to fulfil the economic recommendation of the project (a design that can be performed correctly in a t2.small deployment infrastructure when it is possible), we may try to re-design mainly the OP2 to evaluate if it could be feasible.

In conclusion, as noted, the development process is not actually altered. Engineers keep following the same stages and accomplishing the same activities as before. However, our approach provides them with additional information in order to make the rightest decisions at design time, according to infrastructure cost estimates for every operation chain they are specifying. Such information may be of great help to lead, for example, the refactoring of extremely expensive operation chains

5 Related Work

Optimization of cloud computing infrastructure in the Web at production phase has been addressed in the last years by several works. We can find several authors that tried to face this issue from different perspectives.

Concretely, in [1] and [4] the authors propose different methods for estimating the cloud infrastructure pricing, once the application has been developed.

In [1], a probabilistic model is introduced to determinate the pricing, performance and reliability given a set of service requirements. This information is combined with the real cloud provider prices to find the most suitable cloud infrastructure. A wide range of executions with different values for the parameters used must be performed over the final application to identify the most suitable infrastructure combination. Similarly, [4] presents a model based on the characteristics of three

purchasing options provided by Amazon EC2, which can be used for guiding the capacity planning activity. In both cases, the proposals act once the application has been developed and is ready to be deployed.

On the models domain hand, a slightly more similar solution to our work is proposed in [7], where the authors describe also an approach to calculate operating cost and performance needs to suggest a suitable cloud computing infrastructure but at design phase in this case. It can be performed by means of a UML extension that collects the cloud computing infrastructure capabilities for designing the infrastructure combinations. They model the infrastructure expected with a UML activity diagram. From that model, this approach incorporates a cost estimation algorithm to calculate the production pricing, that uses previously known values for factors like load, storage, concurrency, peaks, and so on, that is, the deployment infrastructure is modeled and the cost is calculated from there.

In [5], the authors propose a heuristic algorithm to help the developer in the decisions related to the placement of the tasks when deploying a web application into a cloud infrastructure. The algorithm deals with the placement of the subtasks in the different nodes of the virtual machines in order to reduce data transmission and communication traffic. The authors claim that the algorithm provides important benefits in terms of completion time of the web applications. Unlike our work, this algorithm would be executed once the system (and its resources) has been completely generated.

Other interesting approach that uses model driven engineering for managing and configuring software systems at runtime to satisfy desired quality attributes is StratusML [15]. StratusML is a modeling framework for cloud applications and it provides views of the different cloud application concerns through utilizing the concept of layers facilitating visual modeling of adaptation rules, and using template-based transformation to deal with platforms heterogeneity.

Also in [18], [17], [16] the authors propose the use of Model Driven Development for featuring the capabilities of the deployment infrastructure by mean of model and metamodels. E.g., applications developed using the framework in [18] are designed at a high level to abstract from the targeted Cloud, and then semi-automatically translated into code able to run on multi-Cloud platforms.

Cloud Computing Infrastructures allow provisioning the developer with multiple instance types that provide different combinations of CPU, memory, disk and networks. These proposals use models/metamodels to get a catalog of capabilities of deployment infrastructures and to model the concrete final schema. They act once the application has been developed and is ready to be deployed. Hence, according to the features of that application these approaches allow to design an infrastructure on a specific cloud provider. However, in all these works the deployment infraestructure is modelled or generated according to the final application so that the system is not optimized to fulfil a particular SLA. This is exactly where our approach comes to the scene since different design decisions may be taken in order to accomplish such as SLA and the performance of the application may be subordinated to the underlying infrastructure or to the limitations of architecture and software. In other words, despite the contributions provided by the previous works that have been aforementioned, none of them have addressed the problem that is faced in this work.

6 Conclusions and Future Work

In this paper, an approach for estimating production costs and cloud infrastructure for data intensive Web applications has been presented. To estimate these costs, a study has been developed where the execution times of different CRUD operations have been measured based on a series of design and production parameters. This analysis shows interesting and promising results regarding the possibility of establishing a first infrastructure estimation based on the significant (independent) variables considered in the study. The variables considered in the study included the operation type, number of attributes, persistence type, potential users and deployment machine capabilities. Based on these 5 different dimensions we observed how they impact the execution time of a canonical system so that the developer may extrapolate the results to other systems and anticipate, in the early stages of design, the computing needs and cloud infrastructure that the application will need for its later deployment and, thus, take the corresponding decisions before deploying the system. To the best of our knowledge, there is not a similar approach for IFML nor other MDWE approach in the literature so that the one presented here comes to bridge this gap.

As a different contribution, we also identified several aspects that can influence the effectiveness of cost estimation such as database type, cache capabilities, the ability to scale the application on several machines by elastic growth or resources sharing by other applications, as well as the correlations between these aspects and the throughput time. The evaluation of these aspects is also planned in our roadmap for future work. Similarly, we also plan to monitor the behavior of the processor, RAM and storage capacity at each point in the execution of the operation chain in order to evaluate the quality requirements in these aspects.

Due to the study is focused on the basic CRUD operations and the executions of them out of a transaction, it could be interesting to replicate the execution of complex transactions that could cover a large amount of data intensive Web applications including a subset of transactions like the creation of complex entities, remove complex entities, and so on.

Complementary, we are planning to evaluate the impact for each type of attribute placed in an entity (float, integer, date, time, text, etc.) and to identify how their different combinations may affect the results. Regarding the data model, we also need to identify the performance data when using 1:N (one to many) and N:M (many to many) relationships.

We will also work in a plugin for WebRatio that will provide the designer with automatic suggestions about what cloud infrastructure is estimated as necessary based on the identification of the design parameters discussed in this work. Morever, by means of connecting to the AWS (Amazon Web Services) infrastructure cost calculator[d], the plugin could also automatically calculate infrastructure pricings based on the selected operating parameters. Even, the plugin could suggest a first visual infrastructure proposal by connecting to AWS Cloudcraft[e] . Finally, in the long-term, we plan to apply the study to other MDWE proposals or even Web development frameworks so that we would be able to estimate, at design time, the infrastructure costs of different frameworks and to compare them, supporting the decision making regarding the chosen infrastructure.

---

[d] AWS Calculator, 2017. https://calculator.s3.amazonaws.com/index.html
[e] Amazon CloudCraft, 2017. https://cloudcraft.co/

Acknowledgements

The authors gratefully acknowledge the support of TIN2015-69957-R (MINECO/FEDER, UE) project, Consejería de Economía e Infraestructuras/Junta de Extremadura (Spain)- European Regional Development Fund (ERDF)- GR15098 project and IB16055 project and Homeria Open Solutions, S.L. to the work here presented.

References

1. Andrzejak, A., Kondo, D. and Yi, S., Decision Model for Cloud Computing under SLA Constraints. In 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, (Miami Beach, FL, United States, 2010) 257-266. doi: 10.1109/MASCOTS.2010.34
2. Boehm, B. W. Software Engineering Economics 1st Edition. Prentice Hall HTR. New Jersey. 1981. ISBN-13: 978-0138221225.
3. Brambilla, M. and Fraternali, P. Interaction Flow Modeling Language – Model-driven UI Engineering of Web and Mobile Apps with IFML. Morgan Kauffman, USA, 2014. ISBN 9780128005323.
4. Tian, C., Wang, Y., Qi, F. and B. Yin, Decision model for provisioning virtual resources in Amazon EC2. In 2012 8th international conference on network and service management (cnsm) and 2012 workshop on systems virtualization management (svm), (Las Vegas, NV, United States, 2012), 159-163. ISBN: 978-3-901882-48-7.
5. Fu, X., Cang, Y., Zhu, X. and Deng, S. Scheduling Method of Data-Intensive Applications in Cloud Computing Environments. Mathematical Problems in Engineering, vol. 2015, Article ID 605439, 2015. doi:10.1155/2015/605439.
6. Hermida, J., Meliá, S., Arias A., XANUI: a Textural Platform-Indepedent Model for Rich User Interfaces. Journal of Web Engineering, 2016, 15(1). 045-083.
7. He, H., Ma, Z., Li, X., Chen, H. and Shao, W., An Approach to Estimating Cost of Running Cloud Applications. Based on AWS. In 19th Asia-Pacific Software Engineering Conference, (Hong Kong, China, 2012), 571-576. doi: 10.1109/APSEC.2012.84
8. Morfonios, K., Konakas, S., Ioannidis, Y. and Kotsis, N. ROLAP implementations of the data cube. ACM Comput. Surv., 2007, 39(4). Article 12.
9. MacQueen, J., Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, 281-297, University of California Press, Berkeley, Calif., 1967.
10. Rodriguez-Echeverria R., Conejero J.M., Preciado J.C. and Sanchez-Figueroa F., AutoCRUD - Automating IFML Specification of CRUD Operations. In Proceedings of the 12th International Conference on Web Information Systems and Technologies - Volume 1: APMDWE, ISBN 978-989-758-186-1, pages 307-314.
11. Rossi, G, Pastor, O, Schwabe, D, Olsina, L.. Web Engineering: Modelling and Implementing Web Applications. Human-Computer Interaction Series, Springer-Verlag, London, 2007. ISBN 978-1-84628-923-1
12. Suresh, S. and Sakthivel, S. System modeling and evaluation on factors influencing power and performance management of cloud load balancing algorithms, Journal of Web Engineering, 2016, 15(5&6). 484-500
13. Toffetti, G., Comai, S., Preciado, J. C. and Linaje, M. State-of-the Art and trends in the Systematic Development of Rich Internet Applications. Journal of Web Engineering, 2011, 10(1). 70-86.
14. Robles E., Rossi G., Rivero M., Dominguez-Mayo F., García-García J. and Escalona M. Towards Fast Metamodel Evolution in LIQUIDML. Journal of Web Engineering, 2017, 16(3). 183-211.

15. Hamdaqa, M. and Tahvildari, L. Stratus ML: A layered cloud modeling framework. In Cloud Engineering (IC2E), 2015 IEEE International Conference on. 96-105.
16. Alili, H., Drira, R. and Ghezala, H. H. B. Model Driven Framework for the Configuration and the Deployment of Applications in the Cloud. In CLOUD COMPUTING 2016 : The Seventh International Conference on Cloud Computing, GRIDs, and Virtualization, 73.
17. Caglar, F., An, K., Shekhar, S. and Gokhale, A. Model-driven performance estimation, deployment, and resource management for cloud-hosted services. In Proceedings of the 2013 ACM workshop on Domain-specific modelling. ACM. 21-26.
18. Ardagna, D., Casale, G., Petcu, D., MODACLOUDS: A Model-Driven Approach for the Design and Execution of Applications on Multiple Clouds, MiSE-2012. In Proceedings of the 4th International Workshop on Modeling in Software Engineering (MiSE '12). IEEE Press, Piscataway, NJ, USA, 50-56.