A Unified Model for Server Usage and Operational Costs Based on User Profiles: An Industrial Evaluation

Johannes Pelto-Piri
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden
Email: johannes.peltopiri@gmail.com

Peter Molin
Malvacom AB
Soft Center Fridhemsvägen 8
SE-372 25 Ronneby, Sweden
Email: peter.molin@malvacom.com

Richard Torkar
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden
Email: richard.torkar@gmail.com

***Abstract*—Capacity planning is essential for providing good quality of service, for that reason we need to be able to predict the usage that the applications will impose on our servers. This paper presents a unified model that can predict the usage, hardware requirements and, ultimately, the operational costs. The goal of this study is to present a model for capacity planning. The model presented is developed and evaluated within the industry. The evaluation is used to analyze the possibilities of the proposed model. The models have been evaluated within a company using historical data that originates from production software. The evaluation was done by running the model against three applications and mapping the result to a selection of Amazon EC2 cloud instances. We then provided the same data to five developers and asked them which instance they would have chosen for the applications. Two of the developers suggested the same instance as the model, the second smallest on the scale. The remaining three developers chose the instance one step above the model's recommendation. The evaluation showed that the model is able to produce estimations that are comparable to expert opinions. The unified model in this paper can be used as a tool to estimate the usage, hardware requirements and the final cost of the servers. The model is easy to setup within a spreadsheet and contains parameters that are easy to obtain from access logs and various logging tools.**

I. Introduction

Mobile applications have gained a high market penetration [1] that is continuously growing. That, in combination with the rise of cloud computing, makes it easier than ever for small companies and independent developers to reach out to a large user population. For mobile applications it is more and more common to utilize context aware and multimedia services, which require more computational resources than their predecessors [2]. As cloud computing is expensive for applications with a heavy load and large data transfers [3], [4], small companies and independent developers need be able to make informed decisions about which infrastructure to use in order to provide cost-effective Quality of Service (QoS).

This study has been conducted at Malvacom AB, a growing startup in Sweden that develops a data synchronization service called mAppBridge. The goal of this study is to investigate how users and applications can be modeled and then derive the server usage for mobile applications and, consequently, set requirements for the infrastructure.

Capacity planning is the process in which we determine the capacity needed for our services in order to provide QoS. Literature has proposed many methods [5], [6], [7], [8], [9] for this task. Menascé and Almeida describes capacity planning as a series of steps where we identify the workload, forecast the performance and then do a cost analysis [5], in which it us up to us to implement and calibrate a workload and a performance model. However, Gunther [10] discusses the need for a simple to use capacity planning framework, arguing that the frameworks currently used are too complex. In this study we aim to create a unified model that can be easily implemented to provide an overview of applications' hardware requirements and final costs. Thus, the main focus of our study, and hence also for this paper are:

- Present a unified model that can be used to receive an initial estimate about the hardware requirements needed to sustain a certain size of user population (Section II).
- Show that the unified model is easy to calibrate and capable of modeling server usage, hardware requirements and costs (Section III).

II. A Unified Model

To be able to derive the cost for the applications, we will first model both the application and its users, and derive the traffic from those two entities. Secondly, we will also model the software's hardware requirement for that traffic. Finally, once we have the hardware requirements we will derive the operational costs.

We have created a unified model for this. We have created a User Model that models a user's profile, i.e. their availability. The User Model is then used by the Application Model that models the load imposed on the servers. Next, we use a Software Model that models how much hardware that the software will require under the load derived from the Application Model. The Hardware Model is then, in its turn, used to model the cost of the hardware requirements generated by the Software Model. An overview of the models can be seen

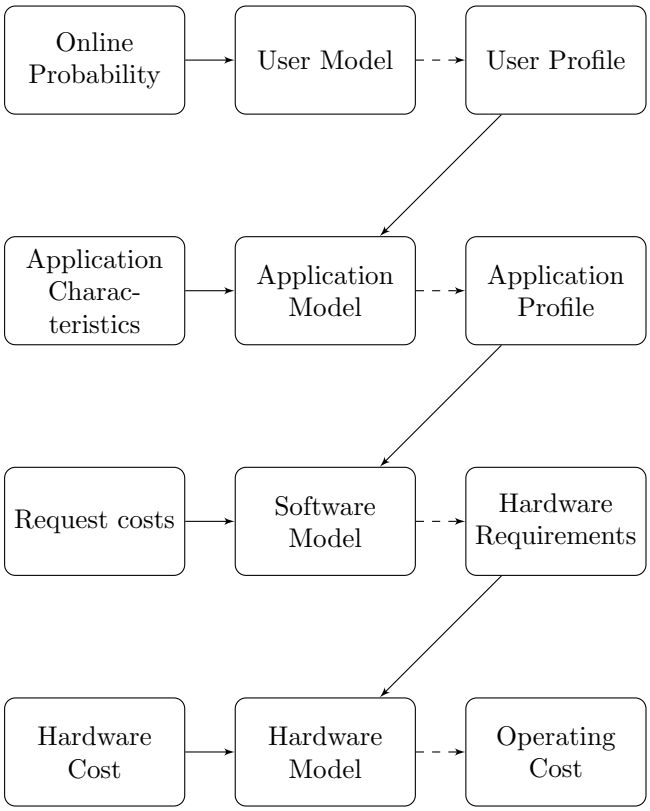

Fig. 1. Model overview.

in Fig. 1. (The boxes to the left represent the parameters to the model while the boxes on the right represents the outputs from the models.)

A. User Model

The user model consists of one parameter P that contains 24 values; each value represents an hour t of any given day i.e. $T = \{t_1, t_2, \ldots, t_{23}, t_{24}\}$, and describes the probability that a user will be online during t. The P parameter will later be used as an input to the Application Model.

$$P_t = \{P_1, P_2, \ldots, P_{23}, P_{24}\}$$

B. Application Model

The Application Model describes how many users the application has. It uses the User Model's probability function to determine the number of users online at a given moment. This is later used to derive the maximum concurrent users at a given hour and the total requests for one day. The Application Model takes the following parameters for modeling the application:

- **Data Pattern (DP_t)** The data activity for the hour t.
- **User Population (U_{pop})** The user population of the application.

The DP parameter contains a vector of decimals that describes the usage profile for the application that we are modeling, $DP = \{DP_1, DP_2, \ldots, DP_{23}, DP_{24}\}$. Where 1 means that there is always something new to fetch from the server. This is used to model applications that rely heavily on push notifications, in which the server is responsible for initiating the communication between the server and the client. For example, an application that sends out four updates per hour is likely to have more traffic than an application that only sends out two. The U_{pop} parameter is a natural number that represents the number of users for our application. The output of the model is in the form of a vector R that contains 24 elements, one element for each hour that describes the number of requests in that hour. The equation for each element in the vector is described in (1).

$$R_t = U_{pop} \times DP_t \times P_t \tag{1}$$

$$R_{max} = max(R_t) \tag{2}$$

$$R_{sum} = sum(R_t) \tag{3}$$

Where R_{sum} (described in (3)) is the total number of predicted requests for one day and R_{max} (described in (2)) will be the maximum load that the servers will be expected to handle, hence the maximum concurrent users at any time in the system is not expected to exceed R_{max}.

C. Software Model

The Software Model models the amount of hardware resources that will be used by the application and its users. The Software Model requires the following parameters:

- **Request Size (R_{size}):** The average size of each request in bytes.
- **Request Cost (R_{cost}):** The cost of one request for the CPU in megahertz (MHz).
- **User Size (U_{size}):** The size required per user.

As well as these parameters the Software Model also uses U_{pop} and R from the Application Model in order to calculate the requirements.

In short, the model needs to calculate the following requirements: CPU, storage, and bandwidth. We need to provide a CPU measurement strong enough for dealing with our peak loads. The equation for the CPU requirement can be found in (4).

The bandwidth requirement, on the other hand, is divided into two different sub-requirements: *a*) We need to derive the bandwidth that is needed to support the traffic peaks and, *b*) we want to know the daily load to know how much data we will handle per day. For the bandwidth required to we use R_{max} in (5) and for the total traffic per day we use R_{sum} in (6). The storage requirement can be found in (7).

$$CPU_{requirement} = R_{max} \times R_{cost} \tag{4}$$

$$Bandwidth_{peak} = R_{max} \times R_{size} \tag{5}$$

$$Bandwidth_{daily} = R_{size} \times R_{sum} \quad (6)$$

$$Storage_{requirement} = U_{pop} \times U_{size} \quad (7)$$

The models output are described in the formulas above. $CPU_{requirement}$ is defined in MHz, while outputs that are dealing with size (bytes) are defined in the same units as R_{size} and U_{size}. Thus defining the parameters in MB will yield the requirements in MB. Keep in mind that the Hardware Model uses GB as inputs, so by defining the size in MB the output will have to be converted to GB in a later stage.

D. Hardware Model

The main function of the Hardware Model is to calculate the cost of the hardware requirements derived from the Software Model. It calculates the cost for a one month period. The Hardware Model uses all but the $bandwidth_{peak}$ output from the Software Model as inputs. The Hardware Model uses the following parameters:

- **CPU Cost** : (C_{CPU}) The cost for the one MHz of CPU.
- **Storage Cost** : ($C_{storage}$) The cost of storing GB in the servers per month.
- **Bandwidth Cost** : ($C_{bandwidth}$) The cost of using GB of bandwidth per month.

The Hardware Model has the following outputs. The cost of fulfilling the CPU requirement (8), the network cost per month (9) and the storage cost (10).

$$CPU_{cost} = C_{CPU} \times R_{CPU} \quad (8)$$

$$Bandwidth_{cost} = C_{bandwidth} \times bandwidth_{daily} \times 30 \quad (9)$$

$$Storage_{cost} = C_{storage} \times Storage_{requirement} \quad (10)$$

Equation (8) is designed to derive the cost for the cost-performance of the modeled application on the given hardware. For applications hosted at Infrastructure as a Service provider (IaaS providers) such as Amazon or Rackspace we also pay for the bandwidth and storage of our applications. For the bandwidth cost described in (9) we start with calculating the daily cost and then converting it to a monthly basis. For the storage cost (described in (10)) it is common to charge a certain amount per GB for each month, hence we do not convert it to a monthly basis. We simply calculate the cost based on how many GB that will be required and the cost for those.

TABLE I
THE APPLICATIONS INCLUDED IN THE VALIDATION.

Application	U_{pop}	R_{size}
1	15140	351839
2	1334	2210956
3	599	676276

III. EVALUATION

This section contains an evaluation of the models. The evaluation was conducted in Malvacom AB; the parameters were estimated using historical data. The validation was done by modeling three applications that are currently in production. The selection of the applications was done by looking at the user population for the applications. We wanted to compare one small application, one medium-sized and one large from the available dataset in order to compare how the attributes such as the user population and the users availability impacts the actual cost. The selected applications can be seen in Table I. Application 1 is an application used for walking and social interactions while Application 2 and 3 is are social networking applications. The results were then analyzed and compared against developer opinions to see how the model performs compares against expert opinion.

In the first step of the evaluation we describe how we estimated the parameters. During the estimation we had access to data that allowed us to estimate the user's availability and various attributes about the requests such as request size. For the last step, when looking at the cost for deploying, we chose Amazon EC2 due their clear documentation regarding the pricing (obviously any other provider can be used as long as they provide clear data regarding pricing).

A. Parameter Estimation

For the parameter P in the User Model we summarized the traffic for mAppBridge, we grouped each request per hour and took the average number of requests based on the number of days we had collected traffic for and the average number of unique users per day. We chose to treat all requests as a virtual user, which results in 100% user activity when the number of requests was equal to the total population. Our extracted user pattern is described in the matrix below. The population and the average request size for each of the applications are described in Table I.

$$P = \begin{pmatrix} 0,113 & 0,091 & 0,087 & 0,135 & 0,260 & 0,383 \\ 0,530 & 0,613 & 0,626 & 0,652 & 0,660 & 0,675 \\ 0,679 & 0,711 & 0,695 & 0,651 & 0,624 & 0,658 \\ 0,645 & 0,655 & 0,597 & 0,439 & 0,266 & 0,165 \end{pmatrix}$$

Unfortunately, we did not have any good source regarding the data activity for the application so we assume that there were always something new for the user to fetch and, hence, we set the activity to 100% for each hour, i.e. $DP = \{t_1 = 1, t_2 = 1, \ldots, t_{24} = 1\}$.

TABLE II
THE PARAMETERS FOR THE HARDWARE MODEL.

Parameter:	C_{CPU}.	$C_{Storage}$.	$C_{Bandwidth}$.
Cost	\$0.051.	\$0.383.	\$0.01.

For the request cost (R_{cost}) we have used (11). Unfortunately we did not have access to any historical data regarding the CPU utilization, therefore we made an assumption. We assume that we are currently maintaining one server with a single CPU with a processor speed on 2.2 GHz with an average load on 50% for a benchmark with 30,000 requests. We assume that the requests that the server is currently receiving is equivalent in terms of CPU utilization as the request from application we are trying to model. The estimation of the request cost parameter can be found in (12).

$$R_{cost} = \frac{\text{Processor speed (MHz)} \times \text{processor utilization}}{\text{Observed Requests}} \tag{11}$$

$$R_{cost} = \frac{22000 \times 0.5}{30000} = 0.04 \text{ MHz} \tag{12}$$

For the users we assume that our application will take up 5 megabytes of space per user in personal data, i.e. $U_{\text{size}} = 5\text{MB}$

For the Hardware Model we must estimate the cost for the processor, bandwidth and storage. We choose to model the cost for a single small instance on Amazon EC2. At the time of writing a small instance on Amazon with one EC2 unit, which is equivalent to 1.2 gigahertz and includes 160 gigabytes of storage, has a monthly cost of \$61.2. Traffic that leaves amazon costs \$0.01 per gigabyte. The cost for the storage and the CPU is done by dividing the processor speed and the storage with the cost. The parameters can be seen in Table II.

B. Results

The first step is to calculate the number of requests for each of the applications. This is done with (1). Table III shows the summary of the requests for each of the applications while the full request pattern for each of the application can be found in the below matrices.

$$R_1 = \begin{pmatrix} 1704 & 1376 & 1312 & 2045 & 3941 & 5793 \\ 8018 & 9283 & 9471 & 9866 & 9997 & 10219 \\ 10275 & 10767 & 10524 & 9861 & 9441 & 9963 \\ 9763 & 9913 & 9032 & 6645 & 4023 & 2493 \end{pmatrix}$$

$$R_2 = \begin{pmatrix} 150 & 121 & 116 & 180 & 347 & 510 \\ 706 & 818 & 834 & 869 & 881 & 900 \\ 905 & 949 & 927 & 869 & 832 & 878 \\ 860 & 873 & 796 & 585 & 354 & 220 \end{pmatrix}$$

$$R_3 = \begin{pmatrix} 67 & 54 & 52 & 81 & 156 & 229 \\ 317 & 367 & 375 & 390 & 396 & 404 \\ 407 & 426 & 416 & 390 & 374 & 394 \\ 386 & 392 & 357 & 263 & 159 & 99 \end{pmatrix}$$

TABLE III
THE PREDICTED REQUEST PEAK AND DAILY LOAD FOR EACH OF THE APPLICATIONS.

Application	R_{sum}	R_{max}
1	174,018	10,767
2	15,333	949
3	6,885	426
Total	196,236	12,141

When we have the request pattern for each of the applications we can move on and estimate the hardware requirements with the Software Model. The hardware requirements for the applications can be seen in Table IV. The equations used to calculate the requirements are described in Section II-C. The requirements that were in bytes have been converted to GB for convenience.

The last step of the execution is to derive the final cost for the hardware. We do this for each of the applications. The results can be found in Table V.

C. Analysis

When we analyze the results it is important to keep in mind which service provider we are using. In this case we have mapped the results to Amazon EC2 instances. By looking at the requirements (shown in Table IV) we can see that the requirements fit a small instance. We asked five developers to choose an instance for the applications based on the data given in Section III-A. Two of the developers thought that a small instance would suffice while three developers would have chosen a larger instance. This shows that the output of the model falls within the range of the developers' suggestions.

As can be seen, each application is modeled separately; this has resulted in different hardware requirements for each of the applications. Meaning, that in theory, this output can be used to distribute the applications across a set of servers so that we are utilizing each server in an optimal way, possibly leading to fewer resources allocated as we are keeping waste to a minimum. Also, by modeling existing applications on existing servers we can use the model to determine if we can fit a specified application to it, as we were predicting the CPU utilization of each application. This model also allows us to quickly see how the cost and hardware requirements would change if we change our service provider, as we can use one set of Software and Hardware Models for different service providers to compare the cost between them.

It should be noted that in order to gain more accurate results, more data is needed to calibrate the parameters. The parameters for these models are quite trivial in nature and there exists a wide array of software logging tools that can be installed to gather all the necessary data.

IV. VALIDITY THREATS

The goal of the evaluation was to see if the model is suitable for usage. We evaluated this by looking at how the model's final prediction was positioned in comparison with experts. We only compared against five experts, and we do not know how

TABLE IV
THE ESTIMATED HARDWARE REQUIREMENTS FOR EACH OF THE APPLICATIONS.

Application	$CPU_{requirement}$ (MHz)	$Bandwidth_{peak}$	$Bandwidth_{daily}$	$Storage_{requirement}$
1	394.779	3.528	57.022	75.70
2	34.784	1.953	31.572	6.67
3	15.619	0.268	4.339	2.995
Total	445.183	5.750	92.933	85.365

TABLE V
THE FINAL COST FOR DEPLOYING THE APPLICATIONS.

	Final Cost			
Application	CPU_{cost} (\$)	$Bandwidth_{cost}$ (\$)	$Storage_{cost}$ (\$)	Total
1	20.134	17.106	28.955	66.195
2	1.774	9.427	2.551	13.797
3	0.979	1.302	1.146	3.244
Total (\$)	22.704	27.880	32.652	83.236

the comparison would have fared with more people. Also, the evaluation was targeted against mobile applications, furthering limiting the comparison.

The data at our disposal was able to give us the total number of unique users for the application, the user's activity pattern and the average size for each request. We were, however, unable to use the data to estimate parameters such as the user size (US) and various attributes in the Software Model such as the request cost.

V. LIMITATIONS

There are several limitations with this model. As for now the model does not help us to predict the QoS directly, as it simply models the hardware requirements and their cost. Also, the server's storage and network costs are not that accurate when it comes to the final cost since many service providers use predefined templates for the hardware where a certain amount of storage is already included in the price.

The requirements modeled by the Hardware Model does not take any kind of growth into consideration. This is a limitation when looking at the storage requirement. If we were modeling a data intensive application, in which the users are often uploading/downloading new content the storage will grow and hence the required cost for it.

In order to estimate the parameters in the best possible manner access to historical data is required. Also, different applications may target different users and have different user profiles as well, so in order for the historical data to be useful it must contain data from a similar application type.

The CPU requirement are modeled after R_{max}, the maximum numbers of users expected within one hour. This approach is a bit crude, a more realistic approach would be to model the arrival rate within each hour using a suitable probability distribution and then derive and use the mean or maximum arrival rate as R_{max}.

VI. CONCLUSIONS

In this study we have presented a unified model that can be used to predict the workload, hardware requirements and operational costs for a server infrastructure. The models are designed with simplicity in mind and they are easily implemented in a spreadsheet making the results easy to share.

Given a simple calibration of the model, the results are comparable to expert opinions. The result can also be used to compare the cost of service providers, and distribute applications based on their hardware requirements. In the end, these models can be used to help us make a more educated guess when we are performing capacity planning in order to assure cost-effective QoS.

We have conducted a static evaluation of the model. The evaluation has been focused on determining what the impact the model can have. Our future work will be focused on: *a*) Addressing some of the issues as described in Section V and also *b*) investigate the possibilities of adding an easy to use QoS model and finally *c*) validate the models from a business perspective

REFERENCES

[1] M. Meeker, J. Dawson, J. Lu, B. Lu, R. Ji, S. Devitt, S. Flannery, N. Delfas, and M. Schneider, "The Mobile Internet Report," 2009.

[2] C. Canali, M. Colajanni, and R. Lancellotti, "Performance Evolution of Mobile Web-Based Services," *IEEE Internet Computing*, vol. 13, no. 2, pp. 60–68, 2009.

[3] B. C. Tak, B. Urgaonkar, and A. Sivasubramaniam, "To move or not to move: The economics of cloud computing," in *Third USENIX Workshop on Hot Topics in Cloud Computing (HOTCLOUD 2011)*, 2011, pp. 1–5.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds : A Berkeley View of Cloud Computing Cloud Computing," EECS Department, University of California, Berkeley, Tech. Rep., 2009.

[5] D. A. Menascé and V. A. F. Almeida, *Capacity Planning for Web Services: metrics, models and methods*. Prentice Hall, Upper Saddle River, 2001.

[6] M. Koorsse, L. Cowley, and A. Calitz, "Network Application Performance Modelling," in *Southern African Networks and Applications Conference*, vol. 27, no. 0, 2004.

[7] R. Lopes, F. Brasileiro, and P. D. Maciel, "Business-driven capacity planning of a cloud-based it infrastructure for the execution of Web applications," *2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*, pp. 1–8, 2010.

[8] R. Garg, H. Saran, and R. Randhawa, "A SLA framework for QoS provisioning and dynamic capacity allocation," in *Quality of Service, 2002. Tenth IEEE Internatinoal Workshop on*, 2002, pp. 129–137.

[9] J. Allspaw, *The Art of Capacity Planning: Scaling Web Resources*. O'Reilly Media,, 2008.

[10] N. Gunther, “Hit-and-run tactics enable guerrilla capacity planning,” *IT professional*, vol. 4, no. 4, pp. 40–46, 2002.