How to Select Significant Workloads in Performance Models

Giuliano Casale
Neptuny
via Durando, 10 - Edificio G
I-20158 Milan, Italy
casale@neptuny.it

Paolo Cremonesi
Politecnico di Milano
p.zza L. da Vinci, 32
I-20133 Milan, Italy
paolo.cremonesi@polimi.it

Roberto Turrin
Politecnico di Milano
p.zza L. da Vinci, 32
I-20133 Milan, Italy
roberto.turrin@polimi.it

Abstract

The complexity of computer systems requires to consider the interaction of several workloads. Only a limited number of business and technical workloads are usually required to properly model the system.

In this paper, we discuss regression-based estimates of service times required for model parametrization and we focus on the selection of significant workloads. We present an experimental comparison, using real performance logs of a distributed enterprise application, illustrating the benefits of constrained estimations over the traditional approach based on ordinary linear regression.

1. Introduction

As information technology (IT) grows, current enterprise infrastructures increase both in size and complexity. For instance, a typical IT system may simultaneously run several applications on hundreds of nodes and it must handle an increasing number of requests as the number of users or interactions increase. Furthermore, in order to meet the increasing demand and to address the Quality of Service (QoS) requirements, system components are upgraded at a much higher pace than in the recent past.

Capacity planning is often used to evaluate if IT systems are properly sized to sustain future traffic or configuration changes without performance degradations. Capacity planning activities are traditionally performed using simulation or analytical models such as queueing systems [9] or network of queues [2, 11, 14]. Queueing models capture the fundamental relationship between performance and capacity, providing estimates, sometimes even bounds, of key metrics such as system throughput, server utilization and experienced delay at each resource. Nevertheless, queueing model parametrization can be a challenging task. It involves two steps: (i) identifying the meaningful business workloads and (ii) estimating the service time placed by requests at each resource.

There is a general lack of standardization and little theoretical understanding of the parameter estimation problem in the context of queueing models. For instance, many practical questions such as how to define workloads and service classes are often addressed by considerations dictated by personal experience. The most problematic topic is the correct estimation of the service times (i.e., the processing time of incoming requests). While many other parameters (e.g., the number of servers, the system throughput, etc.) can be easily measured, service times are much more difficult to estimate. Direct measurement of these parameters by mean of code instrumentation, benchmarking and load testing are either unfeasible, time consuming and very intrusive.

To address this challenging topic, we propose a robust and constrained optimization-based techniques. We consider the case where aggregate measurements (application throughput and server utilization) are available to the analyst. Note that such measurements are easy to collect. We formulate a technique to estimate the service times based on robust and constrained linear regression.

The estimation problem is complicated because of anomalous behavior in the collected metrics. Such anomalies arise for a number of different reasons, most of them relate to changes in the system configuration (e.g., hardware or software upgrades, service switch).

The work is organized as follows. Section 2 reviews previous work. We discuss the queueing network for performance modeling in Section 3. Section 4 explains the issues arising during parameter estimation. A solution to multi-class problems is presented and evaluated in Section 6. Similarly, Section 5 presents and evaluates a robust regression model. Finally, Section 7 draws conclusions and outlines future work.

2. Related Work

There are few papers directly addressing the service time estimation problem.

An early work on queueing network service time estimation can be found in [17], where Rolia and Vetlan use multivariate linear least-squares regression to compute the service times of distributed software systems. The results are compared with simulation data, showing that linear regression achieves an average relative error between 3% and 7%.

Sharma and Mazumdar in [21] estimate service times and arrival rates in a open queueing network by injecting a test workload in the real system and by monitoring the time behavior of the test workload. The technique is best suited for the parameter estimation problem in ATM networks as it requires to know the exact route of all the requests in the network.

Zhang et al. in [24] as well as Liu et al. in [13] and Wynter et al. in [23] estimate service times in open queueing networks by combining utilization, throughput and end-to-end response time measurements with a quadratic programming techniques.

Estimation technique based on polynomial regression is presented in [22]. This may be of interest for *load-dependent* systems [11], i.e., resources with service times that depend on the current number of jobs waiting for service. A technique for time-varying parameters of queueing models is proposed in [25]. This is based on the extended Kalman filter, which gives online estimates both of service times and user average think times from utilization and response time data.

A clustering-based technique for reducing the number of workload classes in multiclass queueing models is presented in [3]. The method provides bounds on the errors introduced by the aggregation of service classes.

However, none of the previous works address the problem of the robustness of the estimation when anomalous behaviors are present in the monitored system (e.g., outliers, hardware and software configuration discontinuities, service switches, linear correlation between workloads).

3. Queueing network models

Capacity planning analysis allows to forecast the impact of modifications on systems behavior. According to capacity planning, an IT system is composed of a set of resources (e.g., servers, network devices, storage boxes,...) and a set of processing requests, collectively called a *workload*. As an example, a web application can run on a three-tiered system composed by web servers, application servers and a database server; in this case, the workload could be the number of the incoming HTTP requests. Enterprise-size systems may have several workloads. For instance, the workloads for a data center of a mobile-telephone operator might be formed by the number of phone calls and the number of SMS messages.

Capacity planning is typically based on queueing network models. Such models depend on a number of parameters, the most important being the *service times* of each station, i.e., the time required for such station to process one request (e.g., an I/O operation) when no other requests are processing.

Such estimation can be theoretically derived from system log files, even though they present several practical issues which complicate the analysis.

For instance, most default logs do not generally provide exhaustive and detailed measurements. It would be required to increase the logging detail level, or to use performance probes. However, such activities are often invasive and not feasible in real systems.

Furthermore, logs generally present only an aggregation of measures, thus preventing to capture the impact of each single workload on system load. Again, it is generally impractical to set a fine-grained collection of necessary metrics. As an example, the measurements of the utilization for the above mentioned mobile operator data center can comprise the cpu power required to serve both phone calls and SMS messages.

For the same reason, we can not collect active measurements (i.e., by injecting into the system a set of additional requests and then monitoring the service requirements).

Because of the above practical problems, cpu utilization and workload information are usually the only available performance metrics. Despite that, queueing network theory can grant a robust and exhaustive performance evaluation of the system, just starting from the utilization measurements. Queueing network models provide the fundamental performance indices of the system with a good trade-off between accuracy and efficiency [11].

The Utilization Law [11] is one of the keystones in the setting of queueing networks. Despite its simplicity, it captures the essential relationship between the service requirement of a server and the processing requests. Moreover, it can be used in almost all systems, since it requires very weak assumptions. In fact, the utilization law is an application of the more general Little's Law [12], which holds with very few hypothesis.

The Utilization Law states that the utilization U of a server is linearly dependent from the mean arrival rate λ of the incoming requests and from the service time S:

$$U = \lambda S \tag{1}$$

In real systems, each server serves several types of workloads, referred to as classes, each one with its own statistical characteristics. Let us consider a web server executing two web applications. We can have a work-

load A corresponding to HTTP requests for application A with mean arrival λ_A and average service time S_A, and a workload B corresponding to application-B's HTTP requests with mean arrival λ_B and average service time S_B. Assuming that there are R different service classes, it is straightforward to extend (1), which becomes:

$$U = \sum_{r=1}^{R} \lambda_r S_r \tag{2}$$

where S_r and λ_r are, respectively, the service time and the mean arrival rate of class-r requests.

We will see in Section 4.1 that it is not easy to select the workloads that better describe the system. By having a correct definition of the R classes and accurate estimation of S_r, it is possible to predict the future usage levels in presence of a certain input workload. Further, for the popular class of open product-form networks [1,5,7], the knowledge of λ_r and S_r, for all classes, is sufficient to compute all performance metrics of interest, such as response times and average amount of unfinished work at each server.

In order to parameterize (2), we assume to dispose of N measurement samples of the system:

$$\left\{ (U^{(n)}, \lambda_1^{(n)}, \ldots, \lambda_R^{(n)}), \quad n = 1, \ldots, N \right\} \tag{3}$$

Thus, we need to fit the model with such data and get an estimation of the service times S_r.

4. Data fitting issues

According to the Utilization Law (2), if there are no requests entering a system the utilization of that system should be zero. In a real architecture this is not always the case, indeed a residual utilization can be observed. For instance, this residual is due to (i) batch processes, (ii) operating system activities, (iii) non-modeled workload classes and (iv) other several factors that influence the relationship between U, λ_r and S_r. For this reason, the law describing the system utilization in a real architecture should be rewritten as:

$$U = \sum_{r=1}^{R} \lambda_r S_r + U_0 \tag{4}$$

where U_0 is the *residual utilization*.

Stating the linear relationship (4), a linear regression method may be used in order to estimate the S_r values and the residual utilization. We call the set of activities generating U_0 as *secondary activities*.

The least-squares method (LS), also known as Model I regression analysis, is the most commonly used statistical technique to estimate the slope and intercept of linearly related data. Let us assume that N simultaneous samples (3) of cpu utilization and average arrival rate have been collected during an observation period. Estimations for the mean service times $\hat{S}_r$ and the average residual utilization $\hat{U}_0$ can be determined by computing a least-squares solution [4] for the overdetermined linear system of N equations and $R+1$ unknowns

$$U^{(n)} = \sum_{r=1}^{R} \lambda_r^{(n)} S_r + U_0^{(n)} + e^{(n)} \quad n = 1, \ldots, N \tag{5}$$

The term

$$e^{(n)} = U^{(n)} - \left(\sum_{r=1}^{R} \lambda_r^{(n)} \hat{S}_r + \hat{U}_0 \right) \tag{6}$$

is conventionally called *residual* and it represents the estimation error between the observation n and the model. It is assumed to be independent of $\lambda_r^{(n)}$. The Gauss-Markov theorem [16] states that the least-squares estimates are optimal if the residuals have expectation zero and they are independent and identically distributed [8]. From now on, we will assume $E[e^{(n)}] = 0$, being E the expectation operator. Moreover, under the hypothesis that the coefficient matrix of the linear system is well-conditioned, we assume that (5) has a unique solution.

The residual utilization $U_0^{(n)}$ is implicitly included in the measurements of $U^{(n)}$. It holds $E[U_0^{(n)}] = U_0$. It represents the system features which are not captured by the performance model, e.g., the operating system accessory activities or the workload classes out of the R representative workloads.

LS minimizes the squared sum of residuals (SSR):

$$SSR = \sum_{n=1}^{N} \left[e^{(n)} \right]^2 \tag{7}$$

Given a least squares solution $(\widehat{U}_0, \widehat{S}_1, \ldots, \widehat{S}_R)$, analytic formulas for confidence intervals and prediction bands are well-known (see e.g., [8]).

4.1. Pitfalls in parameter estimation

Different drawbacks raise out when performing LS regression on real data (Figure 1):

(a) Presence of outliers

(b) Configuration discontinuity of software/hardware

(c) Configuration discontinuity of secondary activities

(d) Service switch

(e) Dependences among workloads

(f) Unacceptable parameter estimation

The first problem arises when there are anomalous utilization measurements. Such anomalies arise because of measurement errors due to short-duration secondary activities. They can result into two different effects:

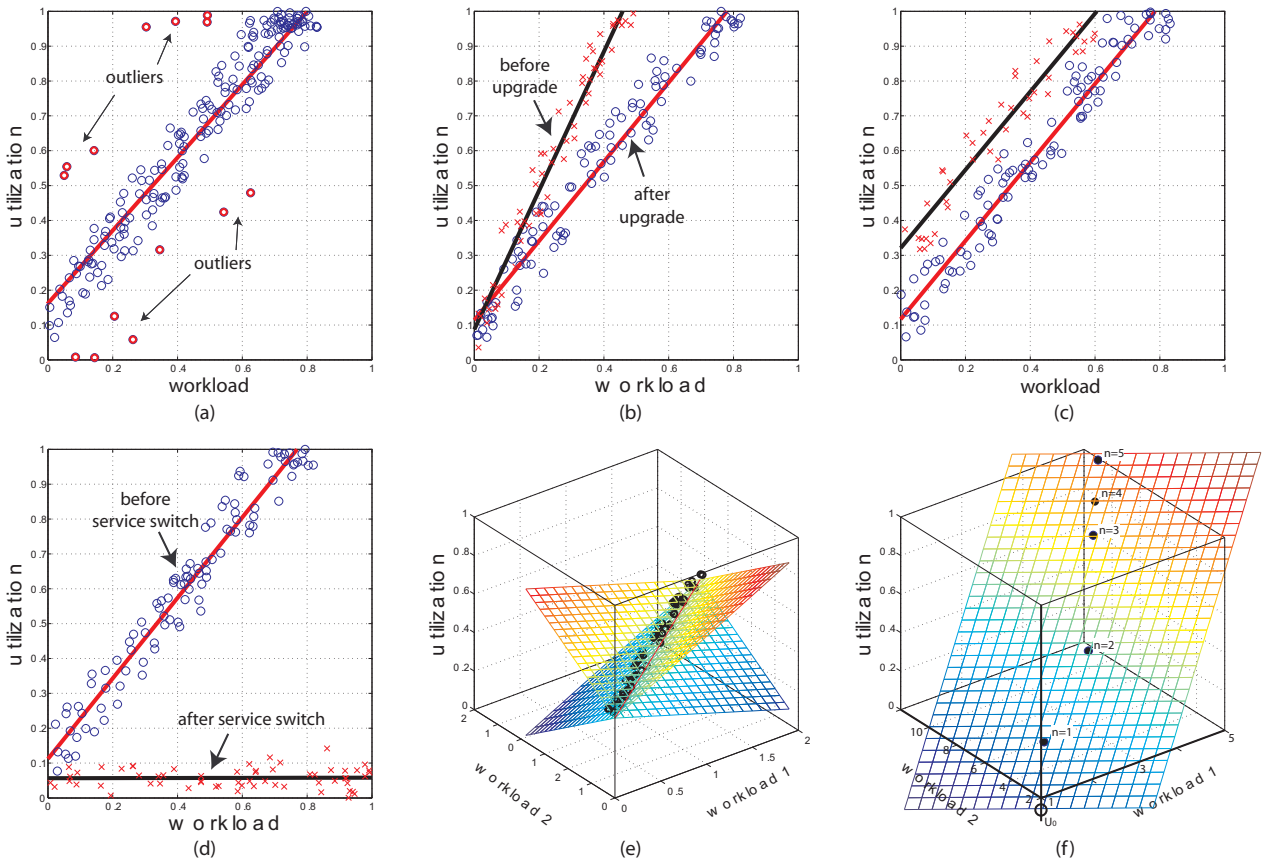


Figure 1. Pitfalls of Least Square Regression on real data: presence of outliers (a), server upgrade (b), secondary-activity discontinuity (c), service switch (d), workload collinearity (e), negative estimation (f).

noise in the measurement and presence of outliers in data. The noise represents collected samples where the secondary activities lightly impact the utilization. While the noise marginally affects the parameter estimation, the presence of outliers strongly skews the LS regression. That is because if the distribution of the outliers is skewed, the LS estimates can be biased (Figure 1(a)). Outlier filtering technique or robust regression should be considered.

The second problem arises because of configuration discontinuities. While monitoring a resource to acquire utilization and workload measurements, system may be affected by modifications in its composition. Such variations include hardware upgrades and software updates, which impact the system performance, resulting in altered measurements. For instance, Figure 1(b) shows the cpu upgrade of a server, which results in a lower service time. Discontinuities can also originate from residual utilization, e.g., a different contribute of accessory activities or the presence of a new workload not included in the R classes describing the system (Figure 1(c)). In both cases, the analysis should better be performed distinctly for each time slot preserving a stable configuration. Figure 2 describes a scenario with two discontinuities. By automatic detecting such changes, we can write three separated models of the system.

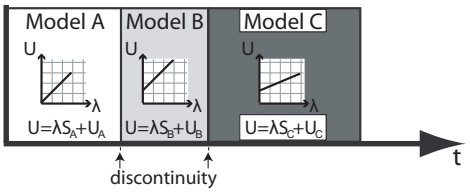


Figure 2. Configuration discontinuities

A similar effect derives from a service switch, i.e., the migration of a service from a system to another. For instance, we can have a transitory situation due to the failure of a machine, and the consequent transfer of an application to a different one, resulting in modified workloads and utilizations of both the involved machines. Figure 1(d) shows the utilization laws both before and after the service switch. Again, it is important to reveal such discontinuities to properly estimate the required parameters.

Furthermore, real systems do face with a multitude of service classes. Typically, only few of them are meaningful to describe the system behavior. Hence, reducing the number of classes can be useful for a correct analysis, being the model easier to understand, explain and parameterize. Moreover, workloads may present almost-linear dependences (near multicollinearity) among themselves, so resulting into an ill-conditioned LS problem and hence the solution is not unique, as illustrated in Figure 1(e). As an example, let consider a web server and two workload classes: workload A represents the incoming HTTP requests, while workload B the visits to HTML pages. Typically, a web page is composed by several objects and it requires multiple HTTP requests. The rate between page visits and HTTP requests can be considered almost constant on average, thus these two workloads result near multicollinear. Removing the meaningless classes can help in preventing such issue.

In addition, when there are several workload classes, it is possible that the LS algorithm returns negative estimates of service times and residual utilization (Figure 1(f)). This may happen because of noise in data, or because of the penalization of some classes having small arrival rates. We will see in Section 6 an example. Nevertheless, this is an unfeasible estimation for such metric.

5. Robust regression

In this section we cope with problems (a)-(d) shown in Section 4.1. Points (e) and (f) will be addressed in Section 6. LS may be affected by the presence of numerous outliers, thus resulting in skewed parameter estimations (problem (a)). Assuming that just a negligible part of collected observations can not be explained by the linear model, we can under-weighing such data in the regression in order to strengthen the contribution of those samples which are correctly explained. An interesting solution is represented by the robust Least Trimmed Squares regression (LTS) [19]. LTS minimizes the merit expression:

$$\sum_{n=1}^{c_N} \left[e^{(n)}\right]^2 \tag{8}$$

where c_N is called estimator covering. Thus, LTS takes into consideration $N/2 \leq c_N \leq N$ samples, ignoring the remaining observations in the regression. Note that, with $c_N = N$, (8) falls back in the LS case (7).

The effectiveness of LTS is so linked to a correct choice of the c_N meaningful samples. The covering cardinality is conventionally set [6] as:

$$c_N = \left\lfloor \frac{N+R+1}{2} \right\rfloor \tag{9}$$

where R is the number of workload classes, even though it is frequently valuable to use larger values of c_N.

The main drawback in using robust estimators is represented by their computational complexity. Indeed, we need to find the optimal subset composed by c_N elements minimizing (8). Sub-optimal solutions can be found by mean of heuristic and iterative algorithms [18, 20].

We can take advantage of robust estimation properties in order to detect configuration discontinuities and service switches (problems (b)-(d)). The main idea is to construct multiple models of the system, one for each period with a stable composition (see Figure 2). For the sake of simplicity, let us consider a single-class workload, i.e., $R = 1$ and

$$U = \lambda S + U_0 \tag{10}$$

An ongoing extension of this paper is meant to describe the multi-class case. Let $\hat{S}$ and $\hat{U_0}$ be, respectively, the estimation of the service time and of the residual utilization. Note that a discontinuity may occur both in the slope S and in the intercept U_0 of the linear relationship (10).

Let suppose to dispose of a sample set $\{1, \ldots, N\}$ collected in a time period. Partitioning such dataset into two subsets, namely $N_A = \{1, \ldots, \bar{N}\}$ and $N_B = \{\bar{N}+1, \ldots, N\}$, with $N/2 < \bar{N} \leq N$ so that $|N_A| > |N_B|$, we can define two separate models with different utilization laws, respectively, $U = \lambda S_A + U_A$ for samples in N_A and $U = \lambda S_B + U_B$ for samples in N_B.

Let first analyze an eventual discontinuity in service time, besides assuming $U_A = U_B = U_0$. We can observe that, when there is a discontinuity in S, i.e., $S_A \neq S_B$, LTS correctly estimates the service time for the greatest observation set N_A: $\hat{S} \cong S_A$. Thus, the residuals for a sample n belonging to N_A results

$$e^{(n)} \cong U_0^{(n)} - \hat{U_0} \tag{11}$$

On the other hand, evaluating the residuals for samples in N_B, they result

$$e^{(n)} \cong (S_B - \hat{S})\lambda^{(n)} + (U_0^{(n)} - \hat{U_0}) \tag{12}$$

We can note that (11) does not depend on workload, while (12) does. Hence, we define the Pearson correlation coefficient between residuals and workload as

$$\rho(\lambda, e^{(n)}) = \frac{cov(\lambda, e^{(n)})}{\sigma_\lambda \sigma_{e^{(n)}}} \tag{13}$$

where $cov(X, Y)$ is the covariance between X and Y and σ_X is the standard deviation of variate X. Whether $\rho(\lambda, e^{(n)})$ is almost null we are facing with samples in N_A, while with $|\rho(\lambda, e^{(n)})| \cong 1$ we are considering samples in N_B. Because of the above considerations, we can

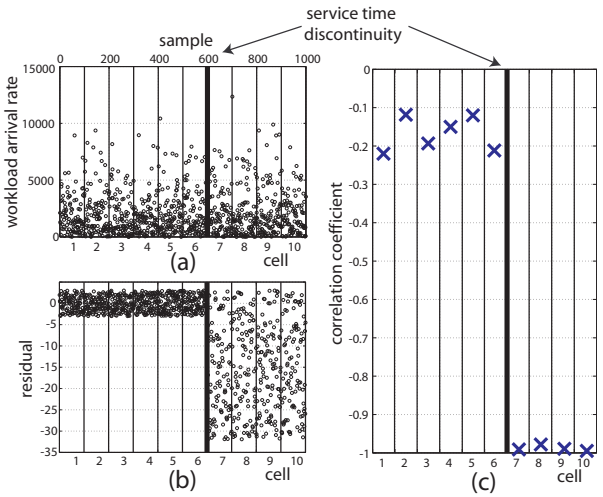


Figure 3. Example of discontinuity analysis on $N = 1000$ samples, partitioned in 10 cells. We plot both the workload arrival rate (a) and the residual (b) of each sample, and the respective correlation coefficient for each cell (c).

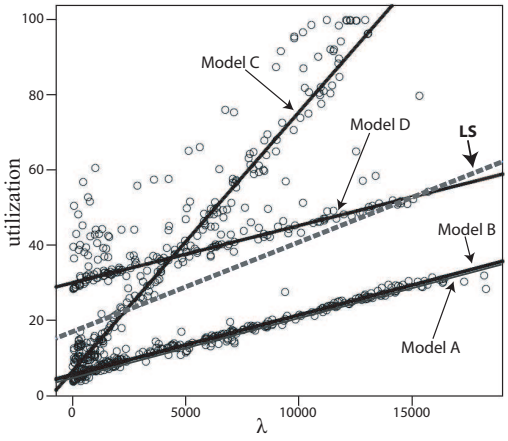


Figure 4. Robust regression: captured models (i)

infer a service time discontinuity because it induces a discontinuity in $\rho(\lambda, e^{(n)})$.

We can similarly proceed to verify the presence of discontinuities in the residual utilization, now supposing $S_A = S_B$. Applying LTS, we obtain a good estimation of samples in the largest period:

$$\hat{U_0} \cong E[U_A^{(n)}] = U_A \tag{14}$$

where $U_A^{(n)}$ is the residual utilization that we implicitly measure for observations in N_A. Reminding $|N_A| > |N_B|$, the expectation of residuals for samples in N_A is:

$$E\left[e^{(n)}\right] \cong E\left[U_A^{(n)}\right] - E\left[U_A\right] = U_A - U_A = 0 \tag{15}$$

while, for samples in N_B, it results:

$$E\left[e^{(n)}\right] \cong E\left[U_B^{(n)}\right] - E\left[U_A\right] = U_B - U_A \neq 0 \tag{16}$$

Consequently, discontinuities in U_0 can be determined by mean of a statistical test on the null hypothesis about the expectation of residuals. Stated the Central Limit Theorem, we can assume that the sample mean of the residuals from N observations follows a normal distribution. Once fixed a confidence interval, whether the statistical test on $E[e^{(n)}] = 0$ fails we can refuse the null hypothesis and state that $e^{(n)}$ has non-null average. Observe that, if the test is positive, we are not able to state anything about $E[e^{(n)}]$.

An algorithm has been created that uses (13), (15) and (16) in order to solve the regression problem in presence of discontinuities. The idea is to partition iteratively the data into different contiguous subsets and to compute the necessary tests (correlation coefficient and null-hypothesis) on each of them in order to identify contiguous zones corresponding to the various performance models.

Figure 3 graphically shows the algorithm behavior. Let us assume we have 1000 observations. We can partition the data into 10 cells, each one composed by 100 samples. We plot (a) the workload arrival rate $\lambda^{(n)}$ and (b) the residual $e^{(n)}$ of each sample $n = 1, \ldots, 1000$. Then, we compute the correlation coefficients (13) between residuals and workload for each cell and we plot the result in Figure 3(c). By observing the different correlation in cells 1–6 against cells 7–10, we can state that there is a discontinuity in sample 600. This is unquestionably an ideal case, since the discontinuity falls exactly between cell 6 and 7. In most cases it will fall in the middle of a cell, for instance cell K, thus it is required to perform an additional analysis by further partitioning K and the cells adjacent to it in order to capture the exact discontinuity point.

5.1. LTS evaluation

In this section we evaluate the accuracy of LTS and the related algorithm presented in Section 5. We collected samples from two real systems affected by different configuration discontinuities. The systems serve a single-class workload, so utilization law is described by (10).

A first example (i) is given in Figure 4. We collected 660 measurements from a web server serving http-based requests. The metrics were collected from the public web site of one of the largest mobile phone operators in Europe. We can visually individuate three models in the server under analysis. The algorithm described in Section 5 identifies four linear models. Table 1 reports the estimated parameters for such models. Discontinuities

Model	samples	$\hat{S}[job/ms]$	$\hat{U}_0$
A	$1-185$	1.59	5.4%
B	$186-299$	1.58	5.6%
C	$300-491$	6.90	6.5%
D	$540-656$	1.52	30.0%

Table 1. Estimates with robust regression (i)

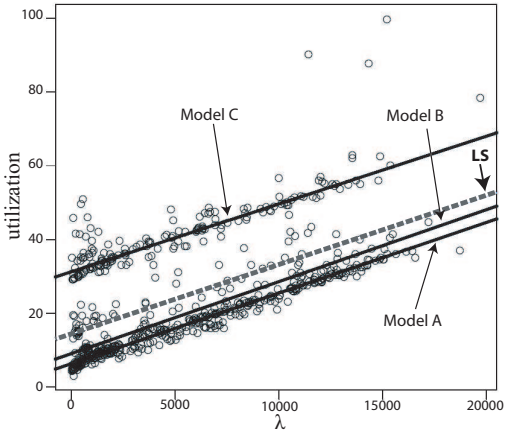


Figure 5. Robust regression: captured models (ii)

are caught at sample 186, 300 and 540. Note that some samples (about 8%) are not explained by any model because they are infact outliers. Although the algorithm individuates four models instead of the actual three, both plot and Table 1 show that two models are in substance overlapped. We estimate the accuracy of the models defining the error $\delta = E[|e^{(n)}|]$, which results about 3%. We plot for comparison also the model obtained by LS regression. As expected, in this case LS is quite inefficient ($\delta \cong 15\%$), since it is not able to identify multiple models.

A second example (ii) on real data is reported in Figure 5. We collected 665 measurements from another server of the aforementioned phone operator IT system. During the observation, the server was affected by an increase of secondary activities. For instance, this is the typical case concerning the installation of a new service in the machine. The algorithm detects the presence of three models, whose parameters are shown in

Model	samples	$\hat{S}[job/ms]$	$\hat{U}_0$
A	$1-275$	1.91	6.3%
B	$276-478$	1.95	9.0%
C	$479-661$	1.84	31.2%

Table 2. Estimates with robust regression (ii)

Data	Algo	$E[e^{(n)}]$	δ	$\sigma_{e^{(n)}}$
(i)	LS	$7.9 \cdot 10^{-15}\%$	15.6%	18.8%
	Robust	$9.9 \cdot 10^{-1}\%$	3%	7.2%
(ii)	LS	$-5.9 \cdot 10^{-15}\%$	10%	11.8%
	Robust	$8 \cdot 10^{-2}\%$	3.3%	6.5%

Table 3. Comparison between LS and LTS-based algorithm on examples (i) and (ii).

n	$\lambda_1^{(n)}$ $[job/sec]$	$\lambda_2^{(n)}$ $[job/sec]$	$U^{(n)}$
1	1	1.8	0.18%
2	2	4.3	0.40%
3	3	6.0	0.60%
4	4	7.5	0.80%
5	5	11.0	0.95%

Table 4. Example regression model with unacceptable parameter estimation

Table 2. Again, it lightly overestimates the actual number of models. The three models explain almost all the data (99.5%). In this case δ is about 3.3%. Figure 5 reports also the LS regression model, with $\delta \cong 10\%$.

Finally, Table 3 summarizes the results for both the above examples. It reports, distinctly for the two tested methodologies, i.e., the least-squares and the robust regression-based algorithm, the average of residuals $E[e^{(n)}]$, the expectation of absolute residuals δ and the standard deviation of residuals $\sigma_{e^{(n)}}$.

6. Workload Reduction

We observed in points (e) and (f) of Section 4.1 that multi-class workloads can be affected by negative parameter estimation and workload dependencies. Let consider a small example. Referring to Figure 1(f), we assume that during an observation of a system with $R = 2$ workloads we collected the $N = 5$ samples given in Table 4. If we fit these data using a LS algorithm, we find the following unfeasible values:

$$\hat{U}_0 = -0.006\%, \quad \hat{S}_1 = 0.251, \quad \hat{S}_2 = -0.027 \quad (17)$$

with $SSR \cong 8 \cdot 10^{-4}$. Notice that LS returns negative values for $\hat{U}_0$ and $\hat{S}_2$. A natural solution to this situation is to reformulate the service time estimation as a more general optimization problem. We define a constrained regression, known as NNLS (Non-Negative Least-Squares) [10], which still minimizes (7), but also specifies additional bounds for the service

times and the residual utilization, so that, respectively, $S_r \in [0, \infty), r = 1, \ldots, R$ and $U_0 \in [0, 1]$.

For instance, the NNLS solution of the example model considered above becomes

$$\hat{U_0} = 0.004\%, \quad \hat{S_1} = 0.194, \quad \hat{S_2} = 0.000 \qquad (18)$$

with $SSR \cong 15 \cdot 10^{-4}$. Therefore, with a small degradation of the squared sum of residuals, we have granted the feasibility of the result and we have understood that the workload λ_2 may be omitted.

Moreover, note that class-1 and class-2 workloads are almost dependent among themselves. In fact, $\lambda_2 \cong 2\lambda_1$. This means that the LS solution (17) is not likely to be unique. For instance, we can refer to a web server where class-1 jobs represent page views, while class-2 jobs are the HTTP requests. Thus, by mean of NNLS, we rule out some meaningless workloads, forcing them to be null, and we are also contributing in discarding dependent workloads. Anyway, NNLS could not completely remove all collinear workloads. It might be required to detect them in advance by analyzing the correlation coefficient among workloads.

6.1. NNLS evaluation

In this section we investigate the effectiveness of the NNLS regression method by analyzing the workload of five applications (referred to as AP1,..., AP5) in the data center of a large European e-commerce enterprise. The data were collected using the Caplan capacity planning tool [15]. The tool was configured to log application performance for several weeks, collecting every hour a new sample of the average system total utilization at each system and of application workloads. For each measure, we collected 1500 samples. During the observation period, the systems did not experience major hardware or software upgrades.

We collected four potential business metrics:

- WL1: number of Web pages requested by clients.
- WL2: number of registered user logins.
- WL3: number of purchased orders.
- WL4: number of credit card transactions.

The diagrams in Figure 6 plot the hourly workload samples as a function of time. The leftmost dashed vertical line limits the first 1-week of data. The 1-month portion is delimited by the next dashed vertical line. These two subsets are used in separate estimation experiments and from now on are referred to as the *training sets*. For instance, the 1-week training set includes all samples numbered from 1 to 168. The interval 721-1500 is instead the 780-samples (approximately one month) *validation set* that is used to assess prediction accuracy of the models defined with the parameters estimated from the training sets.

	WL1	WL2	WL3	WL4
WL1	1.00	0.87	0.98	0.90
WL2	0.87	1.00	0.91	0.88
WL3	0.98	0.91	1.00	0.95
WL4	0.90	0.88	0.95	1.00

Table 5. Workload-workload correlation matrix for the 1-month training set.

We have no exact information about how the workloads are processed by the different applications. Thus, we have little overall information about workloads and we need workload selection. In addition, from Table 5 which shows the workload-workload correlations, we see that the four workloads are highly correlated with each other. Thus it is very difficult to select the most significant classes. Summing up, this considered case study is indeed a stress case for workload selection methods.

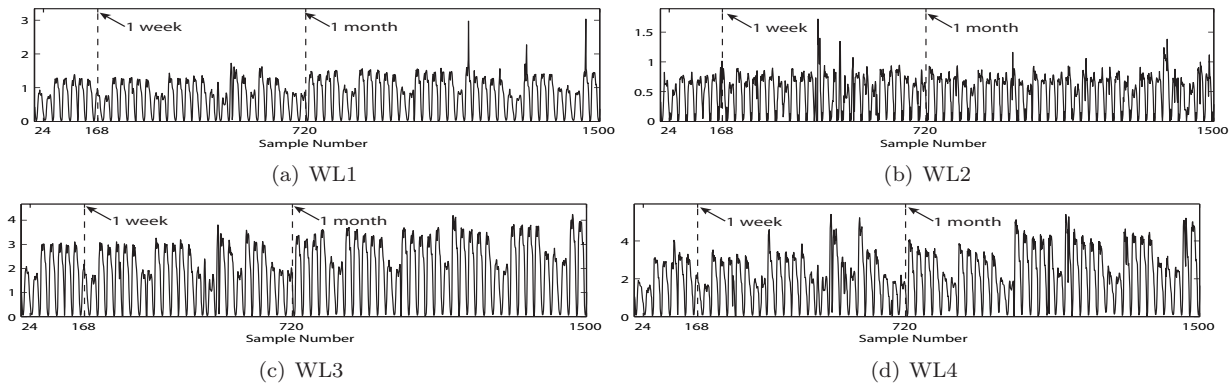
We organized the experimental comparison as follows. We estimated service times using two different training sets (1-week and 1-month) for each considered system. Let $(\widehat{S}_{WL1}, \ldots, \widehat{S}_{WL4})$ be the set of estimated service times for the four workload classes, and let $\widehat{U}_0$ be the estimated residual utilization. After each estimation, we considered the (approximately one month) validation set composed by the samples indexed by $n \in [721, 1500]$, and we computed the following difference

$$A^{(n)} = \left| U^{(n)} - \left(\sum_r \widehat{S}_r \lambda_r^{(n)} + \widehat{U}_0 \right) \right|, \quad n = 721, \ldots, 1500$$

where the summation is on $r \in \{WL1, \ldots, WL4\}$. This quantity is the absolute difference between each measured utilization in the validation set and the corresponding value predicted by the utilization model. Table 6 shows the mean and standard deviations for A_n. The LS columns refer to the ordinary least-squares linear regression; the NNLS columns are for the positively constrained linear regression. Furthermore, for each experiment, we report in Table 7 the sign of the estimated service times, and we explicit which classes are selected by each method. This is important to understand the actual choices performed by each algorithm and their sensitivity with respect to the training sets. Empty cells indicate that the estimated service times are equal to zero. Grey cells, instead, indicate differences between the 1-week and 1-month training sets.

In the following paragraphs, we comment in details the results for the two considered training sets.

Results for the 1-Month Training Set. We begin by considering the 1-month training set columns in Table 6. Looking at Table 7, we immediately see that predic-



(a) WL1 (b) WL2

(c) WL3 (d) WL4

Figure 6. Application Workloads.

tion accuracy $A^{(n)}$ is not a good indicator of the overall quality of the estimation: LS frequently returns negative service times. Concerning the workload selection task, we immediately see that NNLS reduces the number of selected workloads with respect to LS, and, the computational times for both techniques are negligible. The accuracy of NNLS with respect to the training set needs to be slightly less than that of LS, since SSR is always maximized by LS, but in general this has typically no effect in practice and it is fully compensated by the modeling gains of NNLS. For example, a basic closed queueing network model [11] composed by a queue modeling the Web application AP1 and by a delay modeling customer's activity, and populated by 100 requests for each class, can be solved with $\approx 10^8$ operations if the classes are four (LS), but with only $\approx 10^4$ operations if the classes are two (NNLS), with a net-gain of four orders of magnitude and negligible accuracy decrease. Similar savings apply to memory requirements.

Results for the 1*-Week Training Set.* The results for the 1-week training set, in terms of accuracy, number of selected workloads and feasibility of results, are very similar to those obtained for the 1-month set. Therefore, similar consideration apply. Concerning the workload selection problem, we see from Table 7 that NNLS provides estimates that are consistent over time.

Training Set Size : 1-month (720 samples)				
	LS		NNLS	
App	mean	std dev	mean	std dev
AP1	0.024	0.026	0.018	0.023
AP2	0.016	0.017	0.016	0.017
AP3	0.110	0.064	0.100	0.060
AP4	0.015	0.015	0.015	0.015
AP5	0.023	0.025	0.022	0.025
Training Set Size : 1-week (169 samples)				
AP1	0.037	0.038	0.024	0.025
AP2	0.020	0.020	0.016	0.018
AP3	0.130	0.074	0.140	0.074
AP4	0.017	0.014	0.017	0.014
AP5	0.030	0.026	0.030	0.026

Table 6. Evaluation of NNLS.

7. Conclusions

In this paper we face with parameter estimation in queueing network performance models. The idea is to use aggregate measurements (system throughput and utilization of the servers), typically easy to retrieve from log files, and estimate the service times.

A correct service time estimation requires, among other things, the selection of the workloads which better describe the system behavior. By mean of a constrained regression, well-known in statistics as Non-Negative Least-Squares, we both discard the less significant workloads and we avoid unacceptable parameter estimates (e.g., negative service times).

Other difficulties arise when performing the estimation on real data. For instance, the presence of accessory activities may affect the measurements. In addition, nowadays IT systems change their configuration (e.g., software update) in a faster rate than in the past, which results in discontinuities in the queueing model. We provide a robust regression-based methodology in order to discard the anomalous measurements and to determine the changes in system configuration. Both the estimation techniques were successfully tested on real systems. Further developments of this work are meant to extend the robust optimization-based technique to multi-class workloads and to investigate other specific solutions for the workload selection problem (e.g., combining other methods such as clustering and principal component analysis with linear regression).

Training Set Size : 1-month (720 samples)								
	LS				NNLS			
App.	WL1	WL2	WL3	WL4	WL1	WL2	WL3	WL4
AP1	−	−	+	+			+	+
AP2	+	−	+	+			+	+
AP3	+	−	+	+		+		+
AP4	+	+	+	+	+	+	+	+
AP5	+	−	−	+				+
Training Set Size : 1-week (169 samples)								
AP1	−	−	+	+			+	+
AP2	−	−	+	+			+	+
AP3	−	+	+	+		+		
AP4	+	+	+	+	+	+	+	+
AP5	+	−	+	+			+	+

Table 7. Comparison of LS and NNLS techniques. Cells in gray indicate differences between 1-week and 1-month training set.

References

[1] F. Baskett, K. Chandy, R. Muntz, and F. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *J.ACM*, 22(2):248–260, 1975.

[2] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains*. John Wiley and Sons, 1998.

[3] W. C. Cheng and R. R. Muntz. Bounding errors introduced by clustering of customers in closed product-form queuing networks. *J.ACM*, 43(4):641–669, 1996.

[4] G. H. Golub and C. F. V. Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[5] W. J. Gordon and G. F. Newell. Closed queueing systems with exponential servers. *Operations Research*, 15(2):254–265, 1967.

[6] D. Hawkins and D. Olive. Applications and algorithms for least trimmed sum of absolute deviations regression. *Comp. Stat. Data Anal.*, 32(2):119–134, 1999.

[7] J. R. J. Jackson. Jobshop-like queueing systems. *Management Science*, 10(1):131–142, 1963.

[8] R. K. Jain. *The Art of Computer Systems Performance Analysis*. Wiley, 1991.

[9] L. Kleinrock. *Queueing Systems - Volume 1: Theory*. John Wiley and Sons, 1975.

[10] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Prentice-Hall, 1974.

[11] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik. *Quantitative System Performance*. Prentice-Hall, 1984.

[12] J. D. C. Little. A proof of the queueing formula $L = \lambda W$. *Operations Research*, pages 9:383–387, 1961.

[13] Z. Liu, L. Wynter, C. H. Xia, and F. Zhang. Parameter inference of queueing models for it systems using end-to-end measurements. *Perf. Eval.*, 63(1):36–60, 2006.

[14] D. Menasce and V. A. F. Almeida. *Capacity Planning for Web Performance: Metrics, Models, and Methods*. Prentice Hall, 1998.

[15] Neptuny. *Caplan 2.0 User Manual. http://www.neptuny.it/index.jsp?id=caplan*, Milan, Italy, 2005.

[16] R. L. Plackett. Some theorems in least squares. *Biometrika*, 37:149–157, 1950.

[17] J. Rolia and V. Vetland. Parameter estimation for performance models of distributed application systems. In *CASCON '95*, page 54. IBM Press, 1995.

[18] P. J. Rousseeuw and K. Driessen. Computing lts regression for large data sets. *Data Min. Knowl. Discov.*, 12(1):29–45, 2006.

[19] P. J. Rousseeuw and A. M. Leroy. *Robust regression and outlier detection*. John Wiley & Sons, Inc., New York, NY, USA, 1987.

[20] P. J. Rousseeuw and S. Verboven. Robust estimation in very small samples. *Comput. Stat. Data Anal.*, 40(4):741–758, 2002.

[21] V. Sharma and R. Mazumdar. Estimating traffic parameters in queueing systems with local information. *Perform. Eval.*, 32(3):217–230, 1998.

[22] H. Shen and L. D. Brown. Non-parametric modelling of time-varying customer service times at a bank call centre. *Appl. Stoch. Models Bus. Ind.*, 22:297–311, 2006.

[23] L. Wynter, C. H. Xia, and F. Zhang. Parameter inference of queueing models for IT systems using end-to-end measurements. In *Proc. of the 2004 ACM SIGMETRICS/PERF. joint Int'l Conf.*, pages 408–409, 2004.

[24] L. Zhang, C. H. Xia, M. S. Squillante, and W. N. M. Iii. Workload service requirements analysis: A queueing network optimization approach. In *MASCOTS '02: Proc. of the 10th IEEE Internat. Symposium on MASCOTS*, page 23, Washington, DC, USA, 2002. IEEE.

[25] T. Zheng, J. Yang, M. Woodside, M. Litoiu, and G. Iszlai. Tracking time-varying parameters in software systems with extended kalman filters. In *CASCON '05: Proc. of 2005 conf. of the Centre for Advanced Studies on Collaborative research*, pages 334–345. IBM Press, 2005.