

Robust Workload Estimation in Queueing Network Performance Models

Giuliano Casale*
casale@neptuny.it

Paolo Cremonesi†
paolo.cremonesi@polimi.it

Roberto Turrin†
roberto.turrin@polimi.it

Abstract

Traditional approaches for capacity planning are based on queueing network models. However, modeling with queueing networks requires the knowledge of the service demands of each class of workloads at each device described in the model. In real systems, such service demands can be very difficult to measure.

In this paper, we present an optimization-based technique to address the problem. The technique is formulated as a robust linear parameter estimation that can be used with both closed and open queueing network models. We consider the case where aggregate measurements (throughput and utilization) are available. Such measurements are typically much easier to obtain than the service demands. We present experimental results which prove the effectiveness of the constrained and robust linear estimation.

1. Introduction

Capacity planning activities are traditionally performed using simulation or analytical models such as queueing systems or network of queues [1, 3]. Nevertheless, queueing model parametrization can be a challenging task. It involves: (i) identifying the meaningful business workload and (ii) estimating the service time placed by requests at each resource.

The biggest issue is the correct estimation of the service times (i.e., the mean 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 or very intrusive.

To address this challenging topic, we propose service time estimation techniques based on robust and constrained optimization. We consider the case where aggregate measurements (application throughput and server utilization) are available. Note that such measurements are easy to collect.

The estimation problem is complicated because of outliers 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).

There are few works directly addressing the service time estimation problem. An early work on queueing network service time estimation can be found in [9], where Rolia and Vetlan use multivariate linear least-squares regression to compute the service times of distributed software systems. Sharma and Mazumdar in [11] estimate service times and arrival rates in an open queueing network by injecting a test workload in the real system and by monitoring the time behavior of the test workload. Liu et al. in [7] estimate service times in open queueing networks by combining utilization, throughput and end-to-end response time measurements with a quadratic programming techniques. An estimation technique based on polynomial regression is presented in [12]. A technique for time-varying parameters of queueing models is proposed in [13]. 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 [2]. 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).

In the following we define the service time estimation problem and the relative issues. Therefore we present two estimations techniques and we prove their effectiveness.

2. Queueing network models

Queueing network models depend on a number of parameters, the most important being the service times of each station, i.e., the time required for a station to process one request (e.g., an I/O operation) when no other requests are processing. Such parameters can be in line of principle be derived from system log files, even if they present several practical

*Neptuny - Via Durando, 10 - 20158 Milan, Italy

†Politecnico di Milano - Via Ponzio 34/5 - 20133 Milan, Italy

issues. Most 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 production systems.

Because of these problems, cpu utilization and workload information are usually the only available performance metrics. In the rest of the paper we show how to estimate service times from these two metrics, by using the utilization law:

$$U = \sum_{r=1}^R \lambda_r S_r \quad (1)$$

where r is the number of workload classes, and S_r and λ_r are, respectively, the service time and the mean arrival rate of class- r requests. We will see in Section 3 that it is difficult to select the workloads which best describe the system performance. By having a correct definition of the R classes and accurate estimation of S_r , it is possible to predict the future utilization levels in presence of a certain input workload. Further, for the popular class of open product-form networks [1], 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.

According to the utilization law (1), 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:

$$U = \sum_{r=1}^R \lambda_r S_r + U_0 \quad (2)$$

where U_0 is the residual utilization. We call the set of activities generating U_0 as *secondary activities*.

3. Data fitting issues

In order to parameterize (1), we assume to dispose of N measurement samples of the system:

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

Thus, we need to fit the model with such data and get an estimation of the service times S_r . Stating the linear relationship (2), a linear regression method may be used in order to estimate the S_r values and the residual utilization.

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 LS solution [4] for the overdetermined linear system of N equations and $R + 1$ unknowns

$$U^{(n)} = \sum_{r=1}^R \lambda_r^{(n)} S_r + U_0^{(n)} + e^{(n)} \quad n = 1, \dots, N \quad (4)$$

The term $e^{(n)}$ is called *residual* and it represents the estimation error between the observation n and the model. It is assumed to be independent of $\lambda_r^{(n)}$. Given a least squares solution $(\hat{U}_0, \hat{S}_1, \dots, \hat{S}_R)$, analytic formulas for confidence intervals and prediction bands are well-known (see e.g., [4]).

Different drawbacks raise out when performing LS regression on real data (Figure 1): (a) presence of outliers, configuration discontinuities of (b) software/hardware or (c) secondary activities, (d) service switch, (e) dependences among workloads and (f) unacceptable parameter estimation.

The first problem happens 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: noise in the measurement and presence of outliers in data. While the noise marginally affects the parameter estimation, the presence of outliers strongly skews the LS regression (Figure 1(a)). The second problem arises because of configuration discontinuities. While monitoring a system to acquire utilization and workload measurements, the system may be affected by modifications in its composition (i.e., 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. 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 system, resulting in modified workloads and utilizations of both the involved machines.

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. 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(c). 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

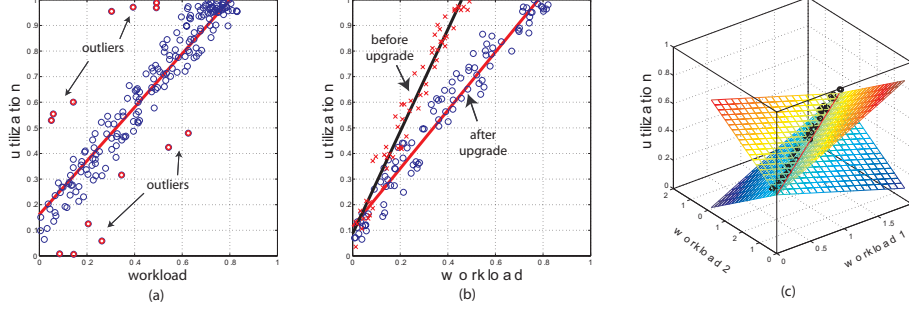


Figure 1. Pitfalls of LS on real data: (a) outliers, (b) server upgrade (c) workload collinearity.

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. This may happen because of noise in data, or because of the penalization of some classes having small arrival rates.

4. Robust regression

In this section we cope with problems (a)-(d) shown in Section 3. LS may be affected by the presence of numerous outliers, thus resulting in skewed parameter estimations. Assuming that just a negligible part of collected observations can not be explained by the linear model, we can underweighting such data in the regression in order to strengthen the contribute of those samples which are correctly explained. An extension to this idea is represented by the robust Least Trimmed Squares regression (LTS) [10]. LTS minimizes the merit expression:

$$\sum_{n=1}^{c_N} [e^{(n)}]^2 \quad (5)$$

where c_N is called estimator covering. Thus, LTS takes into consideration $N/2 \leq c_N \leq N$ samples, ignoring the remaining observations. Note that, when $c_N = N$, (5) falls back in the LS case. The effectiveness of LTS is so linked to a correct choice of the c_N meaningful samples. The covering cardinality is conventionally set [5] as:

$$c_N = \left\lfloor \frac{N + R + 1}{2} \right\rfloor \quad (6)$$

where R is the number of workload classes, even though it is frequently valuable to use larger values of c_N . We can take advantage of robust estimation properties in order to detect configuration discontinuities and service switches. The main idea is to construct multiple models of the system. For the sake of simplicity, let us consider a single-class workload and

$$U = \lambda S + U_0 \quad (7)$$

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 (7). Let suppose to dispose of a sample set $\{1, \dots, N\}$ collected in a time period. Partitioning such dataset into two subsets, namely $N_A = \{1, \dots, \bar{N}\}$ and $N_B = \{\bar{N} + 1, \dots, N\}$, 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

$$e^{(n)} \cong U_0^{(n)} - \hat{U}_0 \quad (8)$$

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

$$e^{(n)} \cong (S_B - \hat{S})\lambda^{(n)} + (U_0^{(n)} - \hat{U}_0) \quad (9)$$

We can note that (8) does not depend on workload, while (9) does. Hence, we define the Pearson correlation coefficient between residuals and workload as

$$\rho(\lambda, e^{(n)}) = \frac{\text{cov}(\lambda, e^{(n)})}{\sigma_\lambda \sigma_{e^{(n)}}} \quad (10)$$

where $\text{cov}(X, Y)$ 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 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:

$$\hat{U}_0 \cong E[U_A^{(n)}] = U_A \quad (11)$$

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:

$$E[e^{(n)}] \cong E[U_A^{(n)}] - E[U_A] = U_A - U_A = 0 \quad (12)$$

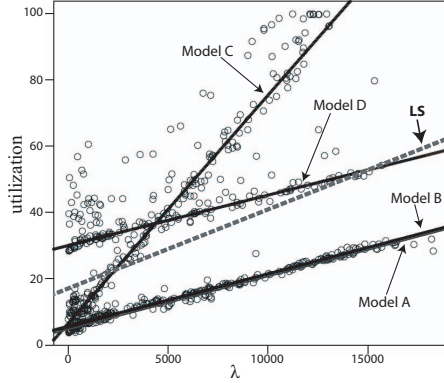


Figure 2. LTS: captured models.

Model	samples	$\hat{S}[\text{job/ms}]$	\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.

while, for samples in N_B , it results:

$$E[e^{(n)}] \cong E[U_B^{(n)}] - E[U_A] = U_B - U_A \neq 0 \quad (13)$$

Consequently, discontinuities in U_0 can be determined by mean of a statistical test on the null hypothesis about the expectation of residuals.

4.1. LTS evaluation

In this section we evaluate the accuracy of LTS. We collected samples from a real system affected by configuration discontinuities. The systems serve a single-class workload, so utilization law is described by (7).

We collected 660 measurements from a web server serving http-based requests. The metrics were collected from the public web site of one of a large European mobile phone operator. We can visually individuate three models in the server under analysis (Figure 2). The algorithm described in Section 4 identifies four linear models. Table 1 reports the estimated parameters for such models. Discontinuities are caught at sample 186, 300 and 540. Note that some samples (about 8%) are not explained by any model because they are outliers. Although the algorithm determines four models instead of the actual three, both Figure 2 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%. Table 1 shows also the result 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. Table 2 summarizes the results. It reports, distinctly for the

Algorithm	δ	$\sigma_{e^{(n)}}$
LS	15.6%	18.8%
Robust	3%	7.2%

Table 2. Comparison between LS and LTS.

two tested methodologies, i.e., the least-squares and the robust regression-based algorithm, the expectation of absolute residuals δ and the standard deviation of residuals $\sigma_{e^{(n)}}$.

5. Workload Reduction

We observed in points (e) and (f) of Section 3 that multi-class workloads can be affected by negative parameter estimation and workload dependencies. A natural solution to this situation is to reformulate the service time estimation as a more general optimization problem. We can use a constrained regression, known as NNLS (Non-Negative Least-Squares) [6], which specifies additional bounds for the service times and the residual utilization, so that, respectively, $S_r \in [0, \infty)$, $r = 1, \dots, R$ and $U_0 \in [0, 1]$.

We have investigated 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 [8]. The tool was configured to log application performance for two months, 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 business metrics: (WL1) number of Web pages requested by clients, (WL2) number of registered user logins, (WL3) number of purchased orders and (WL4) number of credit card transactions.

The dataset has been divided into a *training* and a *validation* subset. The first month of data (training) has been used to estimate the parameters. The second month of data has been used to assess prediction accuracy of the models defined with the parameters estimated from the training sets. 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 3 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 the training set. Let $(\hat{S}_{WL1}, \dots, \hat{S}_{WL4})$ be the set of estimated service times for the four workload classes, and let \hat{U}_0 be the estimated resid-

	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 3. Workload-workload correlation.

ual utilization. After the estimation, we considered the one month validation set and we computed the difference:

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

where the summation is on $r \in \{WL1, \dots, WL4\}$. This quantity is the absolute difference between each measured utilization in the validation set and the corresponding value predicted by the utilization model. Table 4 shows the mean and standard deviations for E . 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 5 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. Empty cells indicate that the estimated service times are equal to zero. Looking at Table 5, we see that prediction accuracy E is not a good indicator of the overall quality of the estimation: LS frequently returns negative service times. Concerning the workload selection task, we see that NNLS reduces the number of selected workloads with respect to LS. The accuracy of NNLS does not increase with respect to LS, but we reduce the number of workload classes used to describe the model.

6. Conclusions

In this paper we have considered 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, in order to and estimate the service times.

A correct service time estimation requires the selection of the workloads which better describe the system behavior. By mean of a constrained regression, well-known in statistics

Training Set Size : 1-month (720 samples)				
App	LS		NNLS	
	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

Table 4. Evaluation of NNLS.

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

Table 5. Comparison of LS and NNLS.

as 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 of IT systems that have changed their configuration (e.g., software update) during the monitoring period. By means of a robust regression-based methodology we are able to detect changes in system configuration.

References

- [1] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains*. John Wiley & Sons, 1998.
- [2] W. C. Cheng and R. R. Muntz. Bounding errors introduced by clustering of customers in closed product-form queueing networks. *JACM*, 43(4):641–669, 1996.
- [3] D. Menascé and V. Almeida. *Capacity Planning for Web Performance: Metrics, Models and Methods*. Prentice Hall, 1998.
- [4] G. Golub and C. F. V. Loan. *Matrix computations (3rd ed)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.
- [5] 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.
- [6] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Prentice-Hall, 1974.
- [7] 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.
- [8] Neptun. *Caplan 2.0 User Manual*. <http://www.neptun.it/index.jsp?id=caplan>, Milan, Italy, 2005.
- [9] J. Rolia and V. Vetland. Parameter estimation for performance models of distributed application systems.
- [10] P. Rousseeuw and A. Leroy. *Robust regression and outlier detection*. John Wiley & Sons, Inc., New York, USA, 1987.
- [11] V. Sharma and R. Mazumdar. Estimating traffic parameters in queueing systems with local information. *Perf. Eval.*, 32(3):217–230, 1998.
- [12] 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.
- [13] 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*, pages 334–345. IBM Press, 2005.