# Predicting the System Performance by Combining Calibrated Performance Models of its Components

## A Preliminary Study

Thomas Begin
LIP, CNRS - ENS Lyon - UCB Lyon 1 - Inria 5668
France
DIVA Lab, University of Ottawa
Canada

Alexandre Brandwajn
Baskin School of Engineering
University of California Santa Cruz
PALLAS International Corporation
San Jose, CA, USA

## ABSTRACT

In this paper we consider the problem of combining calibrated performance models of system components in order to predict overall system performance. We focus on open workload system models, in which, under certain conditions, obtaining and validating the overall system performance measures can be a simple application of Little's law. We discuss the conditions of applicability of such a simple validation methodology, including examples of successful application, as well as examples where this approach fails.

Additionally, we propose to analyze the deviations between the model predictions and system measurements, so as to decide if they correspond to "measurement noise" or if an important system component has not been correctly represented. This approach can be used as an aid in the design of validated system performance models.

## Keywords

performance models; component-level models; overall system performance; validation; calibration.

## 1. INTRODUCTION

Faced with increasingly complex system architectures, an obvious and commonly used approach is to characterize the behavior of selected system components deemed crucial to overall system performance. These performance components have then to be combined in a global system model so as to produce the desired overall system performance characterization. A number of methods have been employed to effect such combination.

For closed system models, fixed-point iterations (e.g. [18]), state-dependent equivalence (e.g. [8]), and near- decomposability (e.g. [11]) are just a very partial list of methods that were employed in this area. The resulting overall performance model must then be validated against real or simulated system performance measurements. In an open system, under certain conditions, obtaining and validating the overall system performance measures can be a simple application of Little's law.

In this paper, we focus on open workload system models, but several aspects of our discussion are applicable to closed systems as well. We discuss the conditions of applicability of such simple validation methodology using Little's law, including examples of successful application, as well as examples where this approach fails. Even if the applicability conditions are met, some degree of deviation between the model and the system measurements is expected. We propose to analyze these deviations, so as to decide if they correspond to simple "measurement noise" or if an important system component has not been correctly represented. As such, this approach can be used as an aid in the design of validated system performance models.

In the next section, we review related work. In Section 3 we discuss the simple use of Little's law in open models and its applicability conditions. Section 4 is devoted to examples of application of this methodology, and in Section 5 we discuss the use of this approach to discover and model hidden system components. Section 6 concludes this paper.

## 2. RELATED WORK

From the early days of computer system and network models, it was clear that "divide and conquer" approaches may be beneficial when dealing with complex systems. This gave rise to many flavors of decomposition approaches where different parts of a system are analyzed in isolation (X-model [14], Kühn [15], near- decomposability [11], Norton equivalent [10, 4], equivalence and decomposition, etc). For instance, it is usually considerably easier to analyze the I/O subsystem separately from the complexity of CPU priority scheduling when analyzing the performance of a computer system.

Of course, we need a way to combine the results of the analysis of decomposed subsystems to obtain the desired overall system performance metrics. Depending on the decomposition approach used, such "assembly" of the decomposed models may be accomplished by replacing a whole subsystem by a simple delay representing the average time a task spends in the subsystem (X-model). In other approaches, the performance of a subsystem may be represented as a state-dependent server (Norton, near- decomposability, equivalence and decomposition). As shown in [9], many of these approaches may be viewed in a unified way as relying (implicitly or explicitly) on the use of marginal and conditional probabilities, which provides, at least in theory,



*ICPE'16, March 12–18, 2016, Delft, Netherlands.*



DOI: http://dx.doi.org/10.1145/2851553.2858658

a clean way to assemble the results of the analysis of decomposed subsystems.

Present-day systems often comprise a number of components that work together to process incoming requests. As mentioned above, it may be easier to model the performance of each individual component than directly that of the whole system. The component modeling may be performed using a constructive approach requiring knowledge and expertise, or using a "black box" approach, which parameterizes predefined models by observing the relationships between the component input and output parameters [6, 2, 3]. Several approaches have been proposed in the literature for combining component-level models into system-level performance.

In the context of disk I/O requests, a case in point is the work done by Ganger and Patt [12]. At that time, accurate and sound models have been proposed for I/O subsystem performance. But it was unclear how the improvements of subsystems will be reflected in the overall system performance. Among other things, the authors stress that looking to improve the overall system performance is not directly the same as improving the I/O subsystem performance. This is because composing the performance of subsystems is not always straightforward.

The issue of predicting performance of a system based on the behavior of its components has also been addressed in the area of autonomic systems. Harbaoui et al. have proposed a framework to predict the performance of a target configuration when planning a system reconfiguration [13]. They decompose a distributed application into black boxes, identifying the queue model for each black box and assembling these models into a queueing network according to the candidate target configuration.

More recently, Kraft et al. have studied the response times experienced by disk I/O requests in consolidated virtualized environments [16, 17]. The authors have shown how to extrapolate the model of a single Virtual Machine (VM) into a model to predict the degree of contention when multiple VMs are accessing a remote storage server.

These previous works have emphasized the need and proposed a specific-area solution to the problem of combining models of components to represent the performance of a system. In this paper, we attempt to propose a more general framework for this problem.

# 3. FROM LOCAL TO GLOBAL SYSTEM PERFORMANCE

We concentrate on systems in which requests (tasks, transactions, jobs) arrive from an outside source, are processed by the system and eventually depart from the system. We refer to such systems as "open systems".

The system considered consists of $K$ known components (see Figure 1) that are deemed important with respect to a specific average performance measure $F$ (e.g. mean number of requests in system ($L$), mean response time ($R$) or loss probability ($P$)). We assume that we have a set of $I$ system-level measurement points $\{x_S^{exp}, f_S^{exp}\}, i = 1, \ldots, I$, where $x_S^{exp}$ denotes the mean measured system throughput, i.e., the number of requests successfully processed by the system per time unit, and $f_S^{exp}$ is the corresponding value of the selected performance measure. We use the superscript *exp* to refer to measured values and the superscript *mod* to denote values obtained from a model. We assume that calibrated models have been developed for each of the $K$ system

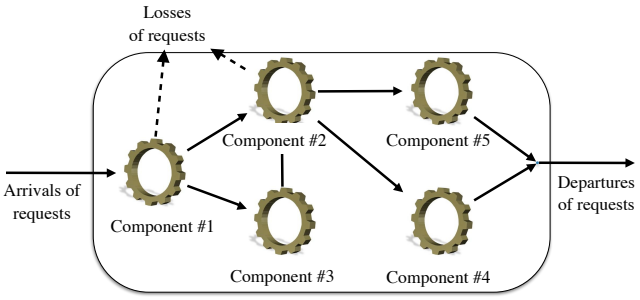



Figure 1: A system with $K = 5$ components.

components. Our $K$ component models can be viewed as a set of $K$ functions $f_k^{mod}(x_k^{mod})$ with $k = 1, \ldots, K$.

In order to be able to use our component models to assess overall system performance, we must know the relationship between the overall system throughput $X_S$ and the component throughputs $X_k$, $k = 1, \ldots, K$. This is an essential assumption. In practice, this relationship will be known from the nature of the system or from measurements. Another essential assumption is that a given request occupies a single component at a time.

Given this assumption, obviously, if the selected performance measure is the mean number of requests $L$, the overall mean number of requests in the system should be equal to the sum of the mean numbers obtained from the component models for throughput levels that correspond to the measured system throughput levels $\{x_S^{exp}\}_i$.

If the selected performance measure is the mean response time $R$, Little's formula [7] can be used to obtain the overall mean response time for arbitrary values of system throughput $x_S^{mod}$:

$$r_S^{mod} = \frac{\sum_{k=1}^{K}(r_k^{mod} x_k^{mod})}{x_S^{mod}} \tag{1}$$

Clearly, to attempt to validate this approach, we select system throughput values $x_S^{mod} = x_S^{exp}$ and we compare the mean response time values obtained from Formula (1) with those known from measurements $r_S^{exp}$ for the same values of system throughput.

If the loss probability is the selected performance measure, the overall loss probability can be expressed as:

$$p_S^{mod} = \frac{1}{1 + \frac{x_S^{mod}}{\sum_{k=1}^{K} x_k^{mod} \cdot p_k^{mod}/(1-p_k^{mod})}} \tag{2}$$

Formula (2) allows us to assess the overall loss probability in terms of component-level loss probabilities and request throughputs.

As mentioned earlier, for our approach to work, we need to know the relationship between the overall system throughput and the throughputs of individual components. If this relationship is not obvious or known from the structure of the system, we may be able to determine it using measurements. To this end, we need at least one set of reasonably synchronized measurements of $x_S^{exp}$ and $x_k^{exp}$, $k = 1, \ldots, K$. If only one set of synchronized measurements is available, the best one can do is to assume that the observed ratios $x_k^{exp}/x_S^{exp}$ carry over to other workload levels, i.e., remain constant as the overall system throughput varies. With multiple measurement points, one can check if this is indeed the case or possibly try to infer a more involved relationship between throughputs.

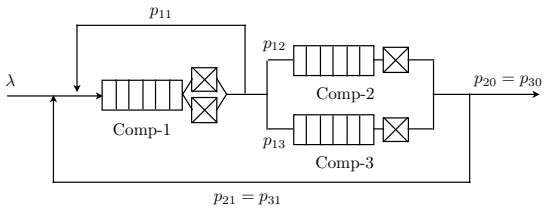


Figure 2: A centralized system architecture.

Note that the throughput ratios discussed above are analogous to the so-called visit ratios in analytical modeling methods such as Mean Value Analysis [19, 20] or BCMP theorem [5]. With few exceptions, in general solution methods, these ratios are taken to be constant. Additionally, the general analytical solutions require specific restrictions on the types of service discipline and service time distributions in order to be applicable. On the other hand, our approach requires no specific assumptions on service disciplines and distributions or arrivals of requests (even if we may assume that the throughput ratios remain constant).

# 4. ILLUSTRATIVE CASES

## 4.1 Successful case

In this first case we focus on the mean response time, $R_S$, for a system with a centralized architecture and a total of $K = 3$ components, one of which, is referred to as the central server(s) and the remaining $K$-1 components are referred as the peripheral servers. Our goal is to study how well $R_S$ can be determined from correctly calibrated performance models of the components.

Centralized architectures are common in telecommunication and computer network. In our example, we assume that the incoming tasks, representing the system workload, require first a burst of service from the central server(s). Upon completion of a service burst, a task may request another service burst at the central server(s) or it may need service from one of the peripheral servers. Upon completion of service at a peripheral server, a task may require another round of processing at the central server or it may leave the system.

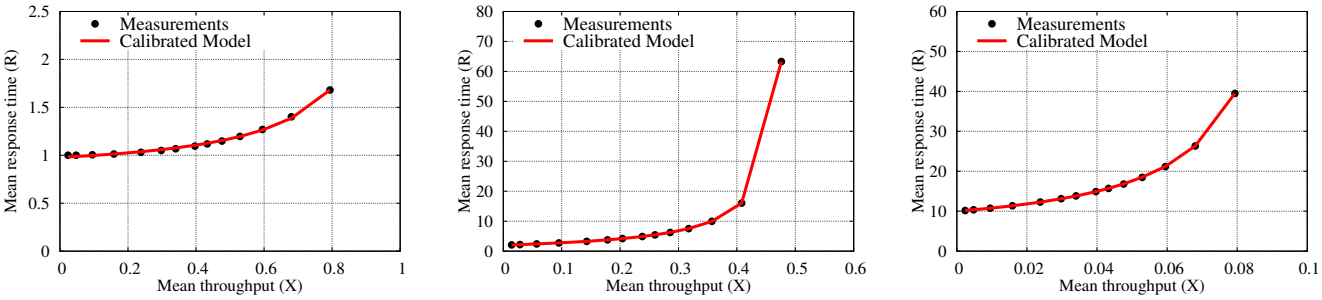
For the sake of convenience and reproducibility, the "measured" values of the overall system performance, $\{x_S^{exp}, r_S^{exp}\}_i$, $i = 1, \ldots, I$ as well as those of its components, were obtained using a discrete-event simulation. The system simulated is the machine repairman network shown in Figure 2. The task arrivals are represented through a Poisson process of rate $\lambda$. The central server(s) and the peripherals are each represented by a single queue, labeled Comp-1, Comp-2 and Comp-3, respectively. Table 1 gives the details of the simulated system parameters. Thus, in our system: (i) 30% of tasks require another burst of service at the central server(s) after completing a service burst ($p_{11} = 0.3$), (ii) two requests can be processed simultaneously at the central servers ($C_1 = 2$), (iii) the service time of Comp-2 is considerably faster than at Comp-3 ($mst_3 = 5.mst_2$), (iv) the variability of service times is much larger for Comp-2 than for Comp-3 (although the mean time is less) ($cvs_2 = 4.cvs_3$), (v) 70% of requests return to the central servers queue upon completion of service at a peripheral server ($p_{21} = p_{31} = 0.7$).

As discussed above, we assume that accurate and calibrated models have been developed for each of the 3 components (Comp-1, Comp-2 and Comp-3) of the system. In other words, the model of the $k$-th component (Comp-$k$) provides a function $r_k^{mod}(x_k^{mod})$ that returns a predicted level of mean response time for any given value of the mean throughput at the given component. Note that we used the high-level modeling approach [6] to find calibrated models for the components. In our case, these happen to be based on queueing theory but any other approach would work provided that the resulting models are accurate. Figure 3 illustrates the accuracy of the component models by showing the "measured" and the model-predicted performance for each system component. We observe that our component models match well the "measured" performance values throughout the range of workload values under consideration.

Table 1: Parameters used for Figure 2.

| | | | |
|---|---|---|---|
| Comp-1 | $C_1$ | Number of servers | 2 |
| | $mst_1$ | Mean service time | 1 |
| | $cvs_1$ | Coefficient of variation for service time | 3 |
| | $p_{11}$ | Probability of returning to Comp-1 upon request completion | 0.3 |
| | $p_{12}$ | Probability of going to Comp-2 upon request completion | 0.6 |
| | $p_{13}$ | Probability of going to Comp-3 upon request completion | 0.1 |
| Comp-2 | $C_2$ | Number of servers | 1 |
| | $mst_2$ | Mean service time | 2 |
| | $cvs_2$ | Coefficient of variation for service time | 2 |
| | $p_{20}$ | Probability of leaving the system upon request completion | 0.3 |
| | $p_{21}$ | Probability of going to Comp-1 upon request completion | 0.7 |
| Comp-3 | $C_3$ | Number of servers | 1 |
| | $mst_3$ | Mean service time | 10 |
| | $cvs_3$ | Coefficient of variation for service time | 0.5 |
| | $p_{30}$ | Probability of leaving the system upon request completion | 0.3 |
| | $p_{31}$ | Probability of going to Comp-1 upon request completion | 0.7 |

Having in hand a calibrated model for each of the system components, we can now use Formula (1) to determine the mean response time for the whole system as a function of the mean system throughput. All we need is to know are the throughput ratio of each component, namely $x_k^{exp}/x_S^{exp}$, $k = 1, \ldots, K$. These quantities may be smaller or larger than 1 as they represent the relative rate of request arrivals at component k as compared to that at the system level. In our example, it is easy to compute these ratios from the system topology (see Figure 2). The corresponding values are $\frac{1}{(1-p_{11})(1-p_{21})} = 4.762$, $\frac{p_{12}}{(1-p_{11})(1-p_{21})} = 2.857$ and $\frac{p_{13}}{(1-p_{11})(1-p_{31})} = 0.4762$ for component Comp-1, Comp-2 and Comp-3, respectively. As mentioned in Section 3, in other cases, these values can be discovered thanks to synchronized measurements of throughput. Next, in order to obtain $r_S^{mod}$ as a function of $x_S^{mod}$, we simply "convert" any given value of $x_S^{mod}$ into values of $x_k^{mod}$, $k = 1, \ldots, K$ by multiplying it by the respective throughput ratio, and then we call the component models to compute the values of $r_k^{mod}$ associated to $x_S^{mod}$, and thereby applying Formula (1). We performed this step for many levels of workload, including low and high levels. Figure 4 shows the resulting overall system performance values obtained using this approach. Clearly, and perhaps not surprisingly, we observe that the performance of the composed model $\{x_S^{mod}, r_S^{mod}\}$ match very well those measured in the simulation $\{x_S^{exp}, r_S^{exp}\}$, validating the proposed approach for this example.




(a) Comp-1 (b) Comp-2 (c) Comp-3

Figure 3: Performance measurements and modeling for each of the 3 components of the centralized system.

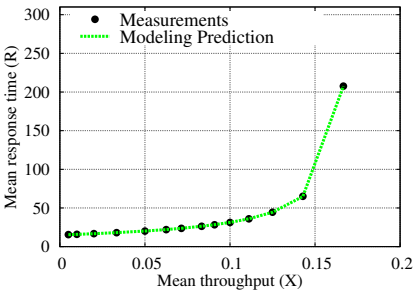



Figure 4: Predictions of system performance against measurements for the centralized system.

## 4.2 Cases of failure

As illustrated by examples in this section, considerably inaccurate performance predictions can be expected when the applicability conditions are not met.

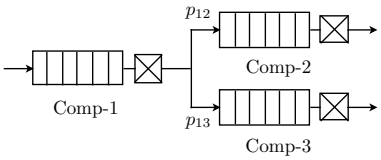



Figure 5: System with state-dependent routing.

Table 2: Parameters used for Figure 5.

| | | | |
|---|---|---|---|
| Comp-1 | $C_1$ | Number of servers | 1 |
| | $mst_1$ | Mean service time | 1 |
| | $cvs_1$ | Coefficient of variation for service time | 1 |
| | $p_{12}$ | Probability of going to Comp-2 upon request completion | 1 or .5 |
| | $p_{13}$ | Probability of going to Comp-3 upon request completion | 0 or .5 |
| Comp-2 | $C_2$ | Number of servers | 1 |
| | $mst_2$ | Mean service time | 2 |
| | $cvs_2$ | Coefficient of variation for service time | 1 |
| Comp-3 | $C_3$ | Number of servers | 1 |
| | $mst_3$ | Mean service time | 2 |
| | $cvs_3$ | Coefficient of variation for service time | 1 |

We now consider a system with 3 components in which, unlike the previous example, the routing probabilities depend on the current state of components. Such state-dependencies may occur in systems with load-balancing policies. Figure 5 illustrates the topology of our example. In this system, the incoming requests, which arrive according to a Poisson process with rate $\lambda$, go through two components. They start with Comp-1 and, if the current number of requests waiting or being served in Comp-2 is larger than 10, they are routed to Comp-3. Otherwise, they are equally likely to be dispatched to Comp-2 and Comp-3. Table 2 summarizes the system parameters used in our simulation. Note that the buffers at each component are assumed to be large enough to avoid overflows.

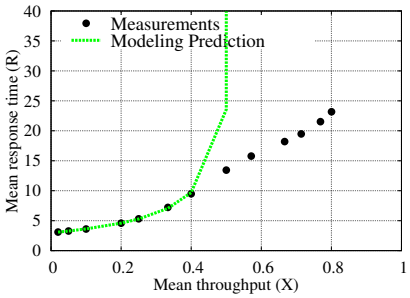



Figure 6: Predictions of system performance against measurements for the system with state-dependent routing probabilities.

Assuming we have calibrated models to capture the performance of each component, we apply the proposed approach to estimate the mean response time of the whole system as a function of the mean throughput following Formula (1). Here, the predicted values do not match well those actually observed ("measured") in the simulation. Figure 6 illustrates this discrepancy.

The reason for the discrepancy shown in Figure 6 is clear: unlike in the previous example, the throughput ratios for components Comp-2 and Comp-3 are not constant. Indeed, as the workload increases, the number of requests waiting or being served in Comp-2 increases, so that requests leaving Comp-1 become more likely to be routed to Comp-3 (up to half of them). In the proposed approach, however, the throughput ratio for each queue was derived at a lower level of workload, and it was applied for other levels of workload considered.

State-dependencies causing failure of the proposed approach can appear in seemingly different examples. Consider a system in which the arrivals and departures of requests may occur at several places in the system. Such behavior may occur in systems exhibiting internal losses of requests (e.g. due to buffer overflow, transmission errors, dynamic routing). Figure 7 shows a system where incoming requests may be routed to Comp-1 or directly to Comp-2 depending on the current number of requests waiting in Comp-1. If the number of queued requests is less than 7, then incom-

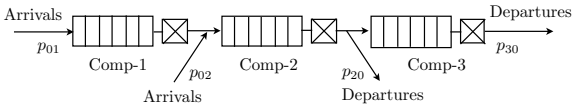



Figure 7: System with internal losses and arrivals.

Table 3: Parameters used for Figure 7.

| | | | |
|---|---|---|---|
| Comp-1 | $C_1$ | Number of servers | 1 |
| | $mst_1$ | Mean service time | 2 |
| | $cvs_1$ | Coefficient of variation for service time | 1 |
| | $p_{01}$ | Probability for new requests to enter at Comp-1 | 1 or 0 |
| Comp-2 | $C_2$ | Number of servers | 1 |
| | $mst_2$ | Mean service time | 1 |
| | $cvs_2$ | Coefficient of variation for service time | 1 |
| | $p_{02}$ | Probability for new requests to enter at Comp-2 | 0 or 1 |
| | $p_{20}$ | Probability of leaving the system upon completion of Comp-2 | 0 or 1 |
| Comp-3 | $C_3$ | Number of servers | 1 |
| | $mst_3$ | Mean service time | 2 |
| | $cvs_3$ | Coefficient of variation for service time | 1 |

ing requests go to Comp-1. Otherwise, they skip Comp-1 and go directly to Comp-2. Similarly, if the current number of requests at Comp-3 is less than 7, requests completed at Comp-2 are routed to Comp-3. Otherwise, upon completion at Comp-2, they skip Comp-3 and leave the system. Table 3 gives the details of the system parameters used in our simulation. As in our preceding example, request arrivals form a Poisson process and the buffer sizes at each component are assumed to be large enough to avoid overflows.

Again, with accurately calibrated performance models for each component, we apply Formula (1) to predict the mean response time of the whole system for different levels of mean system throughput. Here too, as illustrated in Figure 8, the proposed approach is doomed to failure since the throughput ratios of Comp-1 and Comp-3 are not constant.

As discussed above, the proposed approach would work if the rate of arrivals at Comp-2 and the rate of departures from Comp-2 were known in advance or if they were simply proportional to the overall system workload (as would be the case, for example, if requests departures represented transmission errors in a communication network).

In addition to cases in which the throughput ratios vary with workload levels and their variation is not known in advance, the proposed approach is not applicable for systems in which requests may simultaneously "occupy" two or more resources. For instance, in order to get fully processed by a "primary" component, the requests, while holding the component resource, need to receive service from a "secondary" component. The latter may be shared with other competing

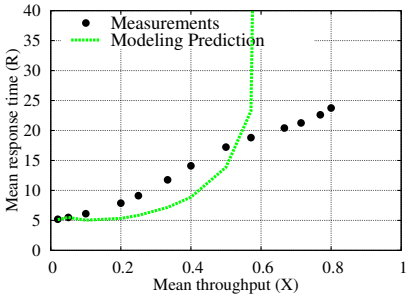



Figure 8: Predictions of system performance against measurements for the system with internal losses and arrivals.

sources of requests. Systems like databases and certain disk controllers may exhibit this behavior.

Figure 9 illustrates a simple example of a system with simultaneous resource possession. The service time in Comp-1 is represented as a two-stage process where the second stage represents the resource holding time while the request potentially waits for and accesses the shared resource at Comp-2. Here, any calibrated model of component Comp-1 is in fact a good candidate to predict the mean response time of the whole system since both coincide. Comp-1 and Comp-2 are too strongly coupled (Comp-2 can be seen as being embedded within Comp-1) and cannot be combined as described in Section 3. Similarly, systems with fork and join mechanisms would cause problems.

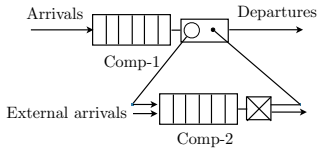



Figure 9: System with simultaneous resource possession.

# 5. DISCOVERY OF UNKNOWN SYSTEM COMPONENTS

Our next system of interest is similar to that discussed in Section 4.1 but it includes an additional component, viz. Comp-4. We assume that available measurements points pertain to the overall system performance as well as to components Comp-1, Comp-2 and Comp-3 with the exception of Comp-4. This latter may have been considered unimportant for the overall system performance or simply overlooked by or unknown to the performance analyst. Comp-4 may represent for example access to internal tables or buffers deemed so fast that it is unlikely to be a factor in the overall system performance. Figure 10 illustrates the corresponding system with a gray box around Comp-4. We re-use the same system parameters for components 1 through 3 as in Table 1 and indicate the parameters for Comp-4 in Table 4. The parameters for Comp-4 were chosen so that it can become a bottleneck because of the frequency with which requests visit it while each visit is very short.

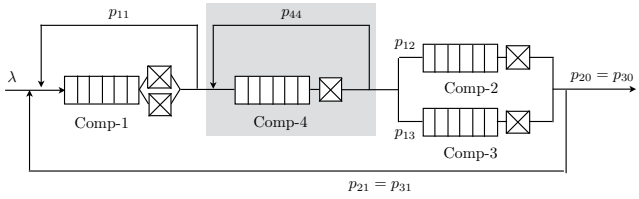
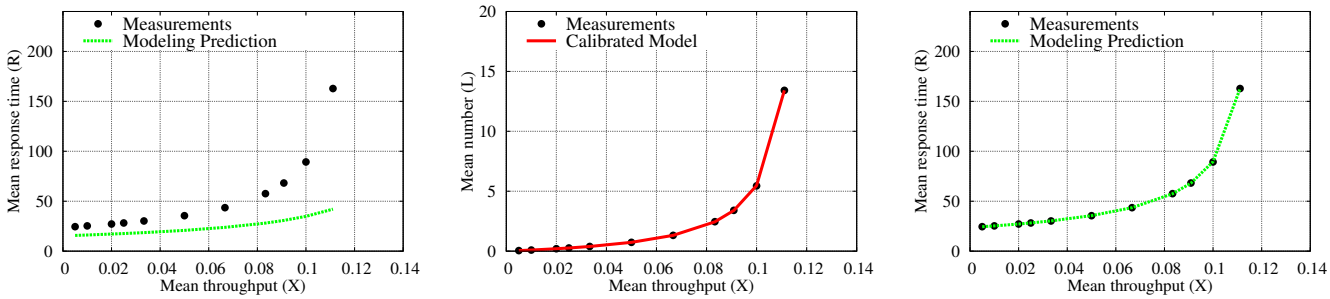



Figure 10: System with an "unknown" component.

Table 4: Parameters used for Figure 10.

| | | | |
|---|---|---|---|
| Comp-4 | $C_4$ | Number of servers | 1 |
| | $mst_4$ | Mean service time | 0.1 |
| | $cvs_4$ | Coefficient of variation for service time | 0 |
| | $p_{44}$ | Probability of returning to Comp-4 upon request completion | 0.96 |

Having developed a calibrated performance model for each of the components Comp-1, Comp-2 and Comp-3, we applied the proposed method following Formula (1) to derive the performance of the whole system. Figure 11a displays the corresponding results. Clearly, the proposed approach is missing something. Now, assuming this missing something




(a) Initial performance prediction (b) Modeling the residual performance (c) Refined performance prediction

Figure 11: Predictions of system performance against measurements for the centralized system with an unknown component.

is an additional component that was not measured (and perhaps even not instrumented) and therefore not modeled, we consider the difference in the performance metric between the system measurement points and the performance curve given by Formula (1). Given this residual pattern of performance, we apply a black-box modeling approach [6] to find a model that reproduces adequately this behavior. It turns out that a simple *M/M/1* queue can fit the data (see Figure 11b).

If the pattern was more chaotic and could not be matched by a reasonable model, it might imply that the residual performance difference is just measurement "noise", or that something else is amiss and that the system does not comply with the set of assumptions required for the proposed approach to work. A possible approach to try to determine whether the residual performance difference is due to "measurement noise" would be to apply simple statistical tests [1].

Now, having in hand a calibrated model for each of the four system components, we can re-apply the proposed approach to forecast the system performance. The corresponding results are shown in Figure 11c and clearly, including a new component was the right choice in our case.

# 6. CONCLUSIONS

In this paper we consider the problem of combining calibrated performance models of individual system components into an accurate system-level performance model. We concentrate on open workload systems and we show that under certain conditions the straightforward application of Little's law allows one to effect this integration. We give examples to illustrate the successful application of the proposed approach, as well as examples that show the extent of inaccuracies if the applicability conditions are not met.

Additionally, we show that by analyzing the discrepancies between the model predictions and the measurements it may be possible to determine if an important system component has not been correctly represented. This can be of help in the design of calibrated system performance models.

As future work, the authors plan to further investigate the important issue of distinguishing "measurement noise" from errors due to missing components. Another area of investigation pertains to the extensions of the proposed framework to closed systems.

# 7. REFERENCES


[1] Allen, A. O. (1990). *Probability, Statistics, and Queueing Theory: With Computer Science Applications*, Academic Press.

[2] Awad, M., & Menascé, D. A. (2014). On the Predictive Properties of Performance Models Derived through Input-Output Relationships. *Computer Performance Engineering*.

[3] Awad, M., & Menascé, D. A. (2014). Dynamic Derivation of Analytical Performance Models in Autonomic Computing Environments. In Proc. of *CMG*.

[4] Balsamo, S., & Iazeolla, G. (1982). An extension of Norton's theorem for queueing networks. *IEEE Transactions on Software Engineering*.

[5] Baskett, F., Chandy, K. M., Muntz, R. R., & Palacios, F. G. (1975). Open, closed, and mixed networks of queues with different classes of customers. *Journal of the ACM*.

[6] Begin, T., Brandwajn, A., Baynat, B., Wolfinger, B. E., & Fdida, S. (2010). High-level approach to modeling of observed system behavior. *Performance Evaluation*.

[7] Bolch, G., Greiner, S., Meer, H., & Trivedi, K. (2005). *Queueing Networks and Markov Chains*.

[8] Brandwajn, A. (1974). A model of a time sharing virtual memory system solved using equivalence and decomposition methods. *Acta Informatica*.

[9] Brandwajn, A. (1985). Equivalence and decomposition in queueing systems - A unified approach. *Performance Evaluation*.

[10] Chandy, K. M., Herzog, U., & Woo, L. (1975). Parametric analysis of queuing networks. *IBM Journal of Research and Development*.

[11] Courtois, P. J. (2014). *Decomposability: queueing and computer system applications*, Academic Press.

[12] Ganger, G. R., & Patt, Y. N. (1993). The process-flow model: examining I/O performance from the system's point of view. In Proc. of *ACM SIGMETRICS*.

[13] Harbaoui, A., Salmi, N., Dillenseger, B., & Vincent, J. M. (2010). Introducing queuing network-based performance awareness in autonomic systems. In Proc. of *IEEE ICAS*.

[14] Herzog, U. (1974). Some remarks concerning the extended analytic models for system evaluation. *IBM RR 4975*.

[15] Kühn, P. (1976). Analysis of Complex Queueing Networks by Decomposition. In Proc. of *IEEE ITC*.

[16] Kraft, S., Casale, G., Krishnamurthy, D., Greer, D., & Kilpatrick, P. (2011). IO performance prediction in consolidated virtualized environments. In Proc. of *ACM SIGSOFT*.

[17] Kraft, S., Casale, G., Krishnamurthy, D., Greer, D., & Kilpatrick, P. (2013). Performance models of storage contention in cloud environments. *Software and Systems Modeling*.

[18] Marie, R. (1979). An approximate analytical method for general queueing networks. *IEEE Transactions on Software Engineering*.

[19] Reiser, M. (1979). Mean Value Analysis fo Queueing Networks - A New Look at an Old Problem. In Proc. of *IFIP PERFORMANCE*.

[20] Reiser, M., & Lavenberg, S. S. (1980). Mean-value analysis of closed multichain queuing networks. *Journal of the ACM*.