A Stable Mean Value Analysis Algorithm for Closed Systems with Load-dependent Queues

Lei Zhang
Department of Computing and Software
McMaster University
1280 Main Street West
Hamilton, ON L8S4K1, Canada
zhangl64@mcmaster.ca

Douglas G. Down
Department of Computing and Software
McMaster University
1280 Main Street West
Hamilton, ON L8S4K1, Canada
downd@mcmaster.ca

ABSTRACT

The load-dependent Mean Value Analysis (MVA) algorithm suffers from numerical instability. Different techniques have been adopted to avoid this issue, however, they either have large complexities or restrictive assumptions. In this paper, we introduce a numerically Stable MVA (SMVA) algorithm for product-form networks that allows for load-dependent queues. The SMVA algorithm offers an efficient and accurate approximate solution. We validate SMVA by comparing it to other MVA algorithms in a concrete example, and analyze its errors. We also extend it to a multi-class model.

CCS Concepts

•**Mathematics of computing** → **Queueing theory;**

Keywords

Performance evaluation; mean value analysis; numerical stability

1. INTRODUCTION

The Mean Value Analysis (MVA) algorithm for closed queueing networks with product-form steady-state distribution was introduced by Lavenberg and Reiser [7]. It relies on product-form assumptions, which can be violated by common features introduced in modern computer systems, e.g., simultaneous resource possession, locking behaviours, priority scheduling, high service demand variability, and process synchronization (see Chapter 15 in [5]). An approximate solution is to reduce a non-product-form network by using Flow-Equivalent Servers (FES) [3]. An FES is load dependent, whose service rate with n jobs present is equal to the observed throughput of the original network with n jobs (as shown in Figure 1). Then, we can solve the performance model by the load-dependent MVA algorithm [6].
Unfortunately, the load-dependent MVA algorithm suffers from numerical instability issues [6, 7]. The underlying reason is that the computation of state probabilities can yield negative results when the utilization is close to one. Consequently, negative values of mean performance measures (i.e., response times and throughputs) can be produced. Static and dynamic scaling techniques are complex, and Casale and Serazzi [2] show that they do not work in general, as the mean queue length computations are not affected. To the best of our knowledge, the literature is lacking efficient solutions for the numerical instability of MVA.

VALUETOOLS 2016, October 25-28, Taormina, Italy

DOI 10.4108/eai.25-10-2016.2266403

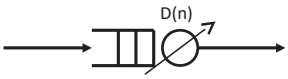


Figure 1: A generic load-dependent queue

To effectively address numerical instability, we propose the Stable MVA (SMVA) algorithm. The main contributions of this paper include: (1) The SMVA algorithm, which is an efficient approximate solution for closed networks with load-dependent queues. (2) An extended multi-class model used to determine class-level mean performance metrics.

2. RELATED WORK

To address numerical issues, Casale [1] introduced the Conditional MVA (CMVA) algorithm, which avoids the computations of the state probabilities, and as a consequence, overcomes the limitation. Although the CMVA algorithm is an exact solution, its time and space complexities grow as $O(MN^2)$, where M is the number of queues, and N is the number of jobs. The time and space complexities for the original MVA algorithm only grow as $O(MN)$.

In the literature, Seidmann's approximation [8] is also widely used to address MVA's numerical issues. However, it is only applicable for multi-server queues in which all servers are load independent. This is a special case of generic load-dependent queues. The basic idea is to replace a queue with m servers by two tandem servers. The first one is a single server queue with service demand D/m, where D is one server's service demand. The second one is a pure delay server with service demand $D \cdot (m-1)/m$. In practice, Seidmann's approximation can yield noticeable errors under intermediate loads, but it has the same time and space complexities as the original MVA algorithm.

3. STABLE MEAN VALUE ANALYSIS

We study a generic load-dependent queue (as shown in

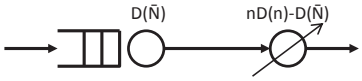


Figure 2: Approximating queues in SMVA

Figure 1) with service demand $D(n)$, where n is the number of jobs in the queue. Here, we assume that the service demand of the load-dependent queue becomes a constant beyond some $\bar{N}$, i.e., there exists a finite $\bar{N}$ such that $D(n) = D(\bar{N})$ for all $n \geq \bar{N}$. This assumption is reasonable for many systems, in particular when $D(n)$ becomes sufficiently close to $D(\bar{N})$.

The SMVA algorithm can be seen as a generalized Seidmann's approximation, replacing the load-dependent queue with two tandem servers (as shown in Figure 2). The first is a load-independent queue with service demand $D^q(n) = D(\bar{N})$. The second is a load-dependent delay centre with service demand

$$D^d(n) = \begin{cases} nD(n) - D(\bar{N}), & \text{if } n < \bar{N} \\ \bar{N} \cdot D(\bar{N}) - D(\bar{N}), & \text{if } n \geq \bar{N}. \end{cases}$$

To make sure the service demands of the delay centre are positive, we assume that $nD(n) \geq D(\bar{N})$, for $n < \bar{N}$. In multi-core systems, it is a common assumption that $D(n)$ decreases as n increases. In other words, we have $D(n) > D(\bar{N})$ when $n < \bar{N}$. Naturally, the assumption of $nD(n) \geq D(\bar{N})$ holds. Although the delay centre is load dependent, there is no need to calculate its state probabilities because it does not have a queue. As a result, the SMVA algorithm is numerically stable.

Algorithm 1 The SMVA algorithm

Input:
$Z, M, N, D_m, \bar{N}_m$
Output:
Q_m, X, R
Condition:
$\forall n, nD_m(n) \geq D_m(\bar{N}_m)$
Initialization:
$Q_m(0) = 0$, for all $m = 1, \ldots, M$
Iteration:
for $m = 1 \rightarrow M$ **do**
  for $n = 1 \rightarrow N$ **do**
    $D^q_m(n) = D_m(\bar{N}_m)$
    $D^d_m(n) = \begin{cases} nD_m(n) - D_m(\bar{N}_m), & \text{if } n < \bar{N}_m \\ \bar{N}_m \cdot D_m(\bar{N}_m) - D_m(\bar{N}_m), & \text{if } n \geq \bar{N}_m \end{cases}$
  end for
end for
for $n = 1 \rightarrow N$ **do**
  for $m = 1 \rightarrow M$ **do**
    $R^q_m(n) = D^q_m(n)[1 + Q_m(n-1)]$
    $R^d_m(n) = D^d_m(n)$
  end for
  $X(n) = n/\{Z + \sum_{m=1}^{M}[R^q_m(n) + R^d_m(n)]\}$
  for $m = 1 \rightarrow M$ **do**
    $Q_m(n) = X(n) \cdot R^q_m(n)$
  end for
end for
$R = \sum_{m=1}^{M}[R^q_m(N) + R^d_m(N)]$

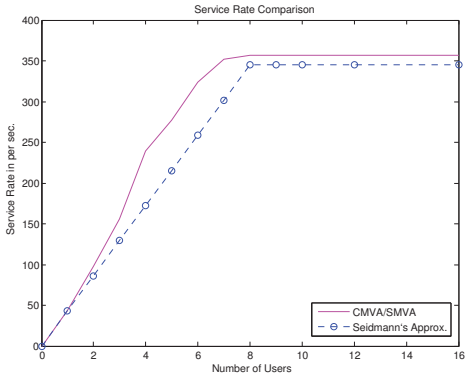


Figure 3: Service rate curves

Under light load, the two tandem servers behave as a server which has service demand $D(n)$. If n jobs are being served and no jobs are waiting in the queue, the time spent by a job in the approximating node is $D(\bar{N}) + nD(n) - D(\bar{N}) = nD(n)$. If there are jobs waiting in the first queue, the time spent by a job in the approximating node is dominated by the time spent at the first queue. The node behaves as a server which has service demand $D(\bar{N})$. As a result, this approximation should perform well for both light and heavy loads. Note that SMVA is identical to Seidmann's approximation when $nD(n) = D(1)$, for $n \leq \bar{N}$.

Algorithm 1 provides the SMVA algorithm. Z is the mean think time. D_m, D^q_m, and D^d_m are the service demands at the mth queue, the queueing resource, and the delay centre, respectively. R, R^q_m, and R^d_m are the response times of the system, the mth queue, and the delay centre, respectively. Finally, X is the system throughput, and Q_m is the mean queue length at the mth queue. The time and space complexities of SMVA are both $O(MN)$. Here, we assume that all queues are load dependent. If the mth queue is load independent, we can simply set $D^q_m = D_m$ and $D^d_m = 0$, and Algorithm 1 is still applicable.

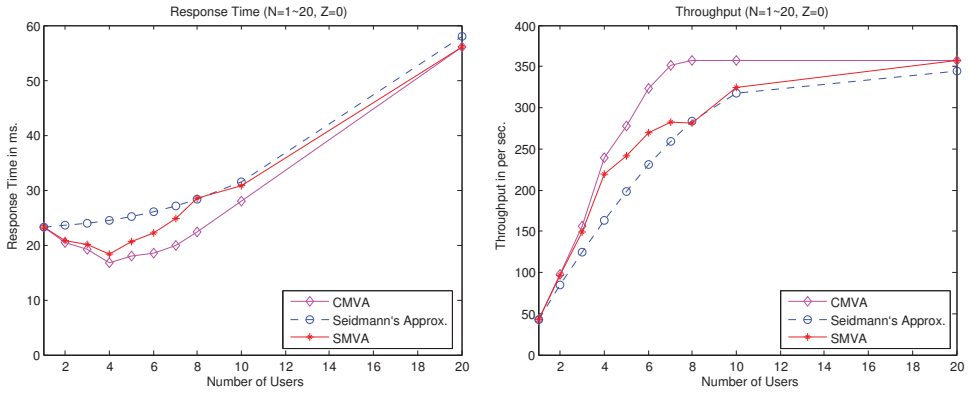
4. EXPERIMENTAL RESULTS

In this section, we compare the results of the CMVA algorithm, Seidmann's approximation, and the SMVA algorithm. We set up a testbed as shown in in Table 1, and employ TPC-W [4] to generate the workload. We aggregate and model the system by an FES. We then obtain service rates (the inverse of service demands) of the FES to parameterize MVA algorithms (as shown in Figure 3). As can be seen in the figure, the service rate curve adopted by Seidmann's approximation can only address the ideal case of a load-dependent server, where $D(n) = D(1)/n$ for $n \leq 8$.

To test the accuracy of SMVA under different loads, we vary the number of users and the mean think times in the system. Both the mean response time and the throughput are compared for the three candidate MVA algorithms. Three sets of results are presented. The results of the first set are presented in Figure 4, where N ranges from 1 to 20, and Z=0. The results of the second set are presented in Figure 5, where N ranges from 1 to 300, and Z=0.7 seconds. The results of the third set are presented in Figure 6, where N ranges from 1 to 1300, and Z=3.5 seconds.

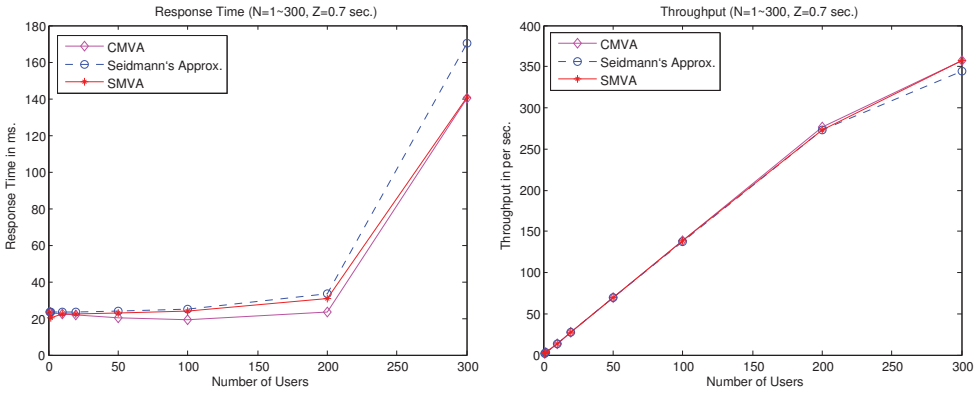
CPU	Memory	Disk	OS	Web Server	Database
Intel i7-2600 quad-core	8 GB	1 TB (7200 RPM)	Ubuntu 12.04.3 LTS	JBoss 3.2.7	MySQL 5.1.70

Table 1: Testbed settings



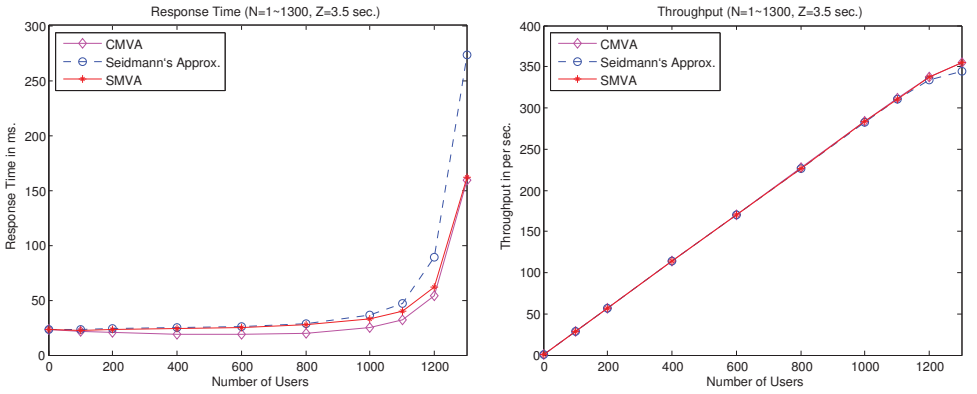
(a) Response time (b) Throughput

Figure 4: Comparison with Z=0



(a) Response time (b) Throughput

Figure 5: Comparison with Z=0.7 seconds



(a) Response time (b) Throughput

Figure 6: Comparison with Z=3.5 seconds

Using CMVA as the benchmark (as it is an exact solution), SMVA works better than Seidmann's approximation in all three cases. However, we also observe some errors for both of the approximate MVA algorithms from those figures, except for Figures 5b and 6b. The reason is that for those figures, the throughput is given by $X(n) = n/[Z + R(n)]$. As Z increases, the error in $R(n)$ has a smaller effect on the accuracy of $X(n)$.

We would like to quantify the errors from SMVA and Seidmann's approximation. In Figure 4, the largest error for the mean response time of SMVA is 27.16%, and the largest error for the throughput of SMVA is −21.36% (negative means underestimate) when $N = 8$. In contrast, the errors for both the mean response time and the throughput of Seidmann's approximation go to the largest when $N = 4$ (46.78% and −31.87%, respectively). We have similar observations from Figures 5 and 6. In Figure 5, the largest error for the mean response time of SMVA is 33.15% when $N = 200$. However, the errors for the throughput are quite small, the largest error is only −1.05% ($N = 200$). In Figure 6, the largest error for the mean response time of SMVA is 35.73% when $N = 800$. The errors for the throughput are negligible. In Figure 6, Seidmann's approximation has its worst case when $N = 1300$, the error for the mean response time is 71.91%.

5. DISCUSSION

Compared to the CMVA algorithm and Seidmann's approximation, the SMVA algorithm has two advantages. First, the time and space complexities of SMVA are a significant improvement over CMVA. Second, the SMVA algorithm is better able to handle cases when the service demands of a load-dependent node do not have a linear relationship.

We have two additional observations about SMVA. Firstly, the SMVA algorithm works as well as the CMVA algorithm for both light and heavy loads. However, the errors of SMVA increase when the system is under intermediate loads (but still performs better than Seidmann's approximation). Secondly, when the mean think time increases with respect to the service demands, the SMVA algorithm might produce less accurate estimates of the mean response times under intermediate load. In contrast, the estimated throughput becomes more accurate.

We conjecture that there are two reasons leading to the more significant error of SMVA under intermediate loads. Firstly, it assumes that all the jobs are being processed at the server when n jobs are in the system. In the load-dependent MVA algorithm, j jobs could be processed with probability $P(j|n)$, where j varies from 1 to n. Secondly, the delay centre can "delay" a job too long, in particular when the time that a job spends at the first queue does not dominate the total time. A good starting point for future research would be to better estimate the service demands of the delay centre.

6. MULTI-CLASS EXTENSION

For completeness, we extend the SMVA algorithm to the case of multi-class closed networks. Consider that there are C classes of transactions, where the job population vector is given by $\vec{N} = (n_1, n_2, \ldots, n_C)$, and the service demand of class c ($1 \le c \le C$) at the mth queueing resource is given by $D^q_{m,c}(n) = D_{m,c}(\bar{N}_m), \forall n$. The service demand at the delay centre becomes

$$D^d_{m,c}(n) = \begin{cases} nD_{m,c}(n) - D_{m,c}(\bar{N}_m), & n < \bar{N}_m \\ (\bar{N}_m - 1)D_{m,c}(\bar{N}_m), & n \ge \bar{N}_m. \end{cases}$$

Then, the multi-class SMVA iterates over all of the classes to compute the mean response times:

$$R^q_{m,c}(\vec{N}) = D^q_{m,c}(n_c)[1 + Q_m(\vec{N} - 1_c)],$$

and $R^d_{m,c}(\vec{N}) = D^d_{m,c}(n_c)$. Here $\vec{N} - 1_c$ is the job population vector with one class c job less in the system. The system throughput of class c is calculated by

$$X_c(\vec{N}) = n_c/\{Z + \sum_{m=1}^{M}[R^q_{m,c}(\vec{N}) + R^d_{m,c}(\vec{N})]\},\ 0 \le n_c \le N_c.$$

The mean queue length at the mth queue is

$$Q_m(\vec{N}) = \sum_{c=1}^{C} X_c(\vec{N}) \cdot R^q_{m,c}(\vec{N}).$$

7. CONCLUSIONS

In this paper, we presented the SMVA algorithm, an efficient approximation to address the numerical instability of the load-dependent MVA algorithm for closed queueing networks. The SMVA algorithm generalizes the applicability of Seidmann's approximation, and achieves better complexities compared to the CMVA algorithm. The SMVA algorithm can be seen as a first step to effectively address the numerical instability of MVA algorithms.

Acknowledgments. The work reported in this paper was supported by the Ontario Research Fund and the Natural Sciences and Engineering Research Council of Canada.

8. REFERENCES

[1] G. Casale. A note on stable flow-equivalent aggregation in closed networks. *Queueing Systems*, 60(3-4):193–202, 2008.

[2] G. Casale and G. Serazzi. Stabilization techniques for load-dependent queueing networks algorithms. In J. A. Barria, editor, *Communication Networks and Computer Systems: A Tribute to Professor Erol Gelenbe*, chapter 8, pages 127–141. Imperial College Press, 2006.

[3] K. M. Chandy, U. Herzog, and L. Woo. Parametric analysis of queuing networks. *IBM Journal of Research and Development*, 19(1):36–42, 1975.

[4] T. Horvath. TPC-W J2EE implementation, 2008. http://www.cs.virginia.edu/∼th8k/downloads, last accessed 24 Feburary 2016.

[5] D. A. Menascé, V. A. Almeida, L. W. Dowdy, and L. Dowdy. *Performance by Design: Computer Capacity Planning by Example*. Prentice Hall PTR, 2004.

[6] M. Reiser. Mean-value analysis and convolution method for queue-dependent servers in closed queueing networks. *Performance Evaluation*, 1(1):7–18, 1981.

[7] M. Reiser and S. S. Lavenberg. Mean-value analysis of closed multichain queuing networks. *Journal of the ACM (JACM)*, 27(2):313–322, 1980.

[8] A. Seidmann, J. Paul, and S. Shalev-Oren. Computerized closed queueing network models of flexible manufacturing systems: A comparative evaluation. *Large Scale Systems*, 12:91–107, 1987.