Towards Response Time Guarantees for e-Service Middleware

Achim Kraiss* Frank Schoen* Gerhard Weikum+ Uwe Deppisch*

*Dresdner Bank AG, Enabling Technology Group, IT Research & Innovations
{achim.kraiss, frank.schoen, uwe.deppisch}@dresdner-bank.com

+University of the Saarland, Computer Science Department, weikum@cs.uni-sb.de

1 Introduction

1.1 The Need for Response Time Guarantees

Quality of service (QoS) is a hot topic for Internet-based e-commerce and other e-services. However, it is much more talked about than it is really provided by deployed systems. Application service providers such as Akamai or performance rating companies such as Keynote contribute to a paradigm shift from 'best effort' to performance guarantees, but these trends still disregard a number of critical issues. First, they mostly focus on mean response time averaged over long time periods like weeks or months, but in high-end applications such as banking or online stock brokerage the variance and the tail of the response time distribution are equally important and it is often the peak load during the busiest hour that matters most. Second, advanced e-services aim to provide differentiated QoS for various classes of customers and requests. This requires a judicious prioritization with regard to resource allocation, scheduling, etc. QoS for multiclass workloads has been adressed in the areas of OLTP (e.g., [FGND93]) and video/audio servers (e.g., [NMW97]), but these approaches focused on specific aspects and do not easily carry over to the modern blend of broader e-Services.

As an application scenario where differentiated QoS is mission-critical consider large-scale online banking including advanced services such as customer portfolio management. In such a setting, we wish to differentiate first-class clients such as agents of the bank's call center (i.e., internal customers that issue requests on behalf of external customers who are waiting on the phone) or premium customers, second-class clients that comprise all regular customers of the bank, and third-class clients whose requests are mere inquiries and who are not necessarily customers of the bank. Orthogonally to this classification, various types of requests such as simple stock purchase or selling orders versus more complex portfolio assessments give rise to a second dimension of the multiclass workload. For certain kinds of services we may even wish to further differentiate these classes by taking into account the client's device and connection: for example, using a cell phone versus using a modem connection versus a high-speed connection.

Thus, the bank has to consider a nontrivial number of different request classes, and each class should be served with adequate response time guarantees. For example, 95 percent of the simple 'bread-and-butter' requests from first-class customers should exhibit a response time of at most 1 second, whereas second-class customers should be satisfied with a response time of 2 seconds in 95 percent of all cases. For more complex requests this kind of bounding the tail of the response time distribution would be less stringent, say 5-second response time with probability 90 percent for portfolio analyses, and this would be acceptable for users because of the requests' higher complexity.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

It has been widely observed that middle-tier application servers and backend database servers at the e-service provider's site(s) are the most performance-critical components of Internet business portals (as opposed to the network itself). Application servers are typically based on some form of message-oriented middleware (MOM), such as Microsoft MESSAGE QUEUEING [Mic00] or IBM MQSERIES [IBM01], which is also the backbone for the communication between the middle tier and the backend(s). Tuning the MOM for appropriate handling of different message types in multiclass workloads is key to ensuring response time guarantees, and it is the particular focus of this paper.

1.2 A HEART for MOM

In this paper we outline the architecture, underlying mathematical models, and implementation of a tuning tool for MOM that we have been developing in the IT research department of Dresdner Bank AG and coined HEART (Help for Ensuring Acceptable Response Time) [KSWD00]. The tool is specifically geared for IBM MQSERIES, which is the MOM product that our bank is using for strategic applications. The key contribution of our work is to automatically derive, from predefined response time goals for different customer and request classes, appropriate priority settings for the various message types that are enqueued at and processed by the MOM server. Goals can be specified in terms of response time moments (i.e., mean, variance) and/or percentiles (i.e., tail bounds) individually for each class. HEART tests for satisfiability of all goals for the given server resources and overall load, and if all goals are feasible then the tool derives a suitable mapping of messages types to ten different priorities supported by MQSERIES. The necessary computations are based on two tiers of performance models: (1) an analytic model using queueing theory for testing the goals that refer to moments and for heuristically pruning the large search space of possible prioritizations, and (2) a more accurate simulation model for identifying a priority assignment that satisfies all goals including the ones that refer to percentiles. Preliminary measurements with MQSERIES confirm the practical viability of HEART.

2 System Architecture

Figure 1 illustrates the three-tier system architecture that underlies our work and an example application scenario. The figure shows a backend server managing a database with customer portfolio data and keeping a MOM input queue for receiving service requests. Applications can send request messages to the input queue of the backend system and obtain reply messages using MOM functionality. Each request in the input queue is processed by the server, and the result is sent back to the requesting application. In our example setting, the server receives requests on behalf of 4 different customer classes with specific response time requirements: call center agents for serving the demands of customers waiting on the phone and customers using a "real-time" brokerage service need lower response times than customers who just "play" with their virtual portfolios and administrative staff gathering statistics for billing purposes. Workflow servers and web application servers are typical examples for middletier application servers hosting business logic and interoperating with backend systems via MOM.

For reliability, a MOM system like MQSERIES internally stores a request into a locally managed, persistent queue before sending it to the input queue of the backend system under the transactional protection of a two-phase commit protocol. For the reliable transmission of the reply message to the requesting application, the MOM again stores the reply in a local queue in order to commit the distributed transaction with the backend server as quickly as possible, and later delivers the message to the application. For two-tier applications with less stringent reliability concerns, MQSERIES also provides direct delivery of messages between the client and the queues of the backend system, as illustrated in Figure 1 for customer class 4.

For controlling the response times for the various request classes at the backend system, different priorities can be assigned to the request messages that are put into the backend input queue. Priority assignment and priority-based scheduling of messages thus provide the basis for ensuring specific response time goals.

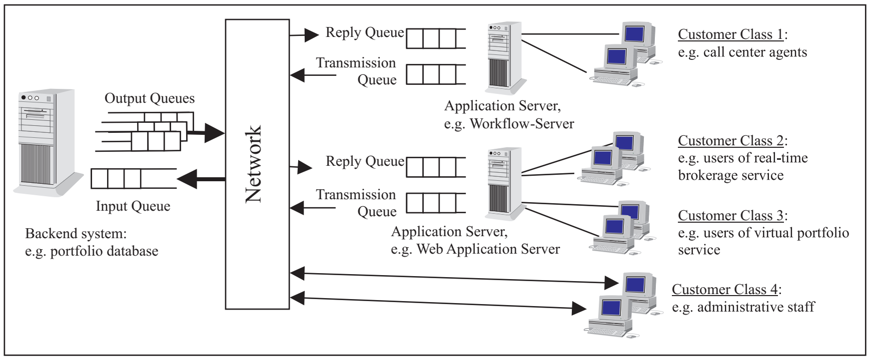

Figure 1: Architecture and example application scenario

3 Overview of the Tuning Tool

The HEART tuning tool takes as input a (statistical) description of the various workload classes and their response time goals. Conceptually the tool enumerates all possible priority assignments for the given classes, and assesses the performance of each such configuration to identify the ones that satisfy all performance goals. The number of possible configurations is quite large. For example, with 3 customer categories, 2 types of requests, and 2 types of connections or user devices that need to be distinguished, we arrive at 12 workload classes that need to be mapped to up to 10 priorities supported by MQSERIES. Then the total number of theoretically possible configurations are more than $24 \cdot 10^9$ possibilities for the given scenario. The tool includes considerations to prune this large search space. In particular, once a configuration has been considered that does not satisfy the specified goals, all configurations that merely refine the unacceptable one (e.g., by combining different priority classes into the same class without reordering the relative priorities of different workload classes) are no longer considered at all.

The performance assessment component of HEART proceeds in two stages. It first derives relevant performance metrics analytically, based on a stochastic model that can be efficiently evaluated. This step works with sufficient accuracy for the first two moments (i.e., mean and variance) of the response time distribution as we will sketch in Section 4. If the goals of the application refer only to these moments, then the analytical assessment is good enough to either accept the configuration or discard it.

When goals refer to percentiles, a more sophisticated analytic model is evaluated that is no longer accurate enough to properly discriminate good vs. bad configurations. However, this model still serves as a conservative bound for the actual value of the response time percentile, and it can identify acceptable configurations if less restrictive percentiles are required. The analytically derived conservative bound for the response time percentile at least can identify the most promising candidate configurations (i.e., the ones with the best analytically estimated percentiles). For all potentially good configurations, a more detailed and accurate simulation model is run as the second stage of HEART's assessment component.

HEART can operate in several modes depending on the administrator's preferences. It can either identify all acceptable configurations and search among those for the best configuration in terms of some subsidiary objective function (e.g., to minimize the overall mean response time across all classes or the mean of the highest-priority class) in addition to satisfying all specified goals, or it can simply stop as soon as it has found an acceptable configuration. At this point, all these computations are offline, and produce mere suggestions to the administrator, but we plan on an online version that would automatically adjust priorities for MQSERIES without interrupting the server. The latter is obviously important for dealing with evolving workloads where, for example, the fractions of the different workload classes (or even their, possibly time-of-day-dependent, response time goals) vary over time. HEART also interacts with the MQSERIES server to extract statistics about arrival rates and service time moments of the various workload classes.

4 Performance Modeling

4.1 Response Time Moments

The stochastic model for the assessment of the response time moments is an M/G/1 model with multiple priority classes and non-preemptive service [Tak91]. This model assumes that requests arrive according to a Poisson process, a common assumption in performance modeling that is justified when a large number of clients issue requests independently of each other, and models a server with multiple request queues, one for each priority class. Whenever the server finishes the execution of a request, it inspects its queues in descending order of priority and selects the first non-empty queue. It then picks the first request in that queue and starts executing it. So the service discipline is FCFS within each priority class, but higher-priority requests are preferred even if they have arrived later than some lower-priority requests. Once a request has started its execution, it will not be preempted, regardless of higher-priority arrivals.

This kind of model has been intensively studied in the literature, with explicit formulas for the first two moments, $\mathrm{E}[\mathrm{R}_p]$ and $\mathrm{E}[\mathrm{R}_p^2]$, of the response time distribution of each priority class p, with 1 being the highest priority and P being the lowest priority [Tak91, Nel95]:

$$\mathrm{E}[\mathrm{R}_p] \;=\; \mathrm{E}[\mathrm{W}_p] + \mathrm{E}[\mathrm{S}_p] \;, \qquad \mathrm{E}[\mathrm{R}_p^2] \;=\; \mathrm{E}[\mathrm{W}_p^2] + \mathrm{E}[\mathrm{S}_p^2] + 2\mathrm{E}[\mathrm{W}_p]\mathrm{E}[\mathrm{S}_p] \tag{1}$$

where $\mathrm{E}[\mathrm{W}_p]$ and $\mathrm{E}[\mathrm{W}_p^2]$ are the first two moments of the waiting time distribution [Tak91]:

$$\mathrm{E}[\mathrm{W}_p] \quad = \quad \frac{\sum_{k=1}^{P} \lambda_k \mathrm{E}[\mathrm{S}_k^2]}{2(1-\rho_{p-1}^{+})(1-\rho_p^{+})} \tag{2}$$

$$\mathrm{E}[\mathrm{W}_p^2] \;=\; \frac{\sum_{k=1}^{P} \lambda_k \mathrm{E}[\mathrm{S}_k^3]}{3(1-\rho_{p-1}^{+})^2(1-\rho_p^{+})} + \frac{\left(\sum_{k=1}^{P} \lambda_k \mathrm{E}[\mathrm{S}_k^2]\right)\left(\sum_{k=1}^{P} \lambda_k \mathrm{E}[\mathrm{S}_k^2]\right)}{2(1-\rho_{p-1}^{+})^2(1-\rho_p^{+})^2} + \frac{\left(\sum_{k=1}^{p-1} \lambda_k \mathrm{E}[\mathrm{S}_k^2]\right)\left(\sum_{k=1}^{P} \lambda_k \mathrm{E}[\mathrm{S}_k^2]\right)}{2(1-\rho_{p-1}^{+})^3(1-\rho_p^{+})} \tag{3}$$

In these formulas, λ_k denotes the arrival rate of class k requests (i.e., the number of requests arriving per time unit), $\mathrm{E}[\mathrm{S}_k]$, $\mathrm{E}[\mathrm{S}_k^2]$ and $\mathrm{E}[\mathrm{S}_k^3]$ are the first three moments of the class-specific service time (i.e., the time for which the server will be used by the request execution once the execution has started), and ρ_p^{+} is the utilization of the server for requests of priorities 1 through p (i.e., the fraction of time for which the server is busy with requests from these priority classes):

$$\rho_p^{+} \;=\; \sum_{k=1}^{p} \lambda_k \mathrm{E}[\mathrm{S}_k] \;. \tag{4}$$

All input parameters of these formulas, the λ_k values and the service time moments for the different classes, are obtained from run-time statistics of the MQSERIES server.

4.2 Response Time Percentiles

For determining percentiles (i.e., certain points in the tail of the distribution), the stochastic model has to analyze the entire response time distribution. Because queueing adds up service times of multiple requests during which a request needs to wait, this involves sums of random variables whose convolution (i.e., the distribution of the sum) is mathematically easier to deal with by applying the Laplace-Stieltjes transform (LST) to the individual variables' distributions. This way we can derive the LST of the response time distribution for each priority class (see [Tak91] for details).

The LST of the response time cannot be easily inverted to reconstruct the distribution in a closed form (a typical situation in stochastic performance modeling). Therefore, we compute the Chernoff bound for the percentiles of interest [Nel95], a reasonably tight bound (much better than the more widely known Chebyshev bound). However, the Chernoff bounds are conservative, and sometimes significantly overestimate the actual response time percentile. This is why the HEART tool can use this part of the stochastic model only (1) for identifying configurations which satisfy weak performance goals, and (2) as a heuristics for identifying the most promising candidate configurations (see Section 3). All computations for evaluating the stochastic model are implemented with Maple [Mapl01].

4.3 Simulation Model

The simulation model uses the process-oriented discrete-event simulation package CSIM [Mesq01], and can be viewed as a very high-level, abstract version of the actual server algorithm. In contrast to the analytical models, it can, however, capture more detail of the real queue manager, scheduler, and execution engine; for example, it can distinguish persistent vs. non-persistent queues and other details that are specific to MQSERIES. Furthermore, the simulation tracks the entire response time distribution of each class through extensive statistics collection (i.e., fine-grained histograms). Simulation runs are continued (or iterated) until the desired confidence level (e.g., 0.95) and confidence interval for the metric of interest (e.g., real mean plus/minus 10 percent) are reached. Note that the time-consuming simulation process is only run for those configurations which have been identified as candidate configurations in the first stage using the analytical model.

5 Experimental Results

To study the practical viability of HEART we have carried out some stress test experiments with synthetic workload profiles. We consider three categories of customers, first class (1), second class (2), and third class (3), that are combined with two types of requests, short ones (A) and long ones (B). For all six resulting classes together we chose an overall arrival rate of $\lambda = 12$ requests per second, so that the server utilization of our target hardware equaled 90 percent (which seemed adequate for a stress test). The relative frequencies of the 3 customer classes are Zipf distributed with class 3 being the most frequent one. We further assume that class A requests dominate the workload with a relative frequency of 9/10, and that this holds across all three customer classes. The service times for request classes are assumed to be exponentially distributed with the mean for class B being six times more resource-intensive than the mean for class A. The first three columns of the following table summarize the synthetic workload profile.

Class	Fraction of overall load	Mean service time	Goal for mean response time	Goal for 95^{th} percentile of response time distribution
1A	2/11 * 9/10	0.05 seconds	0.2 seconds	0.9 seconds
1B	2/11 * 1/10	0.30 seconds	0.5 seconds	1.4 seconds
2A	3/11 * 9/10	0.05 seconds	0.4 seconds	1.1 seconds
2B	3/11 * 1/10	0.30 seconds	1.2 seconds	3.0 seconds
3A	6/11 * 9/10	0.05 seconds	1.6 seconds	no goal
3B	6/11 * 1/10	0.30 seconds	10.0 seconds	no goal

For our stress test experiment we chose fairly tight performance goals, in terms of class specific mean response times and 95th percentiles, as shown in the fourth and fifth columns of the above table. The effect that we wanted to study is that for such tight goals only very few of the possible priority assignments can indeed satisfy all goals. For the given six workload classes, there are 4683 different possibilities of prioritization, but only 3 of them are feasible with regard to all goals.

For this setting, HEART first determined the possible priority assignments that can satisfy the goals with regard to mean response times of the various classes. In the experiment there were 4 candidates that satisfied this criterion. This first step is based on the analytic model and took less than 5 seconds on a commodity PC. Then HEART examined the configurations that passed this first filter with regard to the required 95th percentiles. This second step is based on the simulation model and took approximately 2 minutes for each inspected configuration. The simulation of a configuration was automatically stopped when a 95 percent confidence level was reached for the mean response time plus/minus 0.1 seconds. One of the 4 candidates did not reach the goal for the 95th percentile. Out of the 3 qualifying configurations, HEART chose the priority assignment that minimizes the overall mean response time over all six classes as a subsidiary objective function. The resulting solution was the priority list [{1A}, {1B}, {2A}, {2B, 3A}, {3B}] giving highest priority to class 1A, second highest to 1B, and so on. Note that classes 2B and 3A were combined into a single priority class.

Finally we ran this workload on a real MQSERIES server, and measured the resulting performance. These measurements were run for 1,000,000 message executions, to achieve a confidence level of 95 percent for the mean response times plus/minus 0.1 seconds. The results are shown in the third and fifth columns of the following table (together with the original goals in the second and fourth columns).

Class	Goal for mean response time	Measured mean response time	Goal for 95th percentile of response time distribution	Measured 95th percentile of response time distribution
1A	0.2 seconds	0.2 seconds	0.9 seconds	0.7 seconds
1B	0.5 seconds	0.49 seconds	1.4 seconds	1.31 seconds
2A	0.4 seconds	0.31 seconds	1.1 seconds	1.02 seconds
2B	1.2 seconds	0.88 seconds	3.0 seconds	2.6 seconds
3A	1.6 seconds	0.88 seconds	no goal	2.6 seconds
3B	10.0 seconds	9.17 seconds	no goal	over 30 seconds

The table demonstrates that all goals were indeed satisfied in the real system environment. The bottom line is that, by following the predictions and recommendations of HEART, all goals could be satisfied in the real system. The stress test nature of the experiment is underlined by the fact that there was not much slack left between some of the goals and the actual performance figures. We are currently conducting a comprehensive suite of experiments, and if these confirm the viability of HEART we plan to deploy our tuning tool for production purposes.

References

[FGND93] D. F. Ferguson, L. Georgiadis, C. Nikolaou, K. Davies: *Goal Oriented, Adaptive Transaction Routing for High Performance Transaction Processing Systems*, 2nd Int. Conf. on Parallel and Distributed Information Systems, San Diego, USA, 1993

[IBM01] IBM: *IBM MQSeries Family*, http://www.ibm.com/software/ts/mqseries, 2001

[KSWD00] A. Kraiss, F. Schoen, G. Weikum, U. Deppisch: *Middleware-based Response Time Guarantess for e-Services* (in German), 9th German Database Conference BTW '01, Oldenburg, Germany, 2001

[Mesq01] Mesquite Software: *CSIM18 Simulation Engine*, http://www.mesquite.com, Austin, USA, 2001

[Mic00] Microsoft: *Microsoft Message Queueing*, http://www.microsoft.com/msmq, 2001

[Nel95] R. Nelson: *Probability, Stochastic Processes and Queueing Theory*, Springer-Verlag, Heidelberg, Germany, 1995

[NMW97] G. Nerjes, P. Muth, G. Weikum: *Stochastic Service Guarantees for Continuous Data on Multi-Zone Disks*, Int. Symp. on Principles of Database Systems, Tucson, USA, 1997

[Mapl01] Waterloo Maple Inc.: *MAPLE 6*, http://www.maplesoft.com, 2001

[Tak91] Takagi, H.: *Queueing Analysis: A Foundation of Performance Evaluation — Volume 1: Vacation and Priority Systems (Part 1)*, Elsevier Science Publishers, Amsterdam, The Netherlands, 1991