

Performance Analysis of High Volume/Low Latency Marketplace Systems

Bo Wahlberg and Mathias Barkhagen^{*}
Automatic Control Lab & ACCESS
School of Electrical Engineering, KTH
SE-100 44 Stockholm
Sweden
bo.wahlberg@ee.kth.se

Peter Snellman and Daniel Rufelt
Cinnober Financial Technology AB
Kungsgatan 36
SE-111 35 Stockholm
Sweden
peter.snellman@cinnober.com

ABSTRACT

The aim of this presentation is to discuss some challenging issues in performance analysis and modeling of JAVA based high throughput and low latency computing systems for electronic trading. It is based on test results from systems developed and operated by the Swedish company Cinnober Financial Technologies. A very important performance measure for such systems is the response time or what is also called the door-to-door latency time, i.e. the time it takes from the moment a transaction enters the marketplace to the moment the corresponding response/confirmation is returned from the system. A main source of variations of the response time is the load of the system. The studied systems are JAVA/JVM based, which means that certain processes regularly have to do garbage collection. This will also contribute to the variations of the response time. The latency depends very much on the load characteristics, and the transaction flows are often bursty. Typically, the average latency is given for a certain load, but the median is often a better measure since the underlying distribution, due to garbage collection and queuing effects, typically has a rather fat tail. Hence, it is important to quantify the cumulative distribution function in order to find more detailed performance measures such as quantiles. We have studied how to use a two mode Bernoulli queue model to model garbage collection service rate effects. It is possible to analyze the corresponding model under classical queuing theory assumptions, but due to the uncertain input distribution and other real world constraints the results are for our case not reliable. Instead, simulation based queuing models, derived and calibrated by test data, are used to predict performance under more realistic conditions.

All JAVA/JVM vendors are currently developing and releasing new improved real-time garbage collection technologies. Performance testing and analysis on a system level

^{*}This work was partially supported by the Swedish Research Council and the Linnaeus Center ACCESS at KTH

for e.g. capacity and performance planning is most important to evaluate these new concepts and also new hardware solutions.

1. INTRODUCTION

An increasing part of all the trading conducted at financial marketplaces is executed by computers using different automated strategies. Algorithmic trading has led to increased volumes being traded and thus better liquidity, as well as new opportunities to profit from small price changes and other information by quickly and automatically respond to the information by having a trading strategy that makes the right decision extremely fast. This has created a high demand for marketplaces with low latency, and now the industry is talking about latency figures of milliseconds rather than seconds and the trend to lower figures is obvious. The current state-of-the-art marketplace systems, e.g. the Multilateral Trading Facilities (MTF) Turquoise, BATS and Chi-X in Europe and Alpha Trading Systems in Canada, have response times in the order of milliseconds, while handling very large volumes.

In this work, which is partly based on [12], the probability distribution of the latency in a computerized trading system, i.e. a marketplace, is analyzed by making a simplified stochastic model of the system, with the goal to find bottlenecks, looking at scalability and understanding how the latency distribution is affected when changing different parameters in the system. Special attention is given to the upper tail of distribution, because the tail behavior is crucial when specifying and agreeing on the latency demands of a marketplace. We will describe how to get insights in the system performance by means of certain load tests. Using test data the system is modelled and analyzed using queue theory and stochastic discrete event simulation. It is very difficult to use classical queuing theory to analyze performance since computer systems often are closed queueing circuits with a lot of constraints. Here stochastic simulation in combination with real tests is probably the best approach when dealing with rather complex marketplace systems. This is valid in particular when it comes to systems with service stations having vacations and dependent arrival times which essentially differ from normal Poisson behavior. The conclusions mainly based on the fact that the framework of steady state queue theory is not general enough when it comes to modelling

queue networks, and relies too heavily on the assumptions of exponential distributions.

In financial markets, reacting fast on new information, like world news, price changes, interest rate cuts etc., has always been crucial for maximizing profits. However, in recent years the meaning of fast has changed quite dramatically, which is basically due to the increasing amount of volumes traded by computers. Humans are more and more becoming people who instruct the computers how to trade by using different pricing models, trading ideas and how to react on trading patterns, rather than people who actually trade in real time by making real time decisions. The trading is to a much higher extent being conducted by computer using so called algorithmic trading; common synonyms used in this context is black-box, program, automated or computerized trading. To illustrate how important this type of trading is becoming, let us have a look at some numbers: Today over a third of all stocks in EU and US are traded by algorithmic traders. This has led to higher demands of technology to cope with the increased volumes and to higher demands of low latency. Apparently fully automated electronic marketplaces and algorithmic trading are two key words when predicting the future of financial trading. It is not hard to understand why algorithmic trading leads to higher volumes; placing large amounts of orders each second and running and maintaining many different trading strategies can nowadays be done on a single computer, instead of by a whole team of people trading in real time and more or less manually following some strategy. Before continuing it is meaningful to define what actually is meant by a marketplace. A marketplace is a physical or electronic "place" where traders can enter and update buy and sell orders. The goal of the marketplace is to match those orders, by finding buyers and sellers who agree on price and volume so that they can be executed. Assets traded on these marketplaces can be stocks, commodities, bonds or derived financial instruments like options, futures, swaps etc. A more or less public marketplace for trading stocks is usually called a stock exchange.

Most orders never get matched, and it is usual that a trader sometimes wants to cancel orders that have not yet been matched. When entering an order into a marketplace it is thus natural for the trader to receive a response from the marketplace stating that the order was entered but not matched, entered and directly matched or that the order was not entered, perhaps because of a trading halt. During real time trading, this response is very important since it may affect future orders directly. For instance, a response stating that a sell order was not matched may result in an immediate urge to lower the price or even change the current trade strategy. In this report all requests sent to the system will be denoted transactions. An order is a buy or a sell inquiry on one or several assets traded in the system, and a transaction can either be an entry of a new order, an update of a previously entered order or the removal of an order. A part from being able to handle large volume amounts, marketplaces today have extremely high latency demands and this is above all a requirement created by the increased amount of algorithmic trading.

2. LATENCY

There are many different definitions of the latency of a marketplace, see [13] for an overview and a study to establish a performance benchmark for latency for electronic trading. The so-called response time or door-to-door latency will here be considered the most relevant and best measure, which is motivated below. One can define latency to be the total time a trader has to wait after sending a transaction to the market, before receiving a response from the marketplace verifying it. From a traders point of view, this is the only thing that really matters. However, note that there is a physical distance between the traders computer and the marketplace, and thus a network transfer time which actually is relevant in a millisecond world. Typically this transfer time is dependent on what kind of connections there is between the systems, how occupied these connections are, how long the physical distance is between the servers and so on. From the marketplace point of view it is fair to define latency as the total time spent in the marketplace, from the time of entrance to the marketplace to the time of departure from the system. This time is often called the door-to-door time of a system. One can argue that this is also a good definition from the traders/users point of view, since they will then have a better understanding of what can be done to reduce the latency; either by reducing network transfer time in different ways or by using a marketplace with lower latency. A part of the processing time in a marketplace is spent in a phase called business processing, which consists of the necessary business logic, like order matching, updating prices etc., to complete a transaction. It can perhaps be considered meaningful to define the latency to be the total time spent in the business processing part, since it is a part of the total time spent in the marketplace which is difficult to improve by upgrading the computer hardware. However, marketplaces usually consists of other time consuming steps that have to complete before sending a response to the trader, which can be for instance writing the transaction to a log file for legal or redundant reasons, or replicating the order to a backup server for redundancy. Therefore, every time we consider the latency or the response time of a marketplace in this report, we refer to the total time spent in the marketplace, the door-to-door time, which is the most natural choice. Now that we have a definition we need to define what measure to use. Example of measures is the arithmetic mean, the median, the 75%-quantile etc. In this project, we are interested in specifying the full cumulative distribution function of the latency. That is, if we introduce a stochastic variable T for the latency, we are interested in the probability. From this distribution, one can then of course calculate all thinkable statistical measures like expected value, median, variance, all quantiles, skewness and so on.

3. SYSTEM DESCRIPTION

We start our description of the system by looking at a schematic overview of it, see Figure 1 below. This is a simplified representation that shows the most critical parts of the system that each transaction has to pass through before a response is sent. Basically we have a number of clients/traders connected to the gateway that sends transactions into the system. Each transaction is sent from the client to the client gateway, which then sends the transaction to the so called primary matching engine. The Primary Matching Engine (MEP) is the main part of this system and accounts for

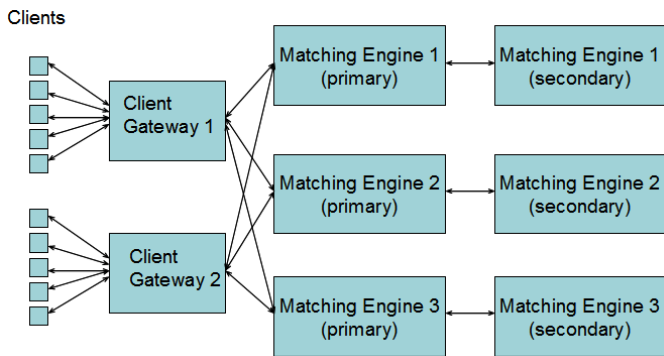


Figure 1: Trading System.

matching different orders. When a transaction enters the MEP some processing of it is done, after which a copy of the transaction is sent to the Secondary Matching Engine (MES). In parallel, the primary and the matching engine now processes the transaction in different ways. After each Matching Engine (ME) is done, additional processing is conducted in the MEP. Then the response is sent back to the gateway, which then sends a response to the client. The reason for having a secondary matching engine is of redundancy reasons; the system is used for marketplaces where the reliability is extremely important and severe downtime can be unbelievably costly in the long run. The idea is that if the primary matching engine fails, the secondary matching engine is ready to take over its role completely and almost without delay; this is a so called "fail-over" mechanism. Note that each component in the figure, in general, represents a physical server and are not just different service stations located on the same physical computer. Note also that there are delays between each of these servers, because they are connected through a network. As discussed earlier, the network delay between the client and the client gateway is discarded, because of our interest in the door-to-door time of the system itself.

In this overview, the quantity of interest is the total time from when the transaction has arrived at the client gateway, to the time when the response to the client leaves the client gateway, i.e. the door-to-door time of the client gateway. The system may sound rather simple, however to cope with high loads and to get as low latency as possible the system is partitioned to spread the load over several physical servers.

4. GARBAGE COLLECTION

The whole marketplace system is programmed in programming language Java. When it comes to latency critical applications the language has a shortcoming, namely so-called garbage collections. A Garbage Collection (GC) is basically a task that is conducted by an automatic process, a garbage collector, which reclaims memory which has once been allocated by an application but now is of no use any more. For Java Virtual Machines (JVM) different types of garbage collectors exists, with different fundamental properties. In the system that we are looking the garbage collector periodically halts all processing in the system for a certain amount of time. We can view it as if we have a process constantly filling the memory with garbage according to some

rate, and when this memory is full a GC is performed and then we have an empty memory which is then filled up with new garbage and so on. From a modelling point of view we basically need to know the following: GCs happens rather often in the system and rather periodically, each garbage collection halts all execution a relevant amount of time and the GC-frequency, i.e., how often they happen, is load dependent. Also it is relevant that the time between GCs and how long they take are not constants and follow some probability distribution that we have to determine. Here we will assume garbage collection in a certain matching engine affects all processing in that particular matching engine, but has no direct halt effect on any other computer in the system. We will make the assumption that GCs in a computer happens totally independent of GCs in other computers.

The literature on GC technologies is waste. All major JVM vendors are developing new products with improved real-time performance. To quote e.g. IBM: "The keystone of this work is our Metronome real-time garbage collection technology, which provides sub-millisecond worst-case latency and deterministic scheduling with guaranteed worst-case utilization. Metronome therefore allows the vast majority of real-time systems to be programmed in standard, garbage-collected Java." The Sun Java Real-Time System (Java RTS) is Sun's commercial implementation of the Real-Time Specification for Java. They state that objective is to make standard Java technology more deterministic and enable it to meet rigorous timing requirements for mission-critical real-time applications. A Swedish example with improved GC performance is the JVM JRockit, which now is a part of Oracle Fusion Middleware. The PhD thesis [11] contains a nice more academic overview of challenges using JAVA for real time applications. The series of International Workshop on Java Technologies for Real-Time and Embedded Systems (JTRES) publish many very relevant publications on novel and improved GC technologies. The research and literature on performance testing of computing systems using probabilistic models is also very extensive. See [3] for an interesting discussion on performance testing of E-commerce applications. An interesting approach, which we will use, to practical performance analysis can be found [8].

5. PERFORMANCE TESTS

First we will describe some real test data which illustrates the challenges in modeling of a JVM based computer systems. All these test were done using commercial software from Cinnober Financial Technologies. To illustrate the performance limits of a computer system it is common to do load saturation tests. The test here consists of multiple traders sending in a mix of sell, buy and update orders. We start with one trader who sends in as many orders as possible for 50 seconds. We then add a second trader and thus two traders send in as many orders as possible for another 50 seconds, and so on up to a total of 12 traders. The average throughput is calculated over each one second interval. The limitation is that each trader only is allowed to have one transaction in the system at the same time. The corresponding performance plot is given in Figure 2.

This simple experiment provides a lot of information. The throughput for one user is around 1300 Transactions per second (TPS), which means that the average latency is about

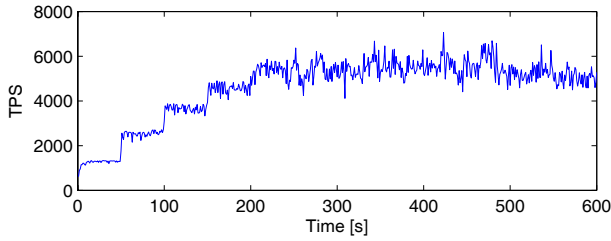


Figure 2: Transactions per second (TPS) for 1 up to 12 traders. The number of traders is increased by one every 50 second.

0.8 ms. We also see that the system scales very well due to parallelism up to around four to five traders, but then saturates at around 5500 TSP, where we have 100% utilization. This means that the highest service demand in the system is around 0.2 ms. We also see that the total throughput is reduced in case we overload the system. All this is well known facts working with partly parallel computing systems. It gets even more clear if we plot the corresponding average response time (average over one second) in Figure 3.

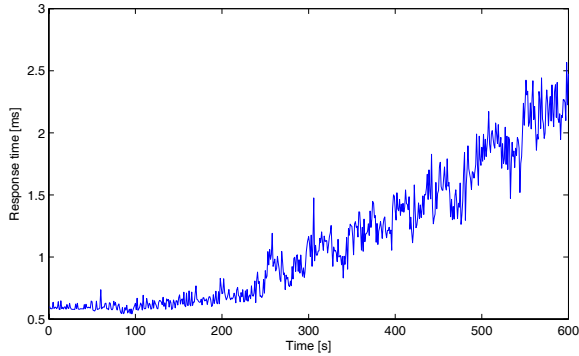


Figure 3: Average Response time [s] for 1 to 12 traders. The number of traders is increased by one every 50 second.

From an computer engineering perspective one should avoid having more than 4 to 5 orders at the same time in the systems. All this match perfectly with the conclusions and recommendations in Chapter 3 in [8]. This insight can be used to design e.g. throttling strategies for access control.

A more challenging issue is to study the distribution of the response times. Figure 4 shows a count plot of response times in certain intervals for a test with up to 40 users (we have counted the total number of response times for a test in specified rather short intervals). This information could be transferred into histograms.

The peak at around 3 ms is probably due to network delays and not GC. The main GC:s take around 50 ms, which is in accordance with 5. The corresponding statistics for a system with up to six users are given in 6 and 7. Notice the much better performance due to less load. The objective of this section was to provide some insights in the performance of the system to be modeled in the next section.

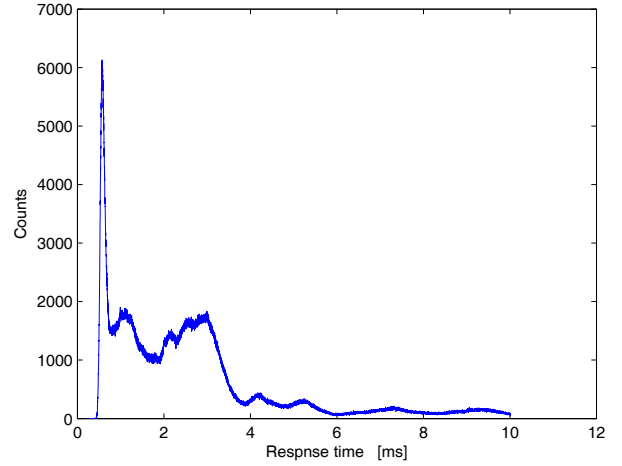


Figure 4: Counts of response times in the interval up to 10 ms for a test with up to 40 traders.

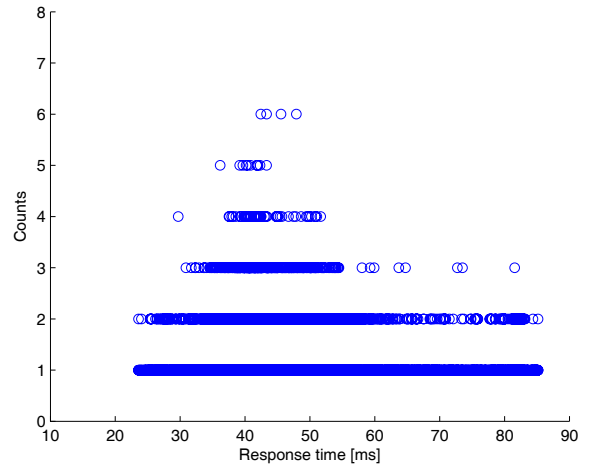


Figure 5: Counts of response times in the interval from 20 to 90 ms for a test with up to 40 traders

6. MODELLING OF QUEUES WITH LIMITED SERVICE

In the classical M/M/1 queue, [9, 6], "customers" arrive according to a Poisson process with arrival rate λ and are served by a single server with exponential service time and corresponding rate μ . Let S be the service time, which is exponential distributed with

$$E\{S\} = \frac{1}{\mu} \quad \text{and} \quad E\{S^2\} = \frac{2}{\mu^2}$$

Hence, the average service time is $1/\mu$. This model could be used when the process is running under normal conditions, but the arrival Poisson rate assumption is difficult to justify for our application. We would also like to model a computer process that take time-outs to do a garbage collection. A simple assumption is that the service process can handle k customers and then needs a vacation of time V , with exponential distributed length and corresponding rate

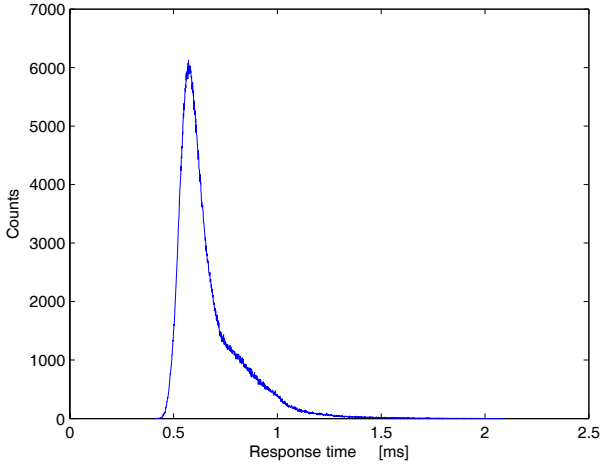


Figure 6: Counts of response times in the interval up to 2 ms for a test with up to 6 traders.

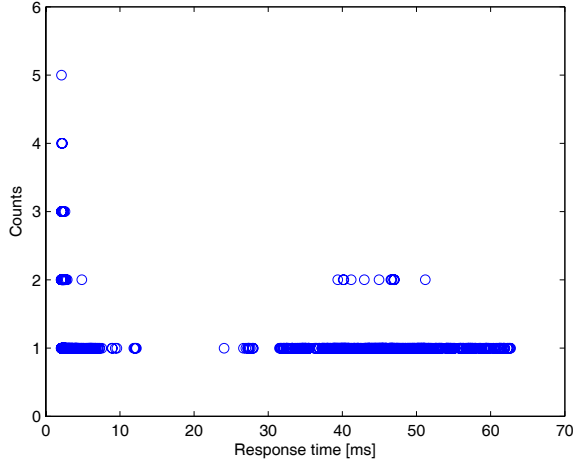


Figure 7: Counts of response times in the interval from 2 to 65 ms for a test with up to 6 traders.

μ_{gc} ,

$$E\{V\} = \frac{1}{\mu_{gc}}$$

To simplify the analysis, we will assume that the process can handle one customer under the time-out (GC), which we can thus view as an extra long service interval. It is, however, more difficult to find the steady state statistics for such a periodic service process, here denoted by S_p . However, the sample average service time equals

$$E\{S_p\} = \frac{k}{k+1} \frac{1}{\mu} + \frac{1}{k+1} \frac{1}{\mu_{gc}} = \frac{1}{\mu_p}$$

and

$$E\{S_p^2\} = \frac{2k^2}{\mu^2} + \frac{2}{\mu_{gc}^2} + \frac{2k}{\mu\mu_{gc}}.$$

We still have the problem to handle the input distribution.

A closely related model is obtained by using a Bernoulli random variable to model the timeouts. We denote the service time for this process with S_B , which then has the following distribution

$$S_B = \begin{cases} S & \text{with probability } p \\ V & \text{with probability } 1-p \end{cases}$$

where S and V are defined as above. The probability p is close related to $k/(k+1)$, since

$$E\{S_B\} = p \frac{1}{\mu} + (1-p) \frac{1}{\mu_{gc}} = \frac{1}{\mu_B}$$

while

$$E\{S_B^2\} = p \frac{2}{\mu^2} + (1-p) \frac{2}{\mu_{gc}^2}$$

It is easy to verify that

$$E\{S_B^2\} > E\{S_p^2\}$$

and thus gives more conservative results (larger variations in average). The utilization is defined as $\rho_B = \lambda/\mu_B$, where as above

$$\frac{1}{\mu_B} = p \frac{1}{\mu} + (1-p) \frac{1}{\mu_{gc}}$$

The Bernoulli based model is called a M/H₂/1 queue, see [10], where the H stands for hyper-exponential and subscript 2 for having a combination of two service processes. Its properties can now be calculated using the so-called P-K transform equation.

The assumption of a Poisson arrival process is not very realistic in our setting. Instead we would like to study G/H₂/1 queues, with more general arrival processes. It is non-trivial to find analytic expressions of the response time of such general queue systems; However, queue theory can still in some cases be used for approximation, to find mean values, to look at heavy-traffic approximations such as the Kingmans bound in rather general situations [7]. But as long as we are only interested in the whole distribution of the response time, things will in general get complicated rather quickly, especially if we look at queue networks, [1].

7. SIMULATION BASED PERFORMANCE EVALUATION

An advantage of computing systems is that it is very easy to collect data and to do experiment based modeling. In [12] a test data based model based on combining the Bernoulli GC model with empirical distribution models, was derived. The system setup was a full scale system with four client gateways, eight pairs of matching engines and a variable number of clients. Two different types of tests was conducted, one were all load was sent to a single pair of matching engines and another were all load was sent equidistributed to all pairs of matching engines. For each of these tests a number of different levels of load were pushed into the system, and there were different sets of load levels depending on which of the two types of tests that was conducted. This is of course because a system were all load is sent to a number of pair of MEs are expected to cope with more load then a single pair of MEs. The generated load into the system was generated to be as uniform as possible, meaning that we more or less had the same load rate into system independent of the time.

For each transaction the total response time T was measured, and a large data sample was collected for each load level. When it comes to GCs, there is also full statistics of every GC in every computer, i.e. we have the total halt time and the time between them for each single GC.

When we conducted the data analysis we found suitable distributions for the GC-related parameters, but unfortunately we were not able to use them when we applied queue theory on the system. Instead we were limited to exponential distributions, which of course is unsatisfying and a possible source of badly estimated latency distributions. Now with the simulation approach we can use all specified distributions, which of course gives prerequisites for good simulated results. Recall that we also had a specified number of clients in each of the test, and that we had a limitation stating that each client can only have one transaction in the system at a time. Incorporating this into a simulation model is a rather simple task; in our simulation model we will have to have a load generator generating arrivals to the client gateways, and we can simply keep track of the number of transactions in the system and not send in new ones if we reach this limit. Incorporating such a limit, called a finite population, in a general queue network is hard, and more or less only possible for a single queue process under very certain conditions. Introducing blocking and using all the right distributions for all GC-parameters we were able to build a quite good model of the system. Note however that we do not know the service time parameters; we could only give rough estimates of their typical values. The details and results for this model study are presented in [12]. To give an indication of the accuracy the following table which presents the relative error between the predicted latency using the model and validation data. Here we see much better relative error, typically within 5-

# Clients	Load	Q50	Mean	Q90	Q99	Q99.9
130	1589	2	-9	-8	-14	-1
130	3259	-2	-18	-17	-22	-34
130	4780	-13	-31	-37	-20	-79
130	6391	-14	-36	-37	-28	-78
260	8076	-5	-22	-29	-6	-78
260	10389	-7	-25	-32	-20	-80

Figure 8: The relative error between the predicted latency using the model and validation data for different loads and clients. Q50 means the 50% quantile , etc.

30 %. At higher load levels, the 90% quantile seems to be hardest to match, and at lower levels the 99% quantile is the worst ones. The only really large relative error of the 99 % quantile at the load level 191 TPS. At this load level GCs do not occur very often and it is hard to understand the cause of the high latency in the real figures. Probably the non-correspondence is related to some low-frequency effect occurring at lower load levels, which is not due to GCs or queues.

8. CONTROL PROBLEMS

A rate-control throttle is used for overload control for these systems, see e.g. [4]. The design of such input regulation techniques are challenging in this setting. They are traditionally used for overload control, but here we need to use

this for performance control. There are a lot of safety net mechanism in trading system to avoid congestions and large latency times. Typically, the total number of orders/quotes in the system is limited to avoid saturation effects and the CPU loads are often very low. This means that the dynamics of the system is fast when there is no GC. Very interesting guidelines for throughput control design and bottleneck analysis for systems of queues are given in Chapter 3 in [8]. The plots in Chapter 5 were inspired by this work. The design to handle worst case scenarios together with very good performance under normal operation conditions. In trading application it is also important that rate control mechanisms are fair in the sense that the users obtain a more uniform latency degradation. An open problem is good control design techniques that also handle higher quantile performance, and not just the average or worst case effects. The dynamics of these computing systems are quite special from a control perspective, c.f. [2]. For example, the windowing used in the leaky bucket algorithm used the rate-control throttle in itself introduces interesting dynamic effect, see [5]. To conclude: A main challenge in designing good feedback control algorithms for performance control is still reliable models and how to translate operating constraints into control specifications.

9. CONCLUSIONS

The goal of this project is to find the probability distribution of the latency in a certain marketplace system, with a structural overview that can be described using a queue network. The variations in the latency mainly came from variations in the service time distribution, which includes natural variation in the service time as well as full halts due to garbage collections. Under these circumstances there were two natural ways to analyze the system: using queue theory as described in the previous section or using stochastic simulation. The nice thing about classical queue theory are the analytic expressions that it has the potential to deliver in certain cases, but the downside is that these special cases heavily limits the types of systems that can be analyzed. Employing queue theory in a rather complicated queue network with service times having halts turned out to be problematic due to the bursty internal load that the halts naturally lead to. Also, the queue theoretic framework is heavily limited to exponential distributions when it comes to giving expressions for the full probability distributions, which turned out to be an unsuitable distribution for the system in question. Thus one can only expect queue theory to be suitable for usage in very special cases and under rather specific assumptions, and it was not successful in predicting the latency in the marketplace system. Stochastic simulation however has turned out to be a more general framework that can accommodate a much wider class of queue system: virtually any queue network can be analyzed by the use of simulations. Using this approach we have implemented our system model and it turned out to be a very good model that corresponded well to the data from benchmarks. Typically the results were within 5 - 30% from the real data, which is really good results from a rather simple model. This model can thus with confidence be used to test different principal changes, other parameter sets and so on. However caution is of course required when simulating outside the data sample, but on the other hand methods for predicting for instance GC-parameters outside the sample has been proposed. The

main downside with using stochastic simulation is the fact that we get numerical results only; we can not draw conclusions on how a change in a parameter affects a system in general. In a specific application, i.e. under a certain parameter set, the approach is however a really good one since it can be used to look at how the latency is affected when introducing different changes in the system, for instance when changing a parameter drastically, scaling the system or introducing an architectural modification. Also, in real life applications one typically has a base parameter set to work with which is then only partly changed; perhaps only one or two parameters are subject to change. Thus the model is still very usable and has the potential to reduce benchmark and development costs significantly.

10. ACKNOWLEDGMENTS

Most of the experimental work has been done within the masters thesis project of Daniel Rufelt, [12].

11. REFERENCES

- [1] J. Abate, Choudhury GL, and W. Whitt. Calculation of the GI/G/1 waiting time distribution and its cumulants from Pollaczeks formulas. Technical report, 1993.
- [2] Karl-Erik Årzén, Antonio Bicchi, Gianluca Dini, Stephen Hailes, Karl Henrik Johansson, John Lygeros, and Anthony Tzes. A component-based approach to the design of networked control systems. *European Journal of Control*, 13(2-3), June 2007.
- [3] A. Avritzer and E.J. Weyuker. The role of modeling in the performance testing of e-commerce applications. *IEEE Transactions on Software Engineering*, 30(12):1072–1083, 2004.
- [4] A.W. Berger. Performance analysis of a rate-control throttle where tokens and jobs queue. *Selected Areas in Communications, IEEE Journal on*, 9(2):165–170, Feb 1991.
- [5] A.W. Berger and W. Whitt. The Brownian approximation for rate-control throttles and the G/G/1/C queue. *Journal Discrete Event Dynamic Systems*, 2(1), 1992.
- [6] Sanjay Kumar Bose. *Introduction to Queuing Systems*. Kluwer/Plenum Publishers, 2001.
- [7] O. J. Boxma and J. W. Cohen. Heavy-traffic analysis for the GI/G/1 queue with heavy-tailed distributions. *Queueing Syst. Theory Appl.*, 33(1-3):177–204, 1999.
- [8] Neil J. Gunther. *The Practical Performance Analyst*. Iuniverse Inc, 2000.
- [9] Ng Chee Hock. *Queueing Modeling Fundamentals*. Wiley, 1996.
- [10] Leonard Kleinrock. *Queueing systems, Volyme 1: Theory*. John Wiley & Sons, 1975.
- [11] A. Nilsson. *Tailoring native compilation of Java for real-time systems*. PhD thesis, Dept. of Computer Science, Lund University, May 2006.
- [12] D. Rufelt. Fast trading: Stochastic modeling and simulation of latency in marketplace systems. Master’s thesis, KTH, 2008.
- [13] Cinnober Financial Technology. Cinnober white paper on latency. Technical report, 2008.