Self-regulating Message Throughput in Enterprise Messaging Servers – A Feedback Control Solution

Ravi Kumar G
HP, Research Scholar, JNTUH
Bangalore, India

C.Muthusamy
Yahoo
Bangalore, India

A.Vinaya Babu
JNTUH
Hyderabad, India

***Abstract*—Enterprise Messaging is a very popular message exchange concept in asynchronous distributed computing environments. The Enterprise Messaging Servers are heavily used in building business critical Enterprise applications such as Internet based Order processing systems, pricing distribution of B2B, geographically dispersed enterprise applications. It is always desirable that Messaging Servers exhibit high performance to meet the Service Level Agreements (SLAs). There are investigations in this area of managing the performance of the distributed computing systems in different ways such as the IT administrators configuring and tuning the Messaging Servers parameters, implement complex conditional programming to handle the workload dynamics. But in practice it is extremely difficult to handle such dynamics of changing workloads in order to meet the performance requirements. Additionally it is challenging to cater to the future resource requirements based on the future workloads. Though there have been attempts to self-regulate the performance of Enterprise Messaging Servers, there is a limited investigation done in exploring feedback control systems theory in managing the Messaging Servers performance. We propose an adaptive control based solution to not only manage the performance of the servers to meet SLAs but also to pro-actively self-regulate the performance such that the Messaging Servers are capable to meet the current and future workloads. We implemented and evaluated our solution and observed that the control theory based solution will improve the performance of Enterprise Messaging Servers significantly.**

Keywords-Feedback control; Message Oriented Middleware; Enterprise Messaging; Java Messaging Service; JMS Providers; Adaptive Control

I. INTRODUCTION

Enterprise Messaging also known as Message Oriented Middleware [1] is a popular asynchronous message exchange mechanism in heterogeneous distributed applications. It provides the applications in a distributed environment to send and receive messages, but still being loosely coupled. Loose coupling between enterprise class applications and legacy systems such as business workflow applications, databases, and data warehouses plays a significant role in Enterprise Application Integration (EAI) [2]. The Message based integration provides automation and simplifies the time consuming integration tasks like create, deploy and manage integration solutions. There are many such applications such as Business to Business (B2B) solutions, messaging across various entities within a business enterprise that are geographically separate where asynchronous messaging becomes a major building block [2]. Asynchronous Messaging is a backbone for many of the Event driven architectures due to the obvious advantages of asynchronous systems where the message client need not maintain the connection and session with the message receiver; no confirmation is required from the receiving application [2]. As we discussed Enterprise Messaging is an important element in the business critical environments, it is always important for the Enterprise Messaging Servers to exhibit high performance and availability. Typically there would be Service Level Agreements (SLAs) [3] defined between the business service providers and the consumers. Performance is an obvious Service Level Objective in such SLAs. Any violation of performance SLOs [4] will affect the business and reputation of the business enterprise. In this paper we want to discuss the performance regulation of Java based Enterprise Messaging Servers. There are different implementations of such Enterprise Messaging Servers. The Java based Messaging Service is called as Java Message Service [5], included in the specification for Java based Enterprise Environments called as JEE (called as J2EE previously) [6]. There are different vendors who implemented the JMS Specification and Java based Enterprise Messaging Servers are referred as JMS Providers. Hence forth in the document the Enterprise Messaging Servers are referred as JMS Providers [7].

Typically the performance of JMS Providers is measured by its message throughput, though CPU and Memory usage [8] are common metrics to measure the performance of any computing server. The message throughput will depend upon various factors such as the number of subscribers, message size, number of publishers, and number of JMS message brokers [9]. By tuning these different parameters the desired performance can be achieved on the JMS Providers. One of the mechanisms to improve the JMS provider's performance is by following some best practices such as setting non-durable messages, set the message time to live parameter appropriately, close message publishers and subscribers when they complete their jobs [10]. But these kinds of practices will not be able to address different kinds of JMS environments and applications limiting the performance improvement. The other mechanism is to provide the facilities to the administrators to configure [11] and fix the various parameters values which influence the JMS Provider performance. Due to the dynamics of messages flow and workload on the JMS Provider, it will be difficult for the administrator to tune these values accurately and

periodically. When there are sudden huge loads administrator may decide to provision additional resources which may be left unutilized [12] later when there are relatively lesser loads. This will eventually lead to either not addressing the performance needs or ineffective utilization of computing server assets. Another way to manage the performance is to include conditional programming within JMS Provider implementation to change the values of the parameters at runtime based on the workload and deviation from the expected performance. This method is though useful it is very complex because during design the workload dynamics have to be accurately estimated. During implementation the conditional programming is implemented which is very complex [13] as the conditions implemented may not be sufficient to meet the run time dynamics, any spikes in the workload. To summarize, though there are different mechanisms to adjust the JMS Providers parameters to regulate and improve the message throughput either it involves manual intervention, involve complex conditional programming implementation.

In order to handle such situations, we propose an adaptive control [14] based solution that regulates the message throughput according to the pre-defined reference using a feedback controller. Also, predict the future load on the JMS Provider, modify the control parameters accordingly. There may be a case where the future load predicted may demand additional servers; our controller will actuate a signal to provision additional resources.

In this paper we will first present a background on the choosing feedback control systems in Distributed computing systems, then a brief overview on the Java Messaging Service, followed by discussing the adaptive controller algorithm that we have implemented to regulate the performance of the JMS Provider.

II. BACKGROUND

We have discussed the importance of JMS Providers and importance of their performance in building business critical applications and services in enterprise level or at internet level. There are attempts to predict the performance of JMS Providers [15], or study and compare the performance of different vendors of JMS Providers [16]. Additionally there are some best practices [10] identified to improve the JMS Providers performance. Manual configuration is one of the most common approaches followed to tune the JMS Provider performance. There is a very limited investigation done in automatic regulation the JMS Providers performance. The message throughput of the JMS Providers depends upon the number of subscribers, publishers and number of brokers. Allowing more number of subscribers on a given JMS Provider may decline message throughput or having a less number of subscribers may leave the JMS Provider less utilized. The control system based solutions provide mechanism to automatically tune the maximum number of subscribers in an optimal operating range. In this paper we propose a control systems based solution for managing the performance by tuning the maximum number of subscribers that influence the message throughput.

Control systems theory has been in investigation to address these kinds of problems related to regulating the performance, in computing [17]. But the majority of focus is on Web Servers [18][19][20], Application Server performance regulation [21], in computer networks such as congestion control [22]. There are recent investigations to explore the applicability of control systems in other areas of Java based Cloud and Enterprise Environments [23], database driver cache hit ratio improvement [24], spring based software applications [25]. But in our study we have observed there is no investigation carried out in applying feedback control system theory in improving the performance of JMS based servers. We investigated to apply control systems theory in Enterprise Messaging server performance improvement and evaluated how the feedback controllers improve the JMS Providers performance significantly.

III. THEORITICAL AND PRACTICAL CONSIDERATIONS

The message throughput of the JMS Providers depends upon various factors such as publishers, subscribers, JMS brokers. The performance varies based on whether the messages are persisted are not. The JMS Providers exhibit higher performance when the messages are not persisted. In this paper the persistence factor is not considered and the performance is evaluated with proposed solution. The subscribers are identified as a significant independent variable influencing the message throughput. The number of publishers and the brokers will have a definite impact on the message throughput to cater huge publisher and subscriber volumes. When the subscribers and message throughput are depicted in mathematical model, the accuracy of the JMS Provider model depends upon the constant values chosen for that model. These constants can be determined by using different data set values of message throughput for varying subscribers. These values may not hold good for different workload conditions on the JMS Provider, but the best suitable constants can be chosen before running the experiments.

IV. ENTERPRISE MESSAGING PRIMITIVES

In this section we discuss a brief overview of the Enterprise messaging [26] also known as Message Oriented Middleware (MoM). The key concept behind MoM is the asynchronous messaging. It means that the sender is not required to wait for the message to be received or handled by the receiver. The Fig 1 shows high level diagram of MoM. The sender can forward the message and continue the processing. The asynchronous messages are treated as autonomic units. The message contains all the data and state needed by the business logic that processes it.

A. Enterprise Messaging Architectures

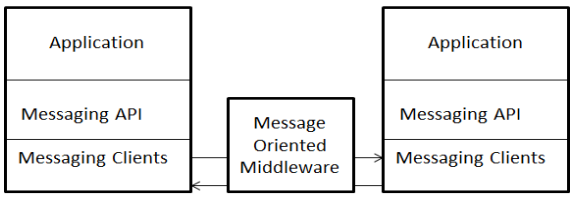

Figure 1. Message Oriented Middleware

1) Centralized Architectures :

In Centralized Architectures there will be a Message Server also called as a message router or message broker that is responsible of sending messages such that the message sender is decoupled from the message receiver. This enables the clients to be added and removed without impacting the system. In this model, the hub-and-spoke topology is used as shown in Fig 2 below:

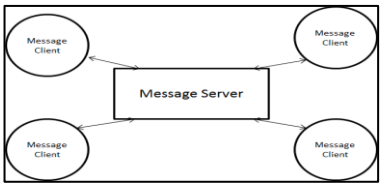

Figure 2. Centralized MoM Architecture

2) Decentralized Architectures

In Decentralized architectures, the IP multicast is used at the network level. There is no centralized server and some of the JMS functionality like persistence, transactions, security is embedded in the client application. The messaging routing is delegated to the network layer by using the IP multicast protocol as shown in Fig 3.

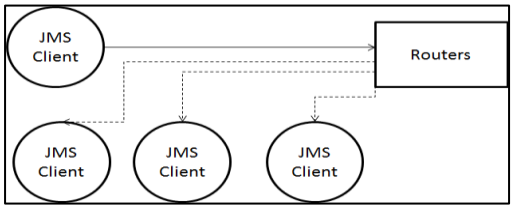

Figure 3. Decentralized MoM Architecture

B. Java Messaging Service [26]

The Java Message Service (JMS) is a specification that proposes programming API for Enterprise Messaging. JMS supports messaging as a first-class java distributed computing paradigm. There are many vendors who implemented the JMS specification and such implementations are called JMS Providers, which are nothing but Enterprise Message Servers based on Java.

1) JMS Messaging Models

The JMS provides two types of messaging models, point-to-point and publish-subscribe models. The intermediate element that enables the communication between the message producer and message consumer in JMS is called a broker. There are two types of JMS brokers as explained below.

a) Point-to-Point

The Fig 4 below shows point-to-point model in which the producer can send a message to only one consumer. In JMS Providers such JMS Brokers called Queues.

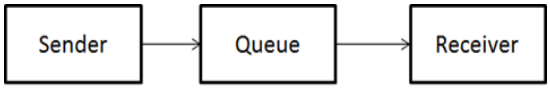

Figure 4. Point-to-Point Model

b) Publish-Subscribe

The Fig 5 below shows publish-subscribe model in which the producer can send a message to many consumers. In JMS Providers such JMS Brokers called Topics.

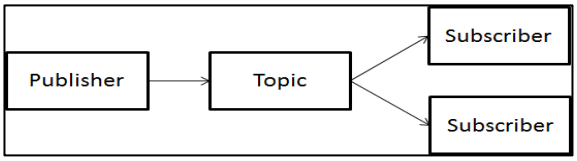

Figure 5. Publish-Subscribe Model

V. Adaptive Control

The message throughput (T) depends upon various factors such as the number of subscribers and publishers to the different brokers of the JMS Provider, The messages size, number of brokers running. In this paper we have considered how the number of subscribers of the JMS brokers affects the message throughput (T). Though there are other parameters that influence the JMS Provider message throughput, the maximum number of subscribers allowed on the server will affect significantly. The number of publishers are considered to be constant as 1 in our implementation. The Fig 6 is a Single Input Single Output (SISO) Adaptive control system [27] that shows how the message throughput is regulated using the controller and the Predictor.

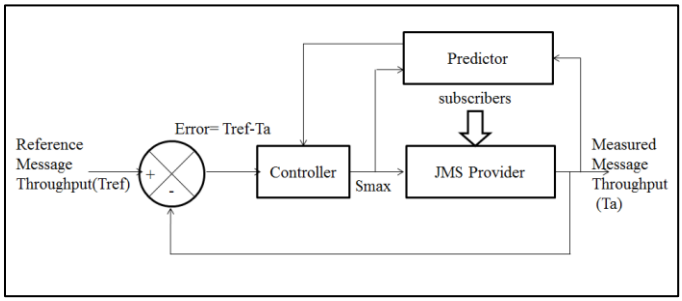

Figure 6. Adaptive Control of JMS Provider

We explain how the message throughput depends upon the number of subscribers of the JMS Provider.

The following equation (1) represents the Message Throughput and its relation with the number of subscribers.

$$T = bS \quad (1)$$

Where

T = Message Throughput of JMS Provider measured as number of messages per unit time

S = Total number of subscribers on the JMS Providers

b = proportional coefficient for the Subscribers

There is a feedback control loop that is implemented which is used to calculate the error signal of the actual Message throughput (Ta) and the Reference Message Throughput (Tref). The error signal is represented by the equation (2)

$$E = T_{ref} - T_a \quad (2)$$

The controller takes the error signal as one input and the other input signal to the controller is the predicted values of number of subscribers and message throughput. The predicted subscribers will help in estimating the possible future subscriber's volume. The predicted subscriber's value is used to predict the message throughput and the latter one is important to determine the future resource requirements. The resources may either be more JMS brokers or additional virtual machines [28] that can be scaled to cater the future load requirements on the JMS Providers. There are threshold values defined for the message throughput based on which the actuator signals are triggered either to add new virtual machines or brokers. The following sections explain the different parts of the solution in detail.

A. Modeling JMS Provider

The JMS Provider, whose message throughput needs to be controlled, has to be mathematically modeled first in order to apply the feedback control techniques. There are many ways to model the compute systems such as difference equations [29], ARMA models [30] that is based on Least Squares Parameter Estimation [31]. In our solution we have used the ARMA model to represent the JMS Provider. The Fig 7 below shows ARMA based modeling of the JMS Provider.

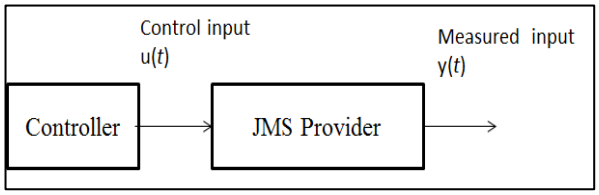

Figure 7. JMS Provider Model for Feedback control

1) Parameter Estimation

In the JMS Provider the message throughput is defined as a function

- The number of subscribers that the JMS Provider is supporting.
- Though the number of publishers and number of message brokers also influence the JMS Provider performance we considered the number of subscribers as the factor in our paper.

According to ARMA, in a Single Input Single Output model, for a given sample data set, the next sample of the output can be predicted using the current and previous inputs. The same is explained in the equation (3) below:

$$y(t+1) = ay(t) + bu(t) \quad (3)$$

Where

$y(t)$ = The current output
$u(t)$ = The current input
a = The model parameter to be estimated
b = the model parameter to be estimated
$y(t+1)$ = the output in the next step

The same ARMA model if is applied to model the JMS Provider, it is represented by the equation (4):

$$T(t+1) = aT(t) + bS_{max}(t) \quad (4)$$

Where

$T(t)$ = The current output of message Throughput
$S_{max}(t)$ = the current input of maximum number of Subscribers
a = the model parameter to be estimated
b = the model parameter to be estimated
$T(t+1)$ = the output in the next step

The ARMA model is used to estimate the model parameters 'a' and 'b'. The details of the experiments and the estimated values are explained in the section VI. "Implementation and Analysis". Based on our experiments the parameter 'a' is determined as 0.91 and 'b' is 0.12.

2) Input Operating Range

It is important to determine the operating range of the maximum number of Subscribers (Smax). The training data set is used again to determine the range of Smax that provides the desired Tref.

In order to achieve the desired value of the Tref, the maximum number of subscribers will have to be adjusted. This value of Smax again will change during runtime due to the stochastic nature of the load and the controller is useful to automatically adjust the Smax to meet the Tref. The details are explained in the Section VI.A "Implementation and Analysis – Modeling JMS Provider"

B. Adaptive Controller

The adaptive controller is designed and implemented to self-regulate the message throughput of the JMS Provider for a pre-defined threshold of message throughput.

We implemented the adaptive control algorithm such that any changes in the JMS Provider load can be well managed such that the desired message throughput (Tref) is achieved at any given point of time. The adaptive control has two different parts.

- Feedback Controller: The feedback controller is reactive in nature and tunes the controller gain based on the current measured message throughput, but cannot handle the future load on the JMS Provider. This runs a "sub-control loop" and at the end of each such loop the controller parameter is tuned such that the message throughput is in an allowed range of Tref
- Predictor: In order to handle the future dynamics of the loads on the JMS Provider, a predictor is used that predicts the Smax and Tref. Based on these predicted values the P-Controller Gain is tuned if predicted desired message throughput is lesser than a pre-defined error. We defined a "parent control loop" that runs periodically. In each parent-control loop the Smax and Tref are predicted. After each parent-control loop, the predicted value of message throughput is compared with the Tref. If the predicted value is less than Tref within a pre-defined deviation then controller tunes the Smax allowed, by adjusting the Controller gain (Kp) such that subsequent loads on the JMS Provider meet the Tref. We used the basic P-Controller [32] to tune

the value of Smax. If this deviation is greater than a pre-defined threshold then it demands additional resources, then the actuator triggers request to create a new Virtual Machine.

Now we explain the two different parts of the proposed Adaptive control solution, Feedback Controller and the Predictor.

1) Feedback P- Controller

The JMS Provider during its operation will have varying workloads that may affect its performance. In order to maintain and regulate the performance in terms of Tref, the maximum number of subscribers that can be allowed on the JMS Provider will have to be tuned. We implemented a P-Controller [32] to adjust the Smax during runtime such that JMS Provider exhibits desired performance. The lower number of subscribers will have the possibility of high Tref, but having a too low value of Smax keeps the JMS Provider under-utilized. In the section VI.A "Implementation and Analysis – Modeling JMS Provider" we have discussed optimal operating region of Smax based on our experiments. In order to keep the desired Tref, the P-Controller will tune the Smax in the operating region. But there may be cases where the actual measure Throughput (Ta) is much lower than Tref. In such scenarios, the control law will trigger a request to provision additional compute resource such as more compute power (e.g., Virtual Machine). The Fig 8 below shows the P-Controller to tune the Smax.

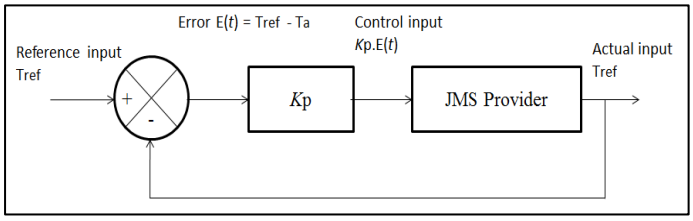

Figure 8. Feedback Control of JMS Provider

The output of the controller is represented by the equation (5) below

$$u(t) = K_p.E(t) \qquad (5)$$

Where

$u(t)$ = The controller output

K_p = Proportional Gain

The P-Controller Gain is represented in the equation (6) in z-Transform

$$K_p = E(z)/U(z) \qquad (6)$$

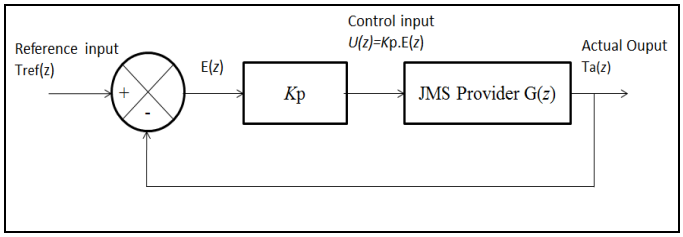

Figure 9. Feedback Control of JMS Provider in z-Transform

The Fig 9 shows the z-Transform [33] of the JMS Provider adaptive loop control.

The equation (5) represents the controller output is. The controller output, which is the new Smax becomes the control input to the JMS Provider. The reactive feedback control runs for every "sub-control loop".

The JMS Provider is represented by G (z) is a first order system as shown in the equation (7) below

$$G(z) = b/(z - a) \qquad (7)$$

2) Predictor

The Predictor is a component in our proposed solution that predicts the maximum number of subscribers for the future periods. The Time-Series Triple Exponential Smoothing [34] is used to predict the Smax that represents the possible future maximum number of subscribers that could be allowed on the JMS Provider based on the past history.

The smoothing technique has the ability to forecast up to 'm' periods ahead. It means that the maximum number of subscribers that can be supported after 'm' periods from the current time can be predicted and hence the corresponding Tref.

The Reference Message Throughput is predicted using the predicted Smax and the previous value of the reference message throughput. The equation (8) below shows how the Tref is predicted

$$T_{ref}(t + 1) = aT_{ref}(t) + bS_{max}(t + 1) \qquad (8)$$

In the Fig 6, we can notice that the Predictor accepts the measured throughput (Ta), current Smax and outputs the predicted Tref. (TrefPred) thus helps in tuning the Kp for the future period.

C. Controller Algorithm

In this section we explain the controller algorithm

The following are the pre-conditions and Initialization operations before the controller is executed

- The JMS Provider model parameters '*a*' and '*b*' are estimated
- The "parent-loop control" and "sub-control loop" is initialized
  - Sampling time of sub-control loop = '*m*'
  - Sampling time of parent-control loop = '*c*' times of '*n*'
- Determine the P-Controller Gain 'K_p'
- Initialize subscribers at the beginning = S_i
- new VM triggering actuating signal message throughput threshold = 'NT_h'
- Error Threshold range to tune the K_p = $E_{T,min}$, $E_{T,max}$
- Parent-control loop execute threshold for message throughput = P_T
  - During running the sub-control loop if the message throughput , when $T_a <= P_T$ then the parent-control loop is triggered

ALGORITHM

i. Start the JMS Provider
ii. Start loading the JMS Provider with initial number of subscribers as S_i
iii. for every '*m*' units of time run the sub-control loop as shown below
  a. measure the actual message throughput (T_a)
  b. If T_a is observed to be less than T_{ref} for more than 4 times, then trigger the parent-control loop (step iv.)
  c. Get T_a , compute the Error '*e*'
  i. If '*e*' is between $E_{r,min}$ and $E_{r,max}$ where $T_a < T_{ref}$, then adjust the P-Controller Gain 'K_p' to meet T_{ref}
  d. Repeat the steps from a. to d. for '*m*' times
iv. For every '*c* x *m*' units of time run the parent-control loop (i,e for every '*c*' sub-control loops)
  a. Define the prediction period '*p*' determines the number of parent control loops from the current parent control loop)
  b. Compute or predict S_{max} for '*p*' periods in advance $S_{max, p}$
  c. Compute or predict T_{ref} for '*p*' periods in advance $T_{ref, p}$
  d. Feed the predicted values to feedback controller
v. The controller will compare the $T_{ref, p}$ and the current message throughput T_a.
  a. If the $T_{ref, p}$ is more by NT_h than T_a, then trigger the actuator to provision new Virtual Machine
  b. If the $T_{ref, p}$ is less by $E_{r,max}$ than T_a, then tune the P-Controller Gain 'K_p'

VI. IMPLEMENATATION AND ANALYSIS

The adaptive control discussed is implemented in Java using an experiment data collected on Apache JMS Provider ActiveMQ [35] running on Ubuntu Linux 10.04 , i5 Intel 2 GHz CPU, 4 GB RAM, 1 TB Hard disk. A sample custom JMS application is run to generate the experiment data. A single JMS topic and a single publisher are used. The subscribers are increased which read different messages from the JMS Topic that are published. The data is collected on the explained experimental setup.

Then the proposed solution is run offline on the experimental data to examine the improvement in the message throughput, without running the proposed controller on the ActiveMQ server online.

The following are the different steps performed for implementing and evaluating the performance of the proposed solution.

- The model parameters (as in Equation (7)) are estimated with two different data sets. The parameters with least error are identified and used for the controller
- Operating range of maximum possible number of subscribers is determined for best possible message throughput, which is between 60 to 90 subscribers
- Based *on* the operating range, *the* P-controller Gain (K_p) is calculated as 2.67 and the Reference Message Throughput (T_{ref}) is determined as 220.
- The Feedback Controller and Predictor are implemented based on the values of P-controller Gain (K_p) and Reference Message Throughput (T_{ref}). The improvement in the message throughput using the proposed controller is evaluated in comparison with the actual message throughput.

We explain the implementation details of modeling the JMS Provider, the controller and discuss the results below.

B. Modeling JMS Provider

The model parameters 'a' and 'b' of the Equation (7) are estimated using the ARMA model where the actual message throughput is measured by linearly increasing the number of subscribers, and predicting the Message throughput. The error percentage is computed between the measured throughput and the predicted throughput. The experiments are run with two different data sets. The Table I shows the estimated model parameters for both the data sets with their error. We observe that the values a = 0.91 and b = 0.12 proved to have a lesser percentage of prediction error.

TABLE I. MODEL PARAMETER ESTIMATION

Data Sets	Model Parameter Estimation		
	a	***b***	***Percentage of Error***
Data Set 1	1	0.28	9.12
Data Set 2	0.91	0.12	8.33

Now using these constants the JMS Provider model in z-Transform is represented as the equation (9) below, using the model parameters estimated.

$$G(z) = 0.12/(z - 0.91) \quad (9)$$

The Fig 10 and Fig 11 show the parameter estimation with actual message throughput (Ta) and the predicted throughput (Tpred). The message throughput in these figures is number of messages per second. In Fig 10 the number of messages is plotted against the increasing number of subscribers. There is a saturation of message throughput after a certain number of subscribers.

C. Adaptive Control

The Fig 12 below shows the performance evaluation of the message throughput without Controller and with adaptive controller proposed in this paper. We observe that the message throughput using proposed Controller is better by about 25 % which is a significant improvement in message throughput over the throughput without controller. We can notice that there are spikes where there is a sudden increase of the number of subscribers. The actual message throughput has reduced suddenly in such cases, but using a P-Controller tuning along with the predictor, provided the adaptive control and has regulated the throughput to be in the operating range between 200 and 250.

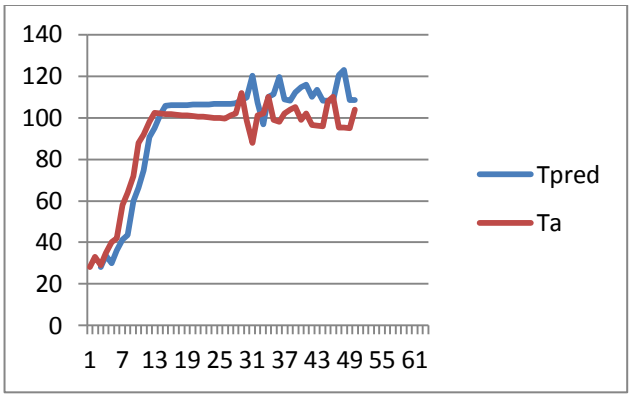

Figure 10. Model Parameter Estimation – Data set 1

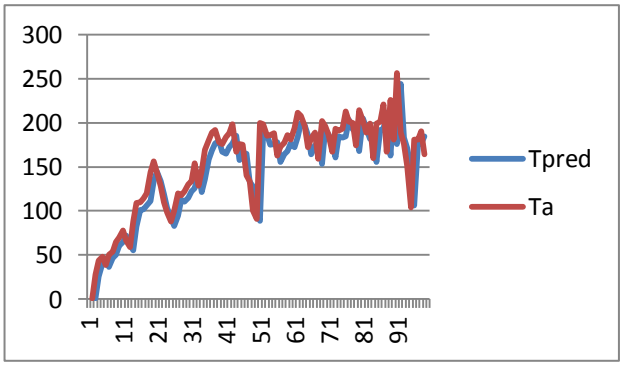

Figure 11. Model Parameter Estimation – Data Set 2

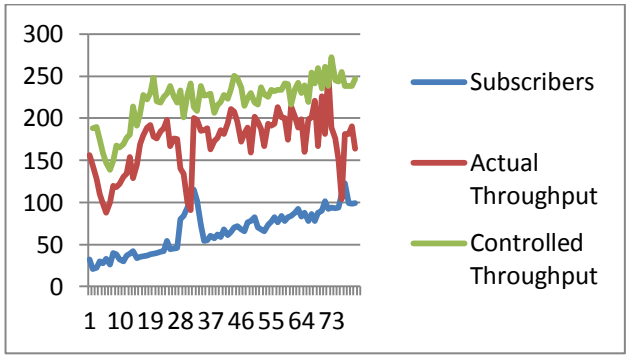

Figure 12. Performance Evaluation of Message Throughput using Adaptive Control

1) Feedback Controller

From the experimental data the value of P-Controller Gain Kp is determined as 2.67. This is computed by adjusting the value of Kp between 2 and 4 and the average Kp is computed. The reference message throughput is computed from the operating range average as 220. By tuning Kp the output of the controller is adjusted which is nothing but the tuning of Smax to obtain the desired reference message throughput. But our implementation has shown that the value of Smax is typically around 59 with a maximum value of 120. The optimal operating range of Smax is $59 \leq Smax \leq 90$. The Table II shows the operating range limits of Kp and Smax.

2) Predictor

We implemented the Triple Exponential Smoothing predictor using openforecast Java API [36]. The Table III shows the different values chosen for Predictor.

TABLE II. OPERATING RANGES

P-Controller Gain (K_p) Range		
K_p Range	S_{max} Range	T_{ref}
2.67, 3, 2, 2.4	59-90	220

TABLE III. PREDICTOR PARAMETER VALUES

Predictor Values		
Triple Exponential Smoothing Coefficients	$E_{r,max}$	*Forecast period (p)*
0.2, 0.6, 0.6	70	1

The Fig 13 below shows the predicted values of the Smax (Smaxp) and the Tref. (Trefpred). These values are predicted using Triple Exponential Smoothing with coefficients shown in the Table III. In our experiment the parent control loop is run once the Trefpred starts decreasing less than 150, which is less than the Tref by 70. From the Fig 13, the predicted Tref (TrefPred) is less than 150, the P-Controller gain Kp is tuned to a value of 4 such that message throughput is regulated without fluctuations. The Er,max is set to as 70 (220-150). The predictor adjusts the Kp once the Tref,pred is less by Er,max (70) than original Tref. In our experimental data we didn't simulate the condition of the Ta exceeding the threshold to trigger addition Virtual Machine requests.

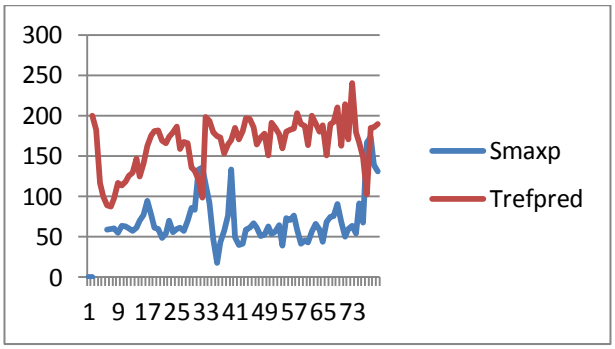

Figure 13. Prediction of Smax and Tref

VII. CONCLUSION

We observed using the P-Controller will have a distinct improvement in the message throughput of the Enterprise messaging servers. Our experiments are currently limited to using the P-Controller only which helps in reducing the rise time [37], but in order to obtain reduce the overshoot and settling time using the PI-Controller [17] is more helpful. Additionally, the parameter estimation is done on experimental data and only two data set samples are used. Our results are based on a simulation like environment as the P-Controller is not directly verified online on the JMS Provider. Our experiments are rather run on the data collected from the JMS Provider by running a sample application with one publisher and one JMS topic. We infer that applicability of adaptive control systems will have significant improvement on the performance of the Enterprise messaging servers in distributed computing systems.

VIII. FUTURE WORK

There is a scope of improvement of the solution explained in this paper. We intend to extend the experiments to adjust the model parameters during runtime such that model represents the behavior of the system to be controlled in a real time. Also, we want to examine the SASO [17] properties of our control system to determine the controller stability and accuracy. We also want to verify the solution on the ActiveMQ server with varying publishers and topics, not limiting to the subscribers only.

We suggest exploring a hybrid approach where techniques like fuzzy control [38] can be used in conjunction with the classic PI controller which can show better performance. The applicability of fuzzy control enables creating a knowledge base of rules and can be evaluated against using Triple Exponential Smoothing for predicting future message throughput. These rules can be helpful when the Enterprise Messaging servers are used in massively large distributed computing systems. We are also studying the different aspects of Data Mining which can be used to build novel prediction algorithms there by the adaptive control system is more robust.

REFERENCES

[1] Message Oriented Middleware (MoM): "http://en.wikipedia.org/wiki/Message-oriented_middleware"

[2] Matjaz B.Juric, S.Jeelani Basha, Rick Leander, Ramesh Nagappan, "Professional J2EE EAI", Shroff Publishers 2005

[3] Service Level Agreement , "http://en.wikipedia.org/wiki/Service-level_agreement"

[4] Service Level Objective, "http://en.wikipedia.org/wiki/Service_level_objectives"

[5] JMS Specification: "http://www.oracle.com/technetwork/java/javaee/tech/index.html"

[6] JEE Specification: "http://www.oracle.com/technetwork/java/javaee/tech/index.html"

[7] JMS Providers: "http://en.wikipedia.org/wiki/Java_Message_Service"

[8] A. Robertsson, B. Wittenmark, M. Kihl, and M. Andersson, "Design and evaluation of load control in web server systems", IEEE American Control Conference, 2004

[9] JMS Performance Benchmarks : "http://www.codeproject.com/KB/showcase/PerformanceBenchmarks.aspx"

[10] http://www.precisejava.com/javaperf/j2ee/JMS.htm#JMS111

[11] Bruce Snyder, Dejan Bosanac and Rob Davies, "ActiveMQ In Action", Dreamtech Press, 2011

[12] Pradeep Padala, Xiaoyun Zhu, Mustafa Uysal et al. Adaptive Control of Virtualized Resources in Utility Environments. In the proceedings of the EuroSys 2007

[13] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, Andrei Voronkov, "Complexity and expressive power of logic programming", ACM Computing Surveys 2003

[14] Karl J.Astrom and Bjorn Wittenmark, "Adaptive Control", Pearson Education, 2009

[15] Yan Liu, Ian Gorton, "Performance Prediction of J2EE Applications using Messaging Protocols", Proceedings of 2005 Symposium on Component-based Software Engineering

[16] Michael Menth, Robert Henjes, Christian Zepfel, and Sebastian Gehrsitz, "Throughput Performance of Popular JMS Servers", ACM SIGMETRICS '06

[17] Joseph L. Hellerstein, Yixin Diao, Sujay Parekh, and Dawn Tilbury, "Feedback Control of Computing Systems", John Wiley 2004

[18] Ying Lu, Avneesh Saxena and Tarek E Abdelzaher, "Differentiated Caching Services; A Control-Theoretical Approach", IEEE International Conference on Distributed Sysytems, 2001

[19] Keqiang Wu, David J. Lilja, Haowei Bai, "The Applicability of Adaptive Control Theory to QoS Design: Limitations and Solutions", IEEE Parallel and Distributed Processing Symposium, 2005

[20] Ying Lu, Tarek Abdelzaher and Gang Tao, "Direct Adaptive Control of A Web Cache System", Proceedings of the American Control Conference, Denver, Colorado, 2003

[21] Giovanna Ferrari, Santosh Shrivastava,Paul Ezhilchelvan, "An Approach to Adaptive Performance Tuning of Application Servers", IEEE International Workshop on QoS in Application Servers, 2004

[22] Seungwan Ryu, Chulhyoe Cho,"PI-PD-controller for robust and adaptive queue management for supporting TCP congestion control", Simulation Symposium, 132 - 139 April 2004

[23] Ravi Kumar Gullapalli, Dr.Chelliah Muthusamy and Dr.A.Vinaya Babu , "Control Systems application in Java based Enterprise and Cloud Environments – A Survey" , IJACSA, Volume 2, No 8, August 2011

[24] Ravi Kumar Gullapalli, Dr.Chelliah Muthusamy, Dr.A.Vinaya Babu and Raj N. Marndi, "A FEEDBACK CONTROL SOLUTION IN IMPROVING DATABASE DRIVER CACHING", IJEST, Vol 3, No 7, July 2011

[25] Dr. Wolfgang Winter , "Applying control theory concepts in software applications", http://www.theserverside.com/feature/Applying-controltheory-concepts-in-software-applications

[26] Richard Monson-Haefel and David A.Chappell, "Java Message Service", O'Reilly 2001

[27] Single Input Single Output : "http://en.wikipedia.org/wiki/Single-input_single-output_system"

[28] Virtual Machines : "http://en.wikipedia.org/wiki/Virtual_machine"

[29] Erwin Kreyzig, "Advanced Engineering Mathematics", John Wiley and Sons

[30] ARMA: http://en.wikipedia.org/wiki/Autoregressive_moving_average_model

[31] MICHAEL L.JOHNSON and LINDSAY M.FAONT Parameter Estimation by Least Squares Error :" http://mljohnson.pharm.virginia.edu/pdfs/174.pdf"

[32] P-Controller, "http://en.wikipedia.org/wiki/Proportional_control"

[33] Z-Transform: Saed Vaseghi, "http://dea.brunel.ac.uk/cmsp/Home_Saeed_Vaseghi/Chapter04-Z-Transform.pdf

[34] Triple Exponential Smoothing: "http://itl.nist.gov/div898/handbook/pmc/section4/pmc435.htm"

[35] Apache ActiveMQ, "http://activemq.apache.org/"

[36] Openforecast API, "http://openforecast.sourceforge.net/docs/"

[37] Jinghua Zhong, "PID Controller : A Short Tutorial", Purdue University, 2006

[38] Jan Jantzen, "Design of Fuzzy Controllers", Tech report, 1988, Technical University of Denmark

AUTHORS PROFILE

Ravi Kumar G is working as a Technical Expert in Hewlett-Packard., Bangalore, India. He obtained his M.Tech in Computer Science from Birla Institute of Technology, Mesra,India. He is currently pursuing Ph.D from JNTU Hyderabad,AP, India.

Dr.Chelliah Muthusamy is Academic Relations Head at Yahoo, Bangalore,. He obtained his Ph.D from Georgia Tech and M.Sc(Engg) in Computer Science from Indian Institute of Science(IISc), Bangalore India

Dr.A.Vinaya Babu is a Professor of Computer Science working as Principal, JNTUH College of Engineering, JNTU Hyderabad, AP, India. He obtained his Ph.D and M.Tech in Computer Science from JNTU, Hydreabad.