Integrated Monitoring and Control for Performance Management of Distributed Enterprise Systems

Rajat Mehrotra*, Abhishek Dubey†, Sherif Abdelwahed*, Asser Tantawi‡
*Electrical and Computer Engineering, Mississippi State University, Miss. State, MS
†Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN
‡IBM TJ Watson Research Center, Hawthorne, NY

I. Introduction

Self-managing techniques in distributed systems have been investigated recently for optimizing operational cost and multidimensional quality of service metrics including response time, throughput and reliability. Practical applications include task scheduling [1], bandwidth allocation and QoS adaptation in web servers [2], load balancing in e-mail and file servers [3], and CPU provisioning [4]. The challenge in developing these techniques such that they remain valid even when the systems are under uncertain and dynamic operating conditions, is to identify and learn the system model. Then only we can develop an appropriate management structure.

Contribution of Our Work: In this paper, we present an integrated framework for estimating workload patterns, system performance using mathematical models and model-predictive control for managing the system's Quality of Service parameters. Our approach starts with performing system model identification through extensive experimentation and then identifies the parameters and underlying model structure of the system using regression and queuing theory techniques. Later, the value of model parameters is estimated and refined using Kalman Filters(KF). We use auto regressive moving average filters for workload forecasting. The effectiveness of our approach has been demonstrated by using a model predictive controller to minimize power consumption of a multi-tier enterprise system while maintaining the system response time within a desired level. A detailed description of these results is available as a technical report [5].

System Setup: This work utilizes IBM Web Sphere Application Server Community Edition with Daytrader as the representative application. Modified version of Httperf is used to generate client requests (see [5]). Table I summarizes configuration of physical machines. Numerous experiments were performed to understand the system behavior with respect to system utilization, various work load profiles, bottleneck resource utilization, and their impact on system performance to develop analytical models. Next subsections describe them in detail.

II. System Modeling Approach

Queuing Model With Runtime Estimation of Various Parameters: Queues are a useful abstraction for understanding the nature of web servers. Typically, a new web request has to wait in a queue for release of computational resources from older requests before entering in to the system. Therefore, the total service time of the enterprise system is directly affected by the queuing policy at each tier. During these tests, we used an equivalent open single-tier queuing model to approximate the combined behavior of all tiers that is shown in Fig. 1. Here S is the average service time for each request. D is a delay corresponding to the time taken to process the request in all subsequent tiers.

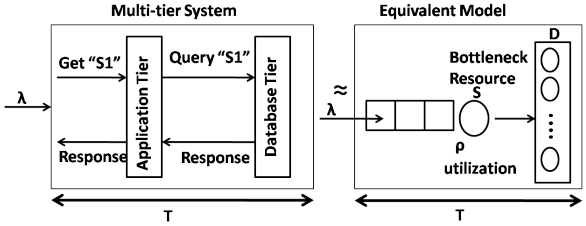

Fig. 1. An equivalent queuing model for the two-tier system.

To obtain the relationship between the state vector and the observation vector on-line, we consider *Processor Sharing(PS)* queue system. From experiments, we found that this model provides good estimation of the system behavior and is easier to analyze compared to *Limited Processor Sharing (LPS)* queue system. We implemented an *exponential Kalman filter* to predict the computational nature of the incident requests over web server by estimating the S and D of a request by observing the current average response time of the incident request and request arrival rate on the web server.

This filter uses an $M/G/1/\infty$ PS queuing model and considers variation in S and D at previous approximation to estimate the S and D at next sample time. It operates on the exponential transformation of the system state variables that allows us to enforce the feasibility constraints, $S, D \geq 0$. Such constraints are not possible in typical Kalman filter implementations as described in [6].

Note: We can approximate the system as a M/G/1/∞ PS queue only if the total number of requests in the system are less than the maximum concurrency limit, or *the bottleneck resource utilization is less than* 1 that represents an infinite PS queue model. Hence, we identify the operating regions through bottleneck utilization and analyze the system in the infinite PS queue region only.

The KF equations, written in the terms of exponentially transformed variables, $[x1 \in \mathbb{R}; x2 \in \mathbb{R}]$ s.t. $S = exp(x1)$ and $D = exp(x2)$ are as follows. Note that this transformation ensures $S, D \in \mathbb{R}^+$: For a given timed index of observa-

TABLE I
PHYSICAL MACHINE CONFIGURATION.

Physical M/C	Cores	Description	RAM	DVFS	Virtual Machines
Nop01	8	2 Quad core 1.9GHz AMD Opteron 2347 HE	8GB	No	Nop04,Nop07 (Development Machines)
Nop02	4	2.0 GHz Intel Xeon E5405 processor	4GB	No	Nop05,Nop08 (Client Machines)
Nop03	8	2 Quad core 1.9GHz AMD Opteron 2350	8GB	Yes	Nop06,Nop09 (Application server)
Nop10	8	2 Quad core 1.9GHz AMD Opteron 2350z	8GB	Yes	Nop11,Nop12 (Database Server)

tion, k, the equations $\begin{pmatrix} exp(x1_k) \\ exp(x2_k) \end{pmatrix} = \begin{pmatrix} exp(x1_{k-1}) \\ exp(x2_{k-1}) \end{pmatrix} + N(0, \mathcal{Q})$ and $T = exp(x1_k) * (1/(1 - \lambda_k * exp(x1_k))) + exp(x2_k) + V(0, \mathcal{R})$ define the state update dynamics and observation. N and V are Gaussian process and measurement noises with mean zero and covariances $\mathcal{Q}$ and $\mathcal{R}$ respectively. The assumption that this process is Gaussian in nature is based on our limited observation. We do not claim that this will be applicable in all situations. Predicted bottleneck utilization is given by $\hat{\rho}_k = \lambda_k * exp(x1_k)$.

Note: For stability reasons and infinite PS queue assumption, the Kalman filter does not update its state when the predicted bottleneck resource utilization becomes equal to 1.

Model for Power Consumption: We extend our previous power consumption model described in [6] with additional parameters and data to achieve greater accuracy. Main observation during our experiments (details available in [5]) was that power consumption model of a physical machine is non-linear because power consumption in these machines depends not only upon the CPU core frequency and utilization, but also depends non-linearly on other power consuming devices e.g. hard drive, CPU cooling fan etc. As a result, a look-up table with near neighbor interpolation was used as the power consumption model of the physical machine. Combination of CPU frequency, and aggregate CPU core usage of the physical machine was used as a key of the lookup table to access the corresponding power consumption value. This aggregate power model was utilized mainly for the controlled experiments described in section IV.

III. THE CONTROLLER

This section describes the implementation of a feedback control based online predictive controller that uses the Kalman filter and the queuing model identified in previous section to maintain the multi-dimensional QoS demands. This controller is similar to the *L0 Controller* described in [7]. It predicts the aggregate response time of the incident requests and the estimated power consumption during the next sample time (look-ahead horizon N) of the system based on different possible combinations of control inputs (CPU core frequency). It optimizes the system behavior in terms of QoS objectives by continuous observation of the system measurements and choosing the best control input for the system in next sample interval.

System Variables: We have chosen a small set of most relevant parameters from list in [5] for our predictive controller to show the performance of our modeling approach. The chosen control input is the *CPU core frequency* due to its impact on the system performance in multiple dimensions for response time of the system and power consumption. Experiments from [5] indicate that the higher value of application queue represents contention in computational resources of the application and total response time value indicates system's capability to process the requests lying in system queue in a timely manner. Therefore *System queue size* and *response time* were the chosen state variables while *power consumption* was the performance variable. These are also the typical variables measured in web service industry and used to define multi-dimensional service level agreements (SLA).

Control Objective: We try to minimize the application queue size and total response time as one of the component in cost function J (described later in this section).

Plant Model: The queuing model identified in the previous section was used to estimate the state of the managed system.

Controller Model: To combine the power consumption, QoS and the predicted response time, the controller uses a different internal model (not same as plant model, which is used for state estimation). The controller model uses the estimated system state, predicted response time and predicted power consumption to make the system decisions. Kalman filter is used at run-time to estimate the service time $\hat{S}_t$ of the incident request at current frequency $u(t)$, which is then used by the controller to estimate the average service time for the next sampling interval.

Request Forecaster: An *autoregressive moving average* model is used as estimator of the environmental input with user specified weights on the current and previous arrival rates for accurate prediction.

Control Algorithm and Performance Specification: We use a limited look ahead controller algorithm, which is a type of model predictive control. Starting from a time t_0, the controller solves an optimization problem defined over a predefined horizon ($t = 1...N$) and chooses the first control input (CPU core frequency $u(t_0)$) that minimizes the total cost of operating the system J within the prediction horizon. During this work, we set the horizon to $N = 2$ to reduce the computation overhead.

The cost function (J) is the weighted conjunction of drift of system state from the desired set point of the system state (desired maximum queue size, desired maximum response time) and power consumption (desired power consumption is 0). The power consumption is predicted with the help of *lookup table* generated from the system power consumption model, the current frequency of the CPU core, and aggregate system utilization of the physical server.

IV. CASE STUDY: POWER CONSUMPTION AND RESPONSE TIME MANAGEMENT

This section uses the concepts introduced in the earlier sections for managing server power consumption while maintaining the predefined QoS requirement of minimum response

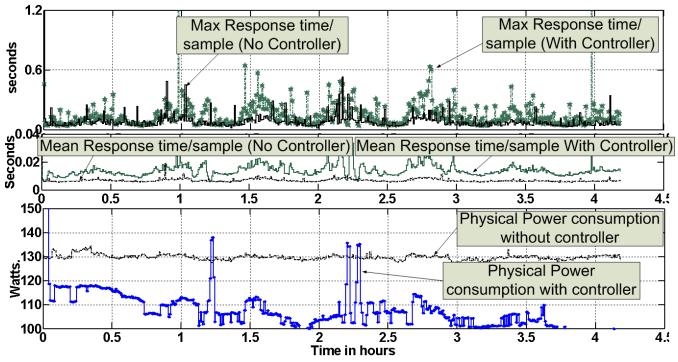

Fig. 2. Comparison of of results with and without controller: Sampling period=30 seconds. Std deviation for all response measurements=0.02 seconds (without controller), 0.019 seconds(with controller)

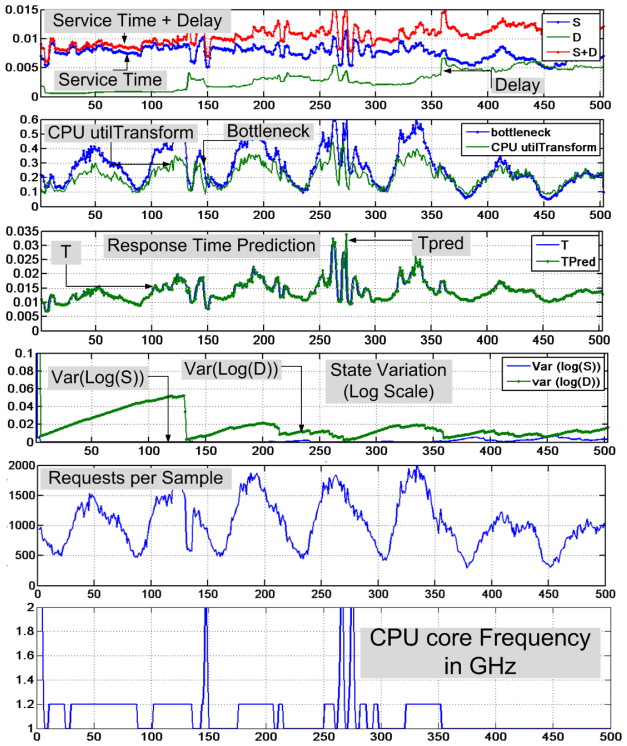

Fig. 3. Online exponential Kalman filter output corresponding to the experiment with controller. Service time and delay are in millisecond range. Response time is specified in seconds.

time under a time varying dynamic workload for application hosted in virtualized environment [5]. During this study, we performed two separate experiments to operate a multi-tier enterprise service *(Daytrader)* described in [5] with and without predictive controller and compared the cost of operating the system in terms of response time and power consumption over a periods of 4 hours.

Analysis of the controller results: Fig. 3 shows the Kalman filter tracking average response time of the incident requests and bottleneck utilization with high accuracy. According to sub-figure 3, predicted response time from the Kalman filter T_{pred} and actual response time T observed at web server are also very close to each other, which indicates accuracy of the Kalman filter estimation. The controlled version runs at a lower frequency most of the times that results into considerable amount of power saving (18%) over a period of four hours of experiment (fig 2) compared to the baseline experiment without controller at max frequency all the time. Fig 3 shows the controller changing the frequency of the CPU core at very few occasions, but it is able to identify the sudden increase in the incident request rate which reflects adaptive nature of the controller in case of dynamic load conditions. This experiment shows that the predictive controller has a negligeable negative effect on the response time as well as CPU (not shown in figure) and memory utilization (not shown in figure), but greatly reduces the power consumption.

V. Conclusion

We have presented a simple and novel approach to develop models with low variance for multi-tier enterprise systems. We showed that the developed model can be integrated with a predictive control framework for dynamically changing the system tuning parameters to achieve a pre-specified QoS objective. The results shown in section IV shows that the developed Kalman filter tracks the system model parameters at run time with high accuracy. Additionally, the proposed power consumption model of the system used by the controller predicts the overall physical server power consumption well (95% accurate). Using this model we showed that we can optimize system performance and achieve 18% reduction of power consumption in four hours of experiment in single server without affecting the response time severely. Furthermore, the experimental results (CPU and RAM consumption with and without the controller) indicates that the proposed approach has low run-time overhead in terms of computational and memory resources. We further plan to extend this framework and verify its performance over a cluster of multi-tier computing systems in hierarchical fashion as described in [7].

Acknowledgment This work was supported in part by the NSF SOD Program, contact number CNS-0804230.

References

[1] Anton Cervin, Johan Eker, Bo Bernhardsson, and Karl-Erik. Feedback–feedforward scheduling of control tasks. *Real-Time Syst.*, 23(1/2):25–53, 2002.

[2] T.F. Abdelzaher, K.G. Shin, and N. Bhatti. Performance guarantees for web server end-systems: a control-theoretical approach. *Parallel and Distributed Systems, IEEE Transactions on*, 13(1):80–96, Jan 2002.

[3] Chenyang Lu, Guillermo A. Alvarez, and John Wilkes. Aqueduct: Online data migration with performance guarantees. In *FAST '02: Proceedings of the 1st USENIX Conference on File and Storage Technologies*, page 21, Berkeley, CA, USA, 2002. USENIX Association.

[4] Dara Kusic, Nagarajan Kandasamy, and Guofei Jiang. Approximation modeling for the online performance management of distributed computing systems. In *ICAC '07: Proceedings of the Fourth International Conference on Autonomic Computing*, page 23, 2007.

[5] Rajat Mehrotra, Abhishek Dubey, Sherif Abdelwahed, and Asser Tantawi. Model identification for performance management of distributed enterprise systems. Technical Report ISIS-10-104, Institute for Software Integrated Systems, Vanderbilt University, April 2010.

[6] Abhishek Dubey, Rajat Mehrotra, Sherif Abdelwahed, and Asser Tantawi. Performance modeling of distributed multi-tier enterprise systems. *SIGMETRICS Performance Evaluation Review*, 37(2):9–11, 2009.

[7] N. Kandasamy, S. Abdelwahed, and M. Khandekar. A hierarchical optimization framework for autonomic performance management of distributed computing systems. In *Proc. 26th IEEE Int'l Conf. Distributed Computing Systems (ICDCS)*, 2006.