12-2003

Using Dynamic Optimization for Control of Real Rate CPU Resource Management Applications

Varin Vahia
Oregon State University

Ashvin Goel
Oregon Graduate Institute of Science & Technology

David Steere
Oregon Graduate Institute of Science & Technology

Jonathan Walpole
Oregon Graduate Institute of Science & Technology

Molly H. Shor
Oregon State University

Paper ID: ACC03-SHT0068

Using Dynamic Optimization for Control of Real Rate CPU Resource Management Applications*

Varin Vahia†, Ashvin Goel‡, David Steere‡, Jonathan Walpole‡, Molly H. Shor†

Abstract:

In this paper we design a proportional-period optimal controller for allocating CPU to real rate multimedia applications on a general-purpose computer system. We model this computer system problem in to state space form. We design a controller based on dynamic optimization LQR tracking techniques to minimize short term and long term time deviation from the current time stamp and also CPU usage. Preliminary results on an experimental set up are encouraging.

Introduction/Motivation:

General-purpose computers must fulfill the CPU and network needs of real-rate multimedia and sensor-based real-time applications. These real-rate flows have bounded end-to-end delay requirements in addition to other jitter and dithering requirements. Current approaches to tackle this problem in the computer system community, such as selective data dropping so that the data rate matches available resources, specialized hardware for particular applications, and reservation-based schemes perform well, but they have their own limitations. For example, in poorly designed data dropping systems, quality can quickly degrade to unacceptable levels. Reservation based schemes waste resources, as the reserved amount of resources may not be optimal. Design and testing of such methods is approached as an experimental science.

Recently, several researchers have started looking into the use of feedback control algorithms for resource allocation. For example, Goel et al. designed a PI controller for Network and CPU resource management for real-rate systems [1], [2]. In this paper we design a feedback controller for real-rate systems based on optimal control and dynamic programming techniques. We set up the computer system problem as a LQR infinite horizon dynamic optimization tracking problem with modified state feedback.

* This work was supported part by DARPA/ITO under the Information Technology Expeditions, Ubiquitous Computing, Quorum, and PCE5 programs, NSF grants ECS-998843 and CCR-9966440 and EIA-0130334, and by Intel
† Department of Electrical and Computer Engineering, Oregon State University, Corvallis OR 97331
‡ Department of Computer Science and Engineering, Oregon Graduate Institute, Portland.

Overview of System:

This controller is a modification of the PI controller designed earlier by Goel et al [1], [2] and uses the same system setup. We describe the system briefly here. For more details, refer to [1], [2]. Our multimedia real-rate systems use a pipelined abstraction from source to sink having intermediate stages (Figure 1). These real-rate flows require CPU, network and possibly other resources at various pipeline stages.

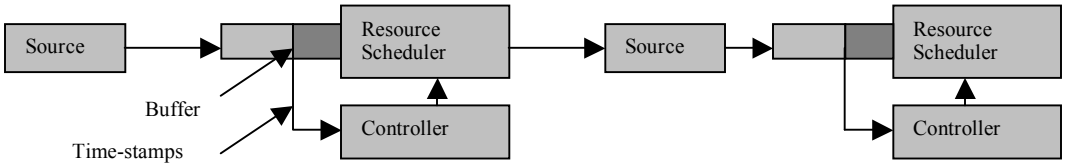

Figure 1. Pipeline configuration

At each pipeline stage, a feedback mechanism decides how much of the resource to allocate based on only local measurements. This will separate the control of each pipeline stage from the others and allow cascading of individual stages without difficulty of implementation.

To control such a system, we need a measurement that tells how far ahead or behind a flow is from the sender's real rate. This can be achieved by marking each packet with a time stamp that indicates the time offset from the first packet in the flow. These time-stamps represent the application's logical time. For example, this logical time may be the playback time of the video application. The real-rate mechanism aims to transmit data in such a way that real system time stays aligned with these time stamps.

If t_i is the time-stamp of the packet at the head of the buffer, where i is the current sampling instant of the controller, and the sampling period is s, then $t_i - t_{i-1}$ would be the logical time-stamp interval between the packets transmitted one sampling period apart and the real rate of the flow at sampling instant i would be $(t_i - t_{i-1})/s$. We define the error variable z^1 to be the difference between this current real rate and the target real rate of one.

We also need another error variable z^2 to keep track of the long-term behavior of the real rate. For this, we can simply add the individual real rates at each sampling instant. This should be equal to the total real time progress made until the current time. These two error variables need to be minimized.

$$z_i^1 = (1/s)(t_i - t_{i-1}) - 1 \qquad (1)$$

$$z_i^2 = (1/s)t_i - i$$

System Modeling:

The next step needed to design a controller is the system's mathematical model, which will describe our system. The following model describes our system reasonably well.

$$t_i = t_{i-1} + (1/k_i)p_{i-1} - (1/k_i)n_{i-1} \qquad (2)$$

Here, p_{i-1} is the allocation assigned during the last period, n_{i-1} is the amount of assigned allocation not used by the application in the last period, and k_i is the variable that relates the amount of progress made to the allocation actually used by the application. The variable k_i is random by nature and varies by a factor of two. This variable relates the amount of progress made in terms of time-stamps to the cycles actually run by the system.

Control Design Setup as a Dynamic Optimization Tracking Problem:

We have a state-space system model (2) and errors that we want to minimize (1). The error variable z^1 includes the previous state t_{i-1}, so we augment the state space in order to obtain error variables in terms of current state variables. These augmented states are selected as $x_i^1 = z_i^1 + 1$ and $x_i^2 = z_i^2 + 1$. Then the new state equations are:

$$x_i^1 = (1/s)(t_i - t_{i-1}) - 1 = (1/(k_i s))p_{i-1} + (1/(k_i s))n_{i-1}$$
$$x_i^2 = x_{i-1}^2 + (1/(k_i s))p_{i-1} + (1/(k_i s))n_{i-1}$$

Also we have $p - n =$ number of cycles run, which is directly measurable from the system. The controlled variable p dominates, and the noise *n* relatively very small, so we neglect *n* in our control design. Taking *u* = *p,* we write the state equations in matrix form:

$$\begin{bmatrix} x_i^1 \\ x_i^2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{i-1}^1 \\ x_{i-1}^2 \end{bmatrix} + \begin{bmatrix} 1/(k_i s) \\ 1/(k_i s) \end{bmatrix} u_{i-1}$$

The next step is to come up with a cost function to be minimized. In our case, we want the augmented states to track unity. In addition we also want to minimize the resource allocated to avoid over-allocating to the application, so we include the input variable in the cost function. After all these considerations, the cost function is given in the equation below:

$$J = \lim_{N \to \infty} \left\{ \sum_{i=0}^{N} ((x_i - \bar{x})' Q_i (x_i - \bar{x}) + u_i' R_i u_i) \right\} \qquad \text{Where, } \bar{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Here, x is the state vector, u the input allocation and i the index of the sampling period. The matrices Q and R are the weightings given to the state and input variables and they are positive semi-definite and positive definite respectively. We have selected Q to be around 1000 times more than R. It is also possible to weight the first and second state variables differently. We have weighted them the same.

Minimizing this cost function, we find that we should use the allocation given in equation (3), which is the steady-state solution to the standard LQR tracking problem with the augmented states. This steady-state solution is applicable under the condition that pair (A, B) is controllable. This condition is satisfied in our case.

$$u(x_i) = Kx_i + K^v v \qquad \begin{aligned} &\text{Where, } K = -(B'SB + R)^{-1} B'SA \\ &S = A'[S - SB_k (B'SB + R)^{-1} B'S]A + Q \\ &K^v = (B'SB + R)^{-1} B' \\ &v = (A - BK)'v + Q\bar{x} \end{aligned} \qquad (3)$$

K is the feedback gain for our system and K^v is the feed forward gain. S is the solution to the Riccati equation. We must solve this Riccati equation in order to calculate the gain. I have solved the Riccati equation using generalized eigenvalue problem [7, 8]. We wanted to have very simple controller with less overhead in computation. So, we have solved the equation analytically.

Generalized Riccati equation is:

$$F^{'}XF - X - F^{'}XG_1(G_2 + G_1^t XG_1)^{-1}G_1^{'}XF + H = 0$$

Comparing it with our Riccati equation,

$$A = F, G_1 = B, G_2 = R, Q = H$$
$$G = G_1 G_2^{-1} G_1^{'}$$

$$G = 1/R \begin{bmatrix} 1/c \\ 1/c \end{bmatrix} \begin{bmatrix} 1/c & 1/c \end{bmatrix} = 1/R \begin{bmatrix} 1/c^2 & 1/c^2 \\ 1/c^2 & 1/c^2 \end{bmatrix}$$

$$M = \begin{bmatrix} F & 0 \\ -H & I \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

$$L = \begin{bmatrix} I & G \\ 0 & F' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1/(c^2R) & 1/(c^2R) \\ 0 & 1 & 1/(c^2R) & 1/(c^2R) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Hence, for getting eigenvalues,

$$M - \lambda L = \begin{bmatrix} -\lambda & 0 & -\lambda/c1 & -\lambda/c1 \\ 0 & 1-\lambda & -\lambda/c1 & -\lambda/c1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1-\lambda \end{bmatrix}$$

$$|M - \lambda L| = -\lambda \begin{vmatrix} 1-\lambda & -\lambda/c1 & -\lambda/c1 \\ 0 & 1 & 0 \\ -1 & 0 & 1-\lambda \end{vmatrix} - -\lambda/c1 \begin{vmatrix} 0 & 1-\lambda & -\lambda/c1 \\ -1 & 0 & 0 \\ 0 & -1 & 1-\lambda \end{vmatrix} + -\lambda/c1 \begin{vmatrix} 0 & 1-\lambda & -\lambda/c1 \\ -1 & 0 & 1 \\ 0 & -1 & 0 \end{vmatrix}$$

Solving $|M - \lambda L| = 0$, we get following eigenvalues,

$$\lambda = \frac{(2c_1 + 3) \pm \sqrt{4c_1 + 5}}{2 * (c_1 + 1)}, 0$$

Where,

$$c_1 = R * (k_i s)^2$$

Now computing eigenvectors,

$$\begin{bmatrix} -\lambda & 0 & -\lambda/c_1 & -\lambda/c_1 \\ 0 & 1-\lambda & -\lambda/c_1 & -\lambda/c_1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1-\lambda \end{bmatrix} \begin{bmatrix} e1 \\ e1 \\ e3 \\ e4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Solving this we get,

$$e1 = e3, e2 = -\left(\frac{-\lambda}{1-\lambda}\right)e1, e4 = \left(\frac{-\lambda}{(1-\lambda)^2}\right)e1$$

So, for $\lambda = 0$

$$\text{eigenvecto} \quad \text{r1} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

For other stableλ,

$$eigenvecto \;\; r2 = \begin{bmatrix} 1 \\ e2 \\ 1 \\ e4 \end{bmatrix}$$

So ,

$$U = \begin{bmatrix} 1 & 1 \\ 0 & e2 \\ 1 & 1 \\ 0 & e4 \end{bmatrix}$$

$$\Rightarrow U_1 = \begin{bmatrix} 1 & 1 \\ 0 & e2 \end{bmatrix}, \; U_2 = \begin{bmatrix} 1 & 1 \\ 0 & e4 \end{bmatrix}$$

$$S = U_2 U_1^{-1} = \begin{bmatrix} 1 & 1 \\ 0 & e4 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 0 & e2 \end{bmatrix}^{-1}$$

$$\Rightarrow S = \begin{bmatrix} 1 & 0 \\ 0 & e4/e2 \end{bmatrix}$$

This is the solution of the Riccati equation.

Hence, feedback gain K for our controller would be:

$$K = (B^{'}SB + R)^{-1} B^{'}SA$$

Computing the K,

$$B^{'}SB = \begin{bmatrix} 1/c & 1/c \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & e4/e2 \end{bmatrix}\begin{bmatrix} 1/c \\ 1/c \end{bmatrix}$$

$$\Rightarrow B^{'}SB = 1/c^2(1 + e4/e2)$$

$$B^{'}SA = \begin{bmatrix} 1/c & 1/c \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & e4/e2 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\Rightarrow B^{'}SA = \begin{bmatrix} 0 & 1/c(e4/e2) \end{bmatrix}$$

$$K = [1/c^2(1+e4/e2)+R]^{-1} * \begin{bmatrix} 0 & 1/c(e4/e2) \end{bmatrix}$$

Computing Feed forward gain K^v,

$$K^v = (B'SB+R)^{-1}B'$$

$$K^v = [1/c^2(1+e4/e2)+R]^{-1}\begin{bmatrix} 1/c & 1/c \end{bmatrix}$$

From equation (3) we can calculate *v*. So, now input will be:

$$u = K * x_{i-1} + K^v v$$

The steps to calculate allocation online would be:

1. Calculate eigenvalue,

$$\lambda = \frac{(2c_1+3)-\sqrt{4c_1+5}}{2*(c_1+1)}$$

Where, $c_1 = R*(k_i s)$

2. Calculate z for eigenvector.

$$e1 = e3 = 1$$

$$e2 = -\left(\frac{-\lambda}{1-\lambda}\right)e1$$

$$e4 = \left(\frac{-\lambda}{(1-\lambda)^2}\right)e1$$

3. Calculate feedback and feed forward gain.

$$K = [1/c^2(1+e4/e2)+R]^{-1} * \begin{bmatrix} 0 & 1/c(e4/e2) \end{bmatrix}$$

$$K^v = [1/c^2(1+e4/e2)+R]^{-1}\begin{bmatrix} 1/c & 1/c \end{bmatrix}$$

4. And finally calculate, input *u*.

$$u = K * x_{i-1} + K^v v$$

Statistics collected from running the system are used to estimate the average k_i in the state equation, and the estimated value of k_i is used to compute the controller. Two extreme approaches are to compute average k_i and compute the controller once, as done in the experiment below, or to update the controller frequently as k_i changes.

Test on experimental setup:

We have implemented this controller on the Linux-2.2 operating system using Kernel developed at OGI [4]. We have used a simple application structured as a producer and a consumer thread (Figure 2).

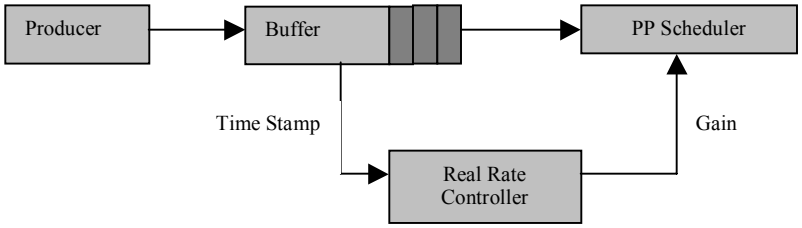

Figure 2 Producer Consumer application structure

The period for this producer-consumer system is 16 msec. The producer works for some number of cycles and writes data into the buffer. We pre specify the allocation to the producer at the command line. Producer varies the rate at which it produces data by factor of two from period to period. The consumer reads from the buffer and works on the data for the number of cycles allocated by the controller in each period.

Figure 3 shows the sample run of the experiment to test the controller. The allocation of the producer is kept fixed at around 20 % of CPU per period. The controller determines the allocation to the consumer to keep up with the producer's real rate. As can be seen, the long-term error and instantaneous error remain close enough to zero. When the producer changes the amount of data produced the consumer has to change its allocation to new value. At this instance, time errors deviate from zero instantaneously. But in a very short time, the controller adjusts the allocation to the consumer and errors return to zero. This instantaneous deviation is less than 20 msec, which is quite acceptable. Also, note that there is no accumulation of error as can be seen from plot of long tem error. Cost function J's value is decreased by around 7 % from the cost of the previous PI controller. So, the controller performs quite well on the experimental setup.

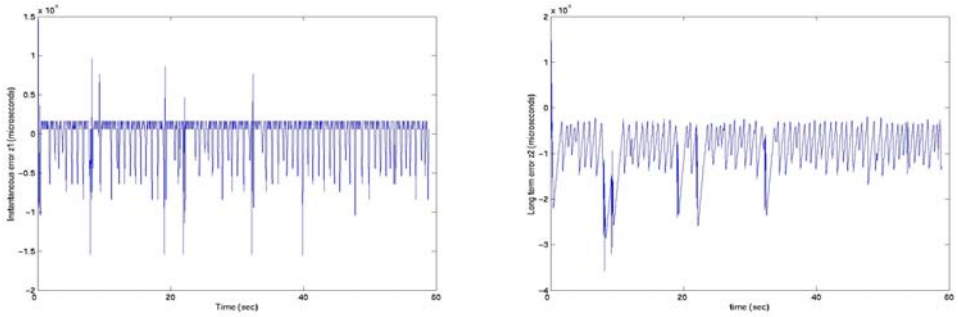

Figure 3 Plots for z^1 and z^2 vs. Time for optimizing controller

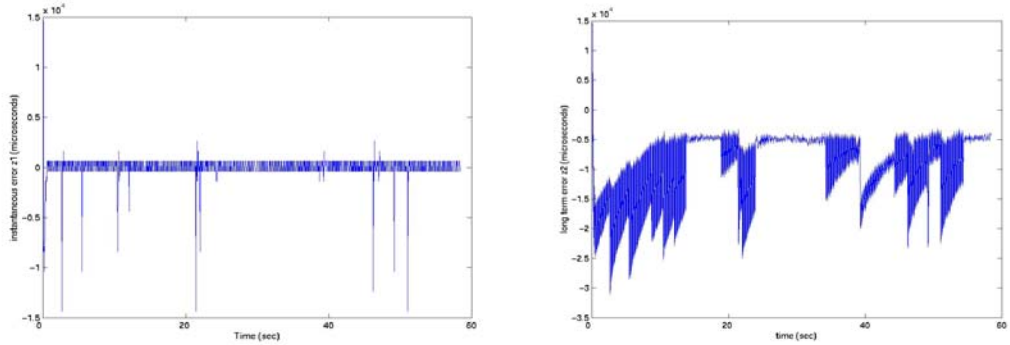

Figure 4 Plots for z^1 and z^2 vs. Time for earlier PI controller

Conclusion and future work:

This is our first attempt to apply dynamic optimization to this computer system resource allocation problem. Initial results are encouraging and more dedicated research in this area can go a long way in helping computer system designers.

Future work includes more testing of the controller on real MPEG videos. MPEG frames have time stamps that are quantized at 33 msec time intervals at each frame. The progress made by a MPEG decoder is only observed by the controller once a complete frame is decoded. This creates some sampling issues that must be addressed because our controller sampling interval is currently 16 msec. We also need to modify our design problem to avoid emptying the buffer of the next stage, in case it is a video display stage.

The related work of Luca [6] uses *a priori* information on the type of video frames to be decoded to allocate CPU for each frame in a real-time O/S. Our goal is to determine control strategies without use of a priori information.

References:

[1] David Steere, Molly H Shor, Ashvin Goel, Jonathan Walpole and Calton Pu, Control and Modeling issues in computer Operating Systems: Resource Management for Real-Rate Computer Applications, *In Proceedings of 39th IEEE Conference on Decision and Control (CDC) December 2000.*

[2] Ashvin Goel, Molly H. Shor, Jonathan Walpole, David C. Steere, and Calton Pu Using Feedback Control for a Network and CPU resource Management Application, *In Proceedings of the 2001 American Control Conference (ACC), June 2001.*

[3] Frank L. Lewis and Vassilis L. Syrmos, Optimal Control, Second Edition, Wiley-Interscience Publications, 1995.

[4] Quasars software and OGI Linux kernel: http://www.cse.ogi.edu/DISC/projects/quasar/releases/

[5] Longer version of paper: ftp://cse/ogi/edu/pub/tech-reports/2002/02-007.pdf

[6] Luca Abeni, Luigi Palopoli, Giuseppe Lipari, and Jonathan Walpole, Analysis of a Reservation-Based Feedback Scheduler. *Proceedings of the IEEE Real-Time Systems Symposium, December 2002.*

[7] Thrasyvoulos Pappas, Alan j. Laub and Nils R. Sandell Jr. On the Numerical Solution of the Discrete-Time Algebraic Riccati Equation. *IEEE Transactions on Automatic Control, Vol. AC-25, No.4, august 1980.*

[8] A. Laub, A Schur Method for Solving Algebraic Riccati Equations.*IEEE transaction on Automatic Control, AC-24 1979.*