Control theory has been used to design adaptive resource management for many classes of applications,

including power management [187], task scheduling [222], QoS adaptation in Web servers [3], and

load balancing. The classical feedback control methods are used in all these cases to regulate the key

operating parameters of the system based on measurement of the system output; the feedback control in

thesemethods assumes a linear time-invariant system model and a closed-loop controller. This controller

is based on an open-loop system transfer function that satisfies stability and sensitivity constraints.

A technique to design self-managing systems based on concepts from control theory is discussed

in [369]. The technique allows multiple QoS objectives and operating constraints to be expressed

as a cost function and can be applied to stand-alone or distributed Web servers, database servers,

high-performance application servers, and even mobile/embedded systems. The following discussion

considers a single processor serving a stream of input requests.We attempt to minimize a cost function

that reflects the response time and the power consumption. Our goal is to illustrate the methodology for

optimal resource management based on control theory concepts. The analysis is intricate and cannot be

easily extended to a collection of servers.

Control Theory Principles. We start our discussion with a brief overview of control theory principles

one could use for optimal resource allocation. Optimal control generates a sequence of control inputs

over a look-ahead horizon while estimating changes in operating conditions. A convex cost function has

arguments x(k), the state at step k, and u(k), the control vector; this cost function is minimized, subject

to the constraints imposed by the system dynamics. The discrete-time optimal control problem is to

determine the sequence of control variables u(i ), u(i + 1), . . . , u(n − 1) to minimize the expression

J (i ) =  (n, x(n)) +

 n−1

k=i

Lk (x(k), u(k)), (6.1)

where  (n, x(n)) is the cost function of the final step, n, and Lk (x(k), u(k)) is a time-varying cost

function at the intermediate step k over the horizon [i , n]. The minimization is subject to the constraints

x(k + 1) = f k (x(k), u(k)), (6.2) where x(k +1), the system state at time k +1, is a function of x(k), the state at time k, and of u(k), the

input at time k; in general, the function f k is time-varying; thus, its superscript.

One of the techniques to solve this problem is based on the Lagrange multiplier method of finding

the extremes (minima or maxima) of a function subject to constrains. More precisely, if we want to

maximize the function g(x, y) subject to the constraint h(x, y) = k, we introduce a Lagrange multiplier

λ. Then we study the function

 (x, y, λ) = g(x, y) + λ   [h(x, y) − k]. (6.3)

A necessary condition for the optimality is that (x, y, λ) is a stationary point for  (x, y, λ). In other

words,

∇x,y,λ (x, y, λ) = 0 or

∂ (x, y, λ)

∂x

,

∂ (x, y, λ)

∂ y

,

∂ (x, y, λ)

∂λ

= 0. (6.4)

The Lagrange multiplier at time step k is λk and we solve Eq. (6.4) as an unconstrained optimization

problem.We define an adjoint cost function that includes the original state constraints as the Hamiltonian

function H, then we construct the adjoint system consisting of the original state equation and the costate

equation governing the Lagrange multiplier. Thus, we define a two-point boundary problem3; the state

xk develops forward in time whereas the costate occurs backward in time.

A Model Capturing Both QoS and Energy Consumption for a Single-Server System. Now we

turn our attention to the case of a single processor serving a stream of input requests. To compute the

optimal inputs over a finite horizon, the controller in Figure 6.1 uses feedback regarding the current

state, as well as an estimation of the future disturbance due to the environment. The control task is

solved as a state regulation problem updating the initial and final states of the control horizon.

We use a simple queuing model to estimate the response time. Requests for service at processor P are

processed on a first-come, first-served (FCFS) basis.We do not assume a priori distributions of the arrival process and of the service process; instead, we use the estimate ˆ  (k) of the arrival rate  (k) at time k.

We also assume that the processor can operate at frequencies u(k) in the range u(k) ∈ [umin, umax ] and

call ˆ c(k) the time to process a request at time k when the processor operates at the highest frequency

in the range, umax . Then we define the scaling factor α(k) = u(k)/umax and we express an estimate of

the processing rate N(k) as α(k)/ ˆ c(k).

The behavior of a single processor is modeled as a nonlinear, time-varying, discrete-time state equation.

If Ts is the sampling period, defined as the time difference between two consecutive observations of

the system, e.g., the one at time (k+1) and the one at time k, then the size of the queue at time (k+1) is

q(k + 1) = max

q(k) +

ˆ  (k) − u(k)

ˆ c(k)   umax

  Ts

, 0

. (6.5)

The first term, q(k), is the size of the input queue at time k, and the second one is the difference between

the number of requests arriving during the sampling period, Ts , and those processed during the same

interval.

The response time ω(k) is the sum of the waiting time and the processing time of the requests

ω(k) = (1 + q(k))  ˆc(k). (6.6)

Indeed, the total number of requests in the system is (1 + q(k)) and the departure rate is 1/ˆ c(k).

We want to capture both the QoS and the energy consumption, since both affect the cost of providing

the service. A utility function, such as the one depicted in Figure 6.4, captures the rewards as well as the

penalties specified by the service-level agreement for the response time. In our queuing model the utility

is a function of the size of the queue; it can be expressed as a quadratic function of the response time

S(q(k)) = 1/2(s   (ω(k) − ω0)2), (6.7)

with ω0, the response time set point and q(0) = q0, the initial value of the queue length. The energy

consumption is a quadratic function of the frequency

R(u(k)) = 1/2(r   u(k)2). (6.8)

The two parameters s and r are weights for the two components of the cost, the one derived from the

utility function and the second from the energy consumption.We have to pay a penalty for the requests

left in the queue at the end of the control horizon, a quadratic function of the queue length

 (q(N)) = 1/2(v   q(n)2). (6.9)

The performance measure of interest is a cost expressed as

J =  (q(N)) +

N −1

k=1

[S(q(k)) + R(u(k))]. (6.10)

The problem is to find the optimal control u∗ and the finite time horizon [0, N] such that the trajectory

of the system subject to optimal control is q∗, and the cost J in Eq. (6.10) is minimized subject to the following constraints

q(k + 1) =

q(k) +

ˆ  (k) − u(k)

ˆ c(k)   umax

  Ts

, q(k)   0, and umin   u(k)   umax . (6.11)

When the state trajectory q( ) corresponding to the control u( ) satisfies the constraints

 1 : q(k) > 0,  2 : u(k)   umin,  3 : u(k)   umax , (6.12)

then the pair

q( ), u( )

is called a feasible state. If the pair minimizes Eq. (6.10), then the pair is

optimal.

The Hamiltonian H in our example is

H = S(q(k)) + R(u(k)) + λ(k + 1)

q(k) +

 (k) − u(k)

c   umax

Ts

+ μ1(k)   (−q(k)) + μ2(k)   (−u(k) + umin) + μ3(k)   (u(k) − umax ).

(6.13)

According to Pontryagin’s minimum principle,4 the necessary condition for a sequence of feasible

pairs to be optimal pairs is the existence of a sequence of costates λ and a Lagrange multiplier

μ = [μ1(k),μ2(k),μ3(k)] such that

H(k, q∗

, u∗

, λ

∗

,μ

∗

)   H(k, q, u∗

, λ

∗

,μ

∗

), ∀q   0 (6.14)

where the Lagrange multipliers, μ1(k),μ2(k),μ3(k), reflect the sensitivity of the cost function to the

queue length at time k and the boundary constraints and satisfy several conditions

μ1(k)   0, μ1(k)(−q(k)) = 0, (6.15)

μ2(k)   0, μ2(k)(−u(k) + umin) = 0, (6.16)

μ3(k)   0, μ3(k)(u(k) − umax ) = 0. (6.17)

A detailed analysis of the methods to solve this problem and the analysis of the stability conditions is

beyond the scope of our discussion and can be found in [369].

The extension of the techniques for optimal resource management from a single system to a cloud

with a very large number of servers is a rather challenging area of research. The problem is even harder

when, instead of transaction-based processing, the cloud applications require the implementation of a

complex workflow.