A utility function relates the “benefits” of an activity or service with the “cost” to provide the service.

For example, the benefit could be revenue and the cost could be the power consumption.

A service-level agreement (SLA) often specifies the rewards as well as the penalties associated with

specific performance metrics. Sometimes the quality of services translates into average response time;

this is the case of cloud-based Web services when the SLA often explicitly specifies this requirement.

For example, Figure 6.4 shows the case when the performance metrics is R, the response time. The

largest reward can be obtained when R   R0; a slightly lower reward corresponds to R0 < R   R1.

When R1 < R   R2, instead of gaining a reward, the provider of service pays a small penalty; the

penalty increases when R > R2. A utility function, U(R), which captures this behavior, is a sequence

of step functions. The utility function is sometimes approximated by a quadratic curve, as we shall see

in Section 6.2.

In this section we discuss a utility-based approach for autonomic management. The goal is to maximize

the total profit computed as the difference between the revenue guaranteed by an SLA and the

total cost to provide the services. Formulated as an optimization problem, the solution discussed in [9]

addresses multiple policies, including QoS. The cloud model for this optimization is quite complex and

requires a fair number of parameters.

We assume a cloud providing |K| different classes of service, each class k involving Nk applications.

For each class k ∈ K call vk the revenue (or the penalty) associated with a response time rk and assume

a linear dependency for this utility function of the form vk = vmax

k

1 − rk/rmax

k

, see Figure 6.5(a).

Call mk = −vmax

k /rmax

k the slope of the utility function.

The system is modeled as a network of queues with multiqueues for each server and with a delay

center that models the think time of the user after the completion of service at one server and the

start of processing at the next server [see Figure 6.5(b)]. Upon completion, a class k request either

completes with probability (1−

k ∈K πk,k  ) or returns to the system as a class k  request with transition

probability πk,k . Call λk the external arrival rate of class k requests and  k the aggregate rate for class

k, where k = λk +

k ∈K  k πk,k  .

Typically, CPU and memory are considered representative for resource allocation; for simplicity we

assume a single CPU that runs at a discrete set of clock frequencies and a discrete set of supply voltages

according to a Dynamic Voltage and Frequency Scaling (DVFS) model. The power consumption of

a server is a function of the clock frequency. The scheduling of a server is work-conserving6 and is

modeled as a Generalized Processor Sharing (GPS) scheduling [385]. Analytical models [4,280] are

too complex for large systems.

The optimization problem formulated in [9] involves five terms: A and B reflect revenues; C the cost

of servers in a low-power, stand-by mode; D the cost of active servers, given their operating frequency;

E, the cost of switching servers from low-power, stand-by mode to active state, and F, the cost of

migrating VMs from one server to another. There are nine constraints  1,  2, . . . ,  9 for this mixed integer,

nonlinear programming problem. The decision variables for this optimization problem are listed

in Table 6.2 and the parameters used are shown in Table 6.3.

The expression to be maximized is:

(A + B) − (C + D + E + F) (6.21)

with

A = max

k∈K

⎛

⎝−mk

i∈I , j∈Nk

λi ,k, j

h∈Hi

Ci ,h   yi ,h

μk, j   φi ,k, j − λi ,k, j

⎞

⎠,

B =

k∈K

uk    k , (6.22)

C =

i∈I

  ci , D =

i∈I ,h∈Hi

ci ,h   yi ,h, E =

i∈I

csi max (0, xi −  xi ), (6.23)

and

F =

i∈I ,k∈K, j∈Nj

cm max (0, zi , j ,k −  zi , j ,k ). (6.24)

The nine constraints are:

( 1)

i∈I λi ,k, j =  k , ∀k ∈ K, j ∈ Nk ⇒ the traffic assigned to all servers for class k requests

equals the predicted load for the class.

( 2)

k∈K, j∈Nk φi ,k, j   1 ∀i ∈ I ⇒server i cannot be allocated an workload more than its capacity.

( 3)

h∈Hi yi ,h = xi , ∀i ∈ I ⇒ if server i ∈ I is active it runs at one frequency in the set Hi , and

only one yi ,h is nonzero.

( 4) zi ,k, j   xi , ∀i ∈ I , k ∈ K, j ∈ Nk ⇒requests can only be assigned to active servers.

( 5) λi ,k, j    k   zi ,k, j , ∀i ∈ I , k ∈ K, j ∈ Nk ⇒ requests may run on server i ∈ I only if the

corresponding application tier has been assigned to server i .

( 6) λi ,k, j

h∈Hi Ci ,h   yi ,h

μk, j   φi ,k, j , ∀i ∈ I , k ∈ K, j ∈ Nk ⇒ resources cannot be

saturated.

( 7) RAMk, j   zi ,k, j   RAMi , ∀i ∈ I , k ∈ K ⇒ the memory on server i is sufficient to support

all applications running on it.

( 8)

Nk

j=1

1 −

M

i=1(1 − awi ,k

i

  Ak , ∀k ∈ K ⇒ the availability of all servers assigned to class k

request should be at least equal to the minimum required by the SLA.

( 9)

Nk

j=1 zi ,k, j   Nk   wi ,k , ∀i ∈ I , k ∈ K

λi , j ,k, φi , j ,k   0, ∀i ∈ I , k ∈ K, j ∈ Nk

xi , yi ,h, zi ,k, j,wi ,k ∈ {0, 1}, ∀i ∈ I , k ∈ K, j ∈ Nk ⇒ constraints and relations among

decision variables.

Clearly, this approach is not scalable to clouds with a very large number of servers. Moreover, the

large number of decision variables and parameters of the model make this approach infeasible for a

realistic cloud computing resource management strategy.