Scheduling is a critical component of cloud resource management. Scheduling is responsible for resource

sharing/multiplexing at several levels. A server can be shared among several virtual machines, each virtual

machine could support several applications, and each application may consist of multiple threads.

CPU scheduling supports the virtualization of a processor, the individual threads acting as virtual processors;

a communication link can be multiplexed among a number of virtual channels, one for each flow.

In addition to the requirement to meet its design objectives, a scheduling algorithm should be efficient,

fair, and starvation-free. The objectives of a scheduler for a batch system are to maximize the throughput

(the number of jobs completed in one unit of time, e.g., in one hour) and to minimize the turnaround

time (the time between job submission and its completion). For a real-time system the objectives are to

meet the deadlines and to be predictable. Schedulers for systems supporting a mix of tasks – some with

hard real-time constraints, others with soft, or no timing constraints – are often subject to contradictory

requirements. Some schedulers are preemptive, allowing a high-priority task to interrupt the execution

of a lower-priority one; others are nonpreemptive.

Two distinct dimensions of resource management must be addressed by a scheduling policy: (a) the

amount or quantity of resources allocated and (b) the timing when access to resources is granted.

Figure 6.7 identifies several broad classes of resource allocation requirements in the space defined by

these two dimensions: best-effort, soft requirements, and hard requirements. Hard-real time systems are

the most challenging because they require strict timing and precise amounts of resources.

There are multiple definitions of a fair scheduling algorithm. First, we discuss the max-min fairness

criterion [128]. Consider a resource with bandwidth B shared among n users who have equal rights.

Each user requests an amount bi and receives Bi . Then, according to the max-min criterion, the following

conditions must be satisfied by a fair allocation:

C1. The amount received by any user is not larger than the amount requested, Bi   bi .

C2. If the minimum allocation of any user is Bmin no allocation satisfying condition C1 has a higher

Bmin than the current allocation.

C3. When we remove the user receiving the minimum allocation Bmin and then reduce the total amount

of the resource available from B to (B − Bmin), the condition C2 remains recursively true.

A fairness criterion for CPU scheduling [142] requires that the amount of work in the time interval

from t1 to t2 of two runnable threads a and b,  a(t1, t2) and  b(t1, t2), respectively, minimize the

expression

 a(t1, t2)

wa

−  b(t1, t2)

wb

, (6.27)

where wa and wb are the weights of the threads a and b, respectively.

The quality-of-service (QoS) requirements differ for different classes of cloud applications and

demand different scheduling policies. Best-effort applications such as batch applications and analytics7

do not require QoS guarantees. Multimedia applications such as audio and video streaming have soft

real-time constraints and require statistically guaranteed maximum delay and throughput. Applications

with hard real-time constraints do not use a public cloud at this time but may do so in the future.

Round-robin, FCFS, shortest-job-first (SJF), and priority algorithms are among the most common

scheduling algorithms for best-effort applications. Each thread is given control of the CPU for a definite

period of time, called a time-slice, in a circular fashion in the case of round-robin scheduling. The

algorithm is fair and starvation-free. The threads are allowed to use the CPU in the order in which they

arrive in the case of the FCFS algorithms and in the order of their running time in the case of SJF

algorithms. Earliest deadline first (EDF) and rate monotonic algorithms (RMA) are used for real-time

applications. Integration of scheduling for the three classes of application is discussed in [56], and two new algorithms for integrated scheduling, resource allocation/dispatching (RAD) and rate-based earliest

deadline (RBED) are proposed.

Next we discuss several algorithms of special interest for computer clouds. These algorithms illustrate

the evolution in thinking regarding the fairness of scheduling and the need to accommodate multiobjective

scheduling – in particular, scheduling for multimedia applications.