Often, an SLA specifies the time when the results of computations done on the cloud should be available.

This motivates us to examine cloud scheduling subject to deadlines, a topic drawing on a vast body of

literature devoted to real-time applications.

Task Characterization and Deadlines. Real-time applications involve periodic or aperiodic tasks

with deadlines. A task is characterized by a tuple (Ai, σi , Di ), where Ai is the arrival time, σi > 0 is the

data size of the task, and Di is the relative deadline. Instances of a periodic task,

q

i , with period q are

identical,

q

i

≡

q , and arrive at times A0, A1, . . . Ai , . . . , with Ai+1 − Ai = q. The deadlines satisfy

the constraint Di   Ai+1 and generally the data size is the same, σi = σ. The individual instances of

aperiodic tasks,

i , are different. Their arrival times Ai are generally uncorrelated, and the amount of

data σi is different for different instances. The absolute deadline for the aperiodic task

i is (Ai + Di ).

We distinguish hard deadlines from soft deadlines. In the first case, if the task is not completed by the

deadline, other tasks that depend on it may be affected and there are penalties; a hard deadline is strict

and expressed precisely as milliseconds or possibly seconds. Soft deadlines play more of a guideline

role and, in general, there are no penalties. Soft deadlines can be missed by fractions of the units used to

express them, e.g., minutes if the deadline is expressed in hours, or hours if the deadlines is expressed

in days. The scheduling of tasks on a cloud is generally subject to soft deadlines, though occasionally

applications with hard deadlines may be encountered.

SystemModel. In our discussion we consider only aperiodic tasks with arbitrarily divisible workloads.

The application runs on a partition of a cloud, a virtual cloud with a head node called S0 and n worker

nodes S1, S2, . . . , Sn. The system is homogeneous, all workers are identical, and the communication

time from the head node to any worker node is the same. The head node distributes the workload to

worker nodes, and this distribution is done sequentially. In this context there are two important problems:

1. The order of execution of the tasks

i .

2. The workload partitioning and the task mapping to worker nodes.

Scheduling Policies. The most common scheduling policies used to determine the order of execution

of the tasks are:

• First in, first out (FIFO). The tasks are scheduled for execution in the order of their arrival.

• Earliest deadline first (EDF). The task with the earliest deadline is scheduled first.

• Maximum workload derivative first (MWF).

The workload derivative DCi (nmin) of a task

i when nmin nodes are assigned to the application,

is defined as

DCi (nmin) = Wi

nmin

i

+ 1

− Wi

nmin

i

, (6.69)

with Wi (n) the workload allocated to task

i when n nodes of the cloud are available; if E(σi , n) is the

execution time of the task, then Wi (n) = n   E(σi , n). The MWF policy requires that:

1. The tasks are scheduled in the order of their derivatives, the one with the highest derivative DCi first.

2. The number n of nodes assigned to the application is kept to a minimum, nmin

i .

We discuss two workload partitioning and task mappings to worker nodes, optimal and the equal

partitioning.

Optimal Partitioning Rule (OPR). The optimality in OPR refers to the execution time; in this case,

the workload is partitioned to ensure the earliest possible completion time, and all tasks are required to complete at the same time. EPR, as the name suggests, means that the workload is partitioned in equal

segments. In our discussion we use the derivations and some of the notations in [218]; these notations

are summarized in Table 6.7.

The timing diagram in Figure 6.13 allows us to determine the execution time E(n, σ) for the OPR as

E(n, σ) =  1 +  1

=  1 +  2 +  2

=  1 +  2 +  3 +  3

...

=  1 +  2 +  3 +     + n +  n.

(6.70)

We substitute the expressions of  i, i , 1   i   n, and rewrite these equations as

E(n, σ) = α1   σ   τ + α1   σ   ρ

= α1   σ   τ + α2   σ   τ + α2   σ   ρ

= α1   σ   τ + α2   σ   τ + α3   σ   τ + α3   σ   ρ

=

...

= α1   σ   τ + α2   σ   τ + α3   σ   τ +     +αn   σ   τ + αn   σ   ρ.

From the first two equations we find the relation between α1 and α2 as

α1 = α2

β

with β = ρ

τ + ρ

, 0   β   1. (6.72)

This implies that α2 = β   α1. It is easy to see that in the general case

αi = β   αi−1 = βi−1   α1. (6.73)

But αi are the components of the load distribution vector; thus,

 n

i=1

αi = 1. (6.74)

Next, we substitute the values of αi and obtain the expression for α1:

α1 + β   α1 + β2   α1 + β3   α1 . . . βn−1   α1 =1 or α1 = 1 − β

1 − βn . (6.75)

We have now determined the load distribution vector and we can now determine the execution time as

E(n, σ) = α1   σ   τ + α1   σ   ρ = 1 − β

1 − βn σ(τ + ρ). (6.76)

Call CA

(n) the completion time of an application A = (A, σ, D), which starts processing at time t0

and runs on n worker nodes; then

CA

(n) = t0 + E(n, σ) = t0 + 1 − β

1 − βn σ(τ + ρ). (6.77)

The application meets its deadline if and only if

CA

(n)   A + D, (6.78)

or

t0 + E(n, σ) = t0 + 1 − β

1 − βn σ(τ + ρ)   A + D. (6.79)

But 0 < β < 1 thus, 1 − βn > 0, and it follows that

(1 − β)σ(τ + ρ)   (1 − βn)(A + D − t0). (6.80)

The application can meet its deadline only if (A+ D −t0) > 0, and under this condition this inequality

becomes

βn   γ with γ = 1 − σ   τ

A + D − t0

. (6.81)

If γ   0, there is not enough time even for data distribution and the application should be rejected.

Whenγ > 0, then n   ln γ

ln β . Thus, the minimum number of nodes for the OPR strategy is

nmin =

ln γ

ln β

 . (6.82)

EqualPartitioning Rule (EPR). EPRassigns an equalworkload to individualworker nodes,αi = 1/n.

From the diagram in Figure 6.14 we see that

E(n, σ) =

 n

i=1

 i +  n = n   σ

n

  τ + σ

n

  ρ = σ   τ + σ

n

  ρ. (6.83)

The condition for meeting the deadline, CA

(n)   A + D, leads to

t0 + σ   τ + σ

n

  ρ   A + D or n   σ   ρ

A + D − t0 − σ   τ

. (6.84)

Thus,

nmin =

σ   ρ

A + D − t0 − σ   τ

. (6.85)

The pseudocode for a general schedulability test for FIFO, EDF, and MWF scheduling policies, for

two-node allocation policies, MN (minimum number of nodes) and AN (all nodes), and for OPR and

EPR partitioning rules is given in reference [218]. The same paper reports on a simulation study for 10

algorithms. The generic format of the names of the algorithms is Sp-No-Pa, with Sp=FIFO/EDF/MWF,

No = MN/AN, and Pa = OPR/EPR. For example, MWF-MN-OPR uses MWF scheduling, minimum

number of nodes, and OPR partitioning. The relative performance of the algorithms depends on the

relations between the unit cost of communication τ and the unit cost of computing ρ.