Indian Institute of Information Technology Vadodara

Cloud computing

Formulation on Start and deadline with priority in scheduling

Mentor: Dr.Reshmi Mitra

Team Members: K. Naveen kumar & K. yeshwanth naik

Date: 13/5/2017

Table of contents:

Formulations for the start and stop time, Dead Line:

Given: For Each cloudlet characteristics

start time: x
stop time: y
Deadline: z

case1: No preemption and no priority and **space shared policy .**
suppose we have cloudlets T1, T2, T3 ....Tn with the above characteristics

considering two cloudlets T1, T2 with 1 virtualmachine 1 host 1 datacenter

for T1 cloudlet characteristics:

start time: x1
stop time: y1
Deadline: z1

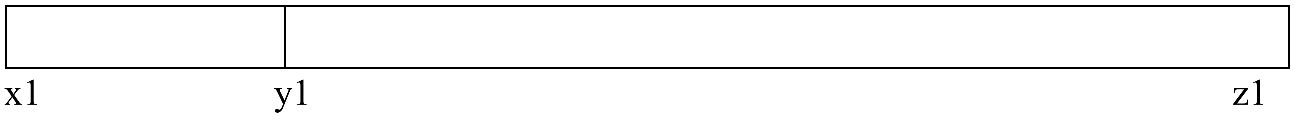
for T2 cloudlet characteristics:

start time: x2
stop time: y2
Deadline: z2

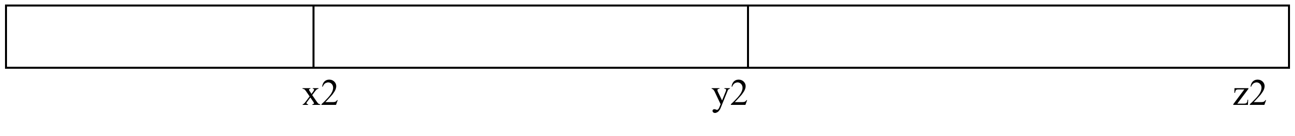
conditions applicable:

x1 < y1 & x2 < y2 and y1< x2 ,y1<y2, x1 < x2 and y1 >= z1 y2 >= z2 z1 less than or greater than or equal to z2

for T1



for T2



formulations:

Execution time of E1 for Task T1 : y1-x1
Execution time of E2 for Task T2: y2-x2

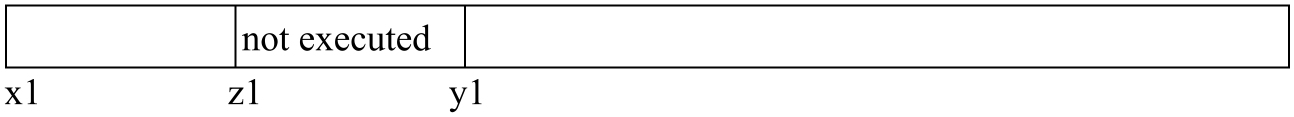
Total execution time TE : E1 + E2

generalizing total execution time with no priority no preemption in space shared policy provided z1 and z2 are greater than or equal to y1 and y2.

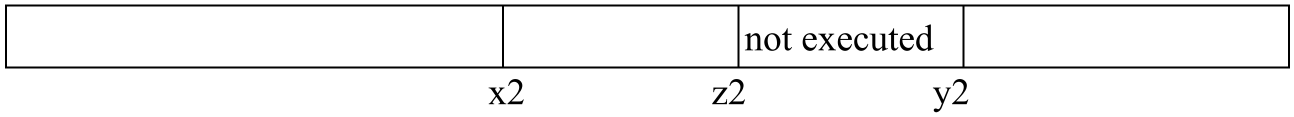
$$TE = \Sigma_{i=1}^{n} E_n$$

for z1<y1 and z2<y2

for T1



for T2



formulations:

Execution time of E1 for Task T1 : z1-x1
Execution time of E2 for Task T2: z2-x2

Total execution time TE : E1 + E2

generalizing total execution time with nopriority nopreemption in space shared policy provided z1 and z2 are greater than or qual to y1 and y2.

$$TE = \Sigma_{i=1}^{n} E_n$$

Given: For Each cloud let characteristics

start time: x
stop time: y
Deadline: z

case2: No preemption and no priority and **Time shared policy .**
suppose we have cloud-lets T1, T2, T3 ....Tn with the above characteristics

considering two cloud-lets T1, T2 1 virtual machine 1 host 1 data center

for T1 cloud-let characteristics:

start time: x1
stop time: y1
Deadline: z1

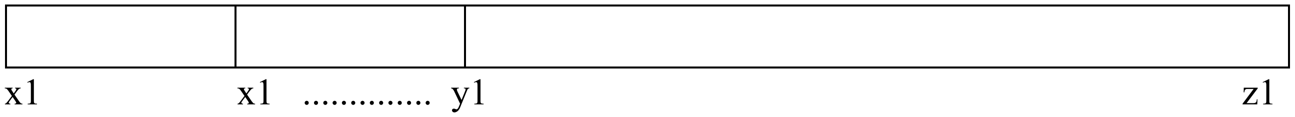
for T2 cloud-let characteristics:

start time: x2
stop time: y2
Deadline: z2

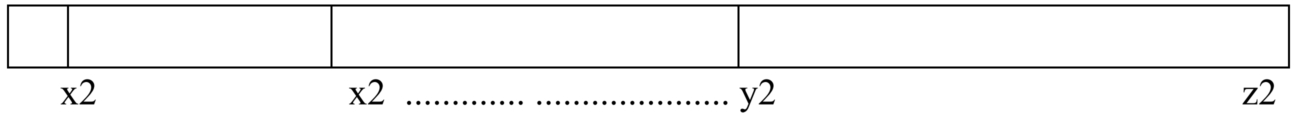
conditions applicable:

x1 < y1 & x2 < y2 and concurrent cases and y1 >= z1 y2 >= z2 z1 less than or greater than or equal to z2

for T1



for T2



formulations:

Execution time of E1 for Task T1 : y1-x1
Execution time of E2 for Task T2: y2-x2

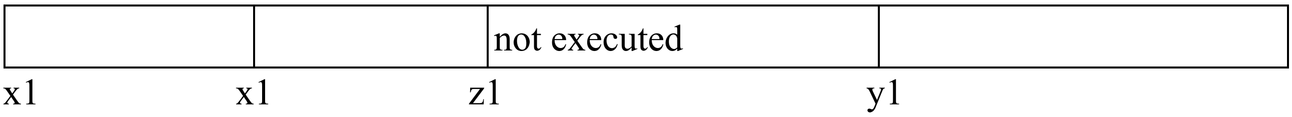
Total execution time TE : max(E1 , E2)

generalizing total execution time with no priority no preemption in space shared policy provided z1 and z2 are greater than or equal to y1 and y2.

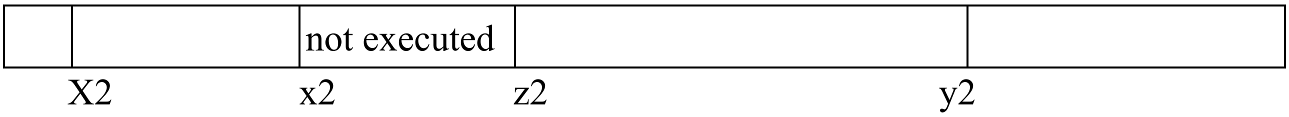
$$TE = \max(E_{1,}E_{2,}......E_{n})$$

for z1<y1 and z2<y2

for T1



for T2



formulations:

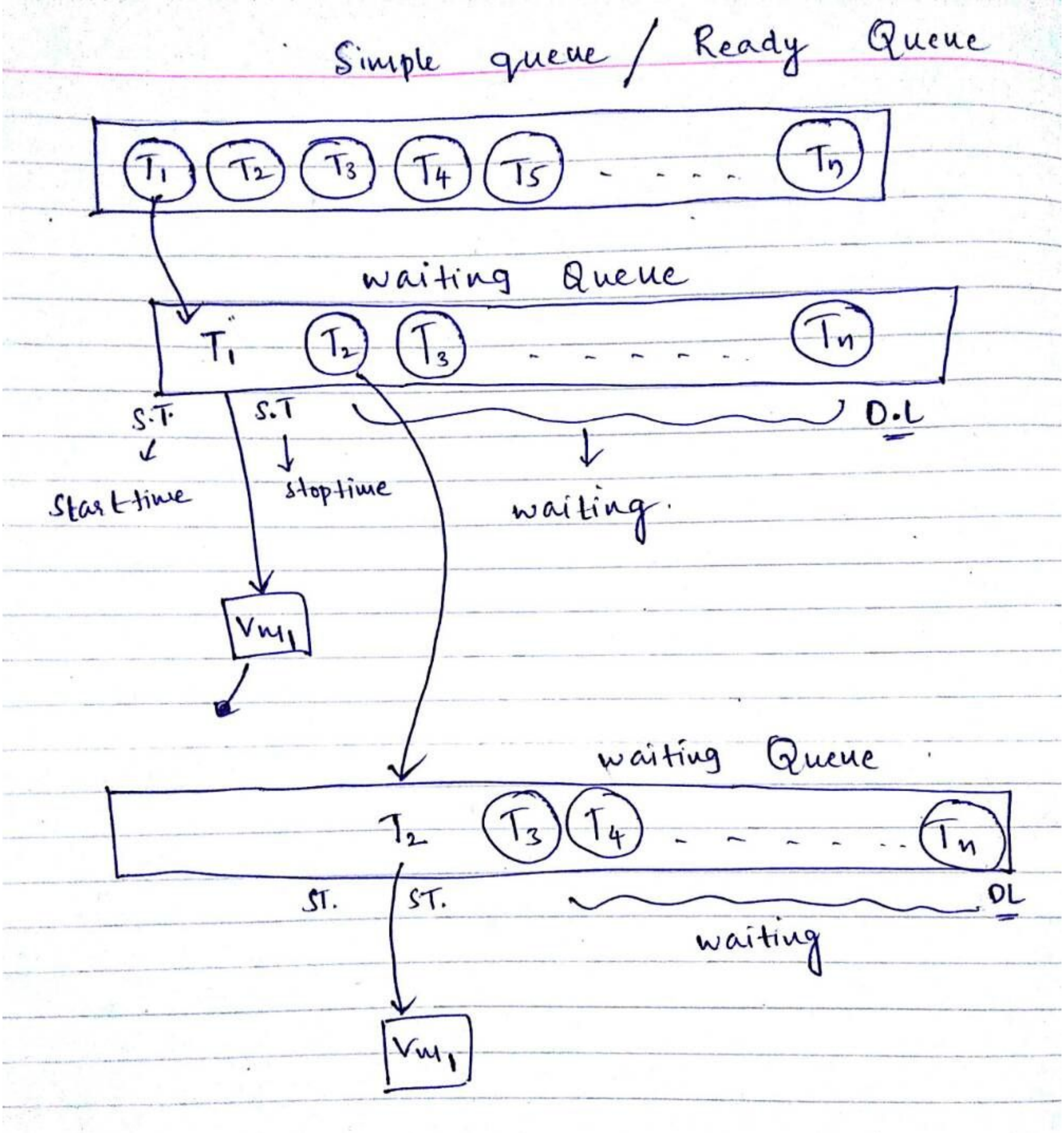
Execution time of E1 for Task T1 : z1-x1
Execution time of E2 for Task T2: z2-x2

Total execution time TE : max(E1 , E2)

generalizing total execution time with no priority no preemption in Time shared policy provided z1 and z2 are greater than or equal to y1 and y2.

$$TE = \max (E_{1,}E_{2,}......E_{n})$$

Design state machines and algorithms for implementation for space shared.

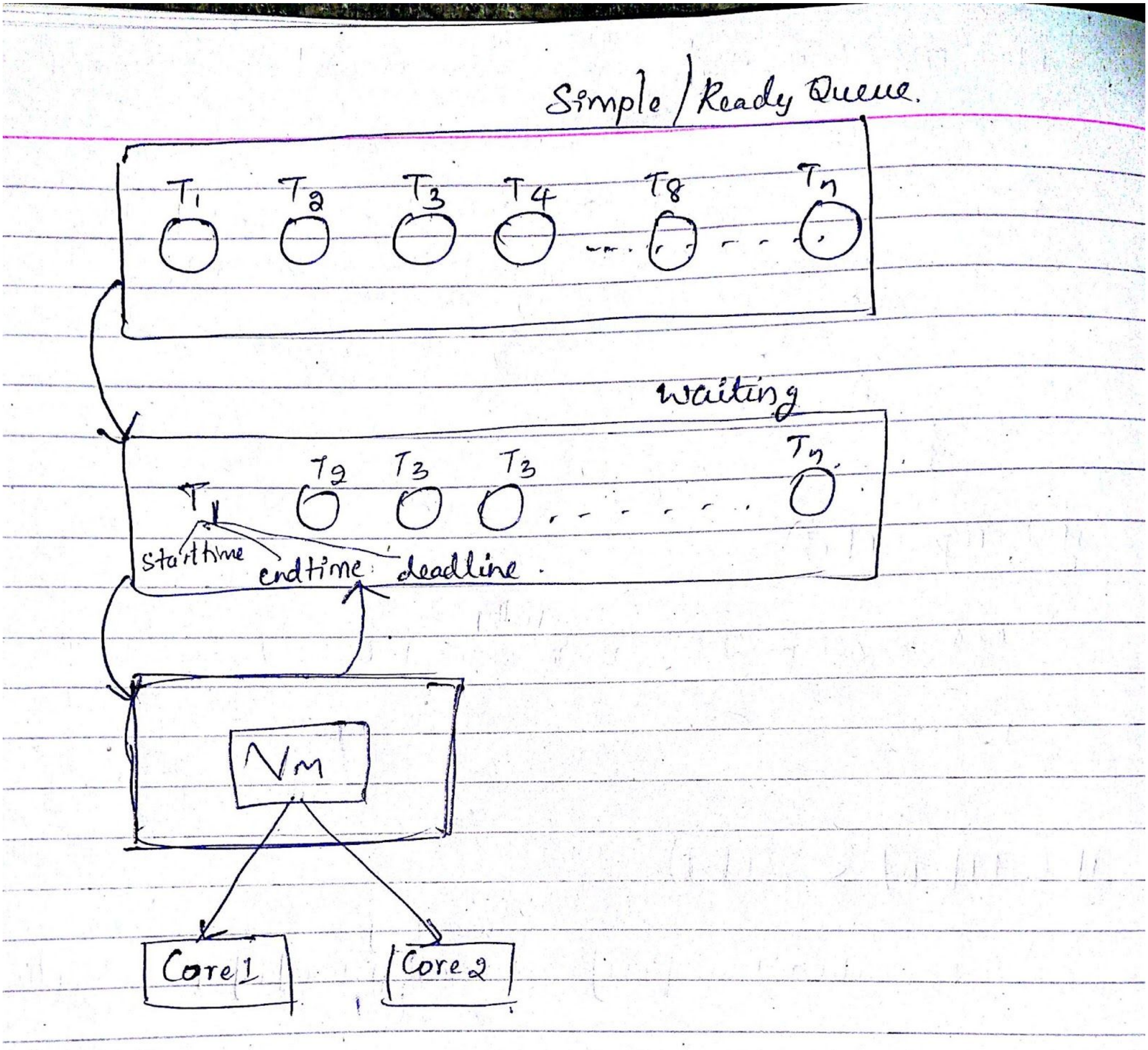


In Space shared cloudlets are executed in a sequential manner,after the completion of first task,second task enters the virtual machine.Now, there are cloud-lets first they are in simple queue/ Ready queue.
Eg: from the above figure there are n Tasks in a simple queue the starting time of T1<T2<T3……..<Tn. Next they enter into waiting queue first T1 gets executed using a virtual machine Remaining tasks will be in the waiting queue until the task T1 is executed. After T1 gets executed T2 gets into virtual machine to get executed and Remaining Tasks are in the waiting queue. Then so on….

Algorithm for space shared policy:

```
import java.io.*;
class fcfs
{

public static void main(String args[]) throws Exception
{
int n,ST[],WT[],WQ[],DL[],STOP TIME[],Execution Time[];
float AWT=0;


System.out.println("Enter number of cloudlets "+(i+1));
n[i]=Integer.parseInt(br.readLine());

if (DL[i] >= ST[i] ){

Execution Time[i]= STOP TIME[i] - ST[i];

}
else
{
        Execution Time[i]=DL[i] - ST[i];
}

InputStreamReader isr=new InputStreamReader(System.in);
BufferedReader br=new BufferedReader(isr);
System.out.println("Enter no of cloudlets");
n=Integer.parseInt(br.readLine());
WT=new int[n];
WQ=new int[n];
STOP TIME=new int[n];
DL=new int[n];
St=new int[n]
System.out.println(“Enter start time for each cloudlets
\n******************************”);
for(int i=0;i<n;i++)
{
System.out.println(“Enter stop time for process “+(i+1));
STOP TIME[i]=Integer.parseInt(br.readLine());
}
System.out.println(“***********************************************”);
for(int i=0;i<n;i++)
{
System.out.println(“Enter Dead line for process”+i);
DL[i]=Integer.parseInt(br.readLine());
}
System.out.println(“***********************************************”);
WT[0]=0;
for(int i=1;i<n;i++)
{
WT[i]=WT[i-1]+Execution Time[i-1];
WT[i]=WT[i]-Execution Time[i];
}

System.out.println(”  PROCESS   BT      WT      TAT     “);
for(int i=0;i<n;i++)
{
System.out.println(”    “+ i + ”       “+WT[i]+”       “+ST[i]+”       “+STOP TIME[i]);
}
AWT=AWT/n;
System.out.println(“***********************************************”);
System.out.println(“Avg waiting
time=”+AWT+”\n***********************************************”);
}
}
```

Design state machine and algorithm for Time shared:



In Time shared cloudlets are executed concurrently,as an when they arrive.
Eg: from the above figure there are n Tasks in a simple queue the starting time of T1<T2<T3……..<Tn. Next they enter in to waiting queue first T1 gets executed using a virtual machine for sometime and then task T2 enters the virtual machine leaving T1 back into waiting queue. Then for some time T2 gets executed and then T3 enters leaving T2 into waiting queue and so

Algorithm for Time shared policy:

First, all the cloudlets are sorted in ascending way based on the tasks start time (the time needed for execution) and sent to the
waiting list

```
1. nt number of tasks(cloudlets)
2. i counter
```

Second, if the tasks count on the waiting list is equal or less than n(number of tasks), send the waiting list tasks to the ready list

```
3. While (WQ != NULL and nt<= n){
4. RQ  WQ

5. For i=1 to nt
6. {
7. TiTi
8. Send task Ti to finish list
9. Ti +1
//WQ: Waiting Queue List
//RQ: Ready Queue List
// TQ: Time Quantum
// Ti: Execution Time of Task i
}
}
```

Third, if the tasks count is greater than n, find duplicates and send them to duplicate list

```
10. While (WQ != NULL and nt>n){
11. If(DL!=NULL)
//DL: Duplicates List
12. Send Tasks from Waiting List to Ready List
13. RQ  WQ
14. TQ  Tn-1
//n: Task counter on the Duplicates List

15. For i=1 to nt
{
If TQ >= Ti
TiTi
Send task Ti to Finish list
Ti +1
Else
Ti TQ
Send task Ti to Waiting list
Ti +1
}
16. Go to First step
```

Fourth, if the tasks count on the waiting list is a greater than n send tasks on waiting list to ready list

```
17. RQ  WQ
18. TQ  ∑Ti / nt
19. For i=1 to nt
20. {
If TQ >= Ti
TiTi
Send task Ti to Finish list
ii +1
Else
Ti TQ
Send task Ti to Waiting list
ii +1
}
}
21. Go to First step
```

Fifth, if the tasks count on the waiting list is greater than n, send tasks on waiting list to ready list

```
22. RQ  WQ
23. TQ  T(nt+1 / 2) / 2)
24. For i=1 to nt
25. {
26. If TQ >= Ti
27. TiTi
Send task Ti to Finish list
ii +1
28. Else
Ti TQ
Send task Ti to Waiting list
i. ii +1
}
29. Go to First step
}
```

Ready-queue, waiting queue and job queue:

job queue: It selects processes from the queue and loads them into memory for execution.

Ready-queue: The Ready queue is a queue of all processes that are waiting to be scheduled

waiting-queue: This queue is used for waiting the other tasks until the first tasks completes

All the data structures that is used for the above three queue is queue data structure

Pre-emption:

In cloud computing, **pre-emption** is the act of temporarily interrupting a task being carried out by a cloud system, without requiring its cooperation, and with the intention of resuming the task at a later time. Such kind of changes of the executed task are known as context switches.

priority:

priority is basically a user defined task, normally it can be based on characteristics of the task(cloudlet) i.e. length of the task, or file size or id.

these are predefined things which has to be done by the user, this is the static way in case of dynamic there are many problems like there may be 10 cloudlets varying each of same sizes and length in that case priority can’t be decided it can be based on the id but we may not ensure that id based would give more priority than static method.

Exceptions and Limitations:

If the waiting queue size is less than the Ready queue given task size then the queue only occupies limited tasks after completion of the tasks, the rest of the tasks will be sent to the waiting queue.

The start time should be always less than deadline time then only the execution is possible fully and stop time should also be less than the deadline then only the full execution is possible in case:

if Deadline < stop time then partial execution is possible and that task is performed once again for completion of the balanced task but it takes a huge time.

In case if Deadline = start time it never execute the task

References:

https://cs.nyu.edu/courses/spring04/V22.0202-003/lecture-04.html
http://www.ijset.net/journal/655.pdf

references from the handouts and prashanth sir report and presentation
Reference from Mastering cloud computing book by author rajkumar buyya