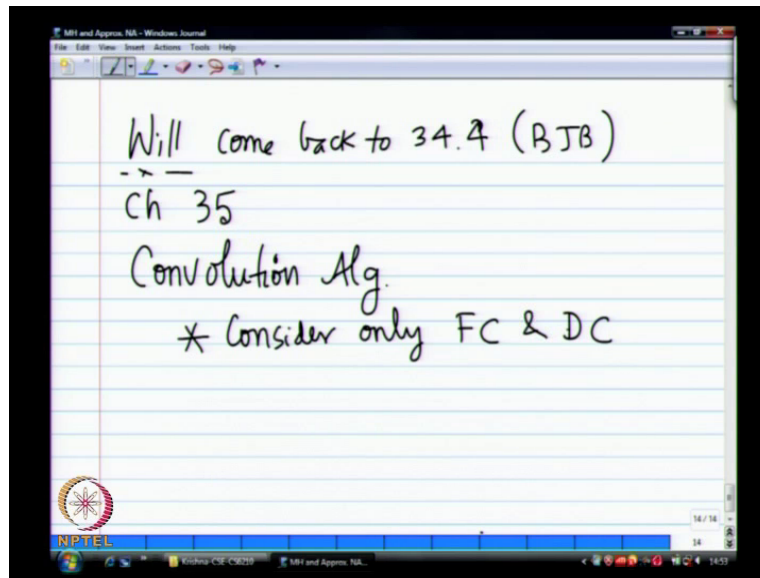


**Performance Evaluation of Computer Systems**  
**Prof. Krishna Moorthy Sivalingam**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Madras**

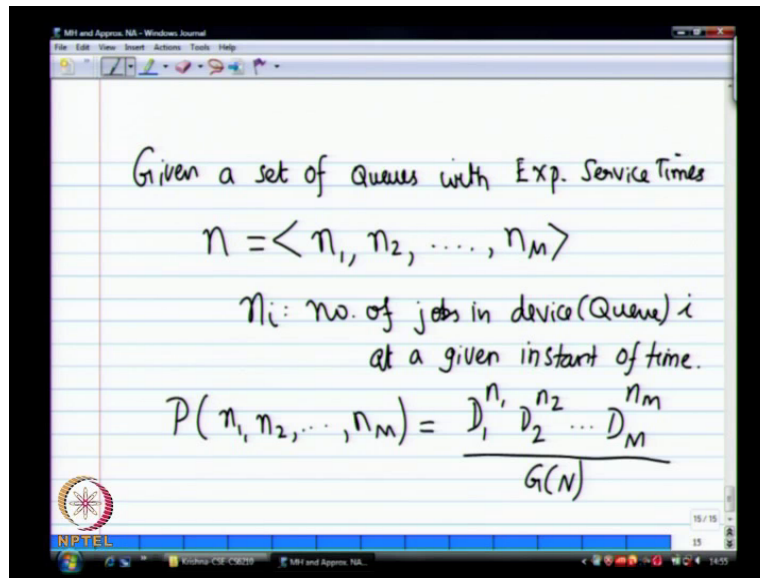
**Lecture No. # 28**  
**Convolution algorithm-I**

(Refer Slide Time: 00:10)



So, now we move to 35. So, here we will look for something; actually quite interesting in terms of the kind of result that we can get with this algorithm. So, again we will only consider initially fixed capacity service centers, and delay centers. So, this algorithm is now considers right now only these two right. So, basically a single server queues or infinite server queues 1 or infinity nothing else in between. So, this result goes back to 19 goes back to 1967, (( )) theorem got of that distribution of a number of jobs in a queuing network.

(Refer Slide Time: 01:33)



So, the theorem basically states as follows right. So, given again a set of queues right, given a with exponential service times etcetera. Then, let us let  $n$  denote this vector right. So, this denotes  $n$  denotes the number of at a given instant of time; there are  $n_1$  customers in queue 1 or  $n_1$  jobs queue 1,  $n_2$  jobs in queue 2, and  $n_m$  jobs in queue  $i$  right; this the number of. So,  $n_i$  is the number of jobs in device or queue  $i$  right, at some instant of time.

(No audio from 02:30 to 02:46)

So, then the probability that. So, what is the probability that for some given  $n$  right, and given system; this is turns out to be simply. So, it is proportional to the it is equal to the product of each demand raise to the corresponding value of  $n$ .

So,  $D_1$  raise to the power  $n_1$ ,  $D_2$  power  $n_2$ , and so on,  $D_M$  to the power  $n_m$ . (( )) proving that the books keep it up proof I went and looked at, we can go back look at the original paper for the proof, but right. So, you only look at the results yeah, because it is convenient. So now, I have. So now, in the case of  $m = 1$  right, what is the state of the system? What did you model a state of the system. Simply the number of customers, and once I know my  $p_i$ 's right, I could compute everything else based on the (( )) law, and so on. So, likewise now I have this probability with check and now compute (()), many other things in the system right. So, this is the probability of having at a given point in time, these many customers each

of these queues. And it just proportional to  $D$ , which is  $D$  again is equal to  $V$  into  $S$  right;  $D$  equals  $V$  into  $S$ ; that is what we saw define in last time.

$D$  is demand time.

$D$  is the demand yeah.

(Refer Slide Time: 04:28)

Where 
$$N = \sum_{i=1}^M n_i$$

$$D_i = \text{demand made on device } i$$

$$= V_i S_i$$

$$G(N) = \text{Normalizing Constant}$$

$$= \sum_n \frac{1}{n!} \prod_{i=1}^M D_i^{n_i}$$

So, we will write this first at put something  $N$  right. So,  $N$  is simply the total number of customers in the system like we always had, and then  $D_i$  is the demand made on device  $i$ , which we have defined so far as  $V_i$ , and  $S_i$ . If you look at other definitions they will use see that  $V_i$  as visit ratio right, divided by  $\mu_i$ , because its exponential  $1$  over  $\mu_i$  is your service time right, so we simply taken.

**(())** like a like an  $M/M/1$  queue,  $D$  is almost equal to the  $\rho$ , because...

There is also one normalizing time. So, there if you look at it will be simply  $1 - \rho$  into  $\rho$  to the  $n$  right.

**(())**

So, where  $\rho$  is basically  $\lambda$  by  $\mu$  right.

So, and but here that there lambda is the arrival right, here that lambda is sort of like your visit count to the system right. If you are the tantrum queue we saw that its simply product of right of the individual queues; there something similar to that. And then there is a normalizing constant, because of the fact that D and n D can be very large right; D can be any number, it is not constrain by when the case of M M 1, we said the rho is less than 1. So, we know that we can always compute that right; whereas, D is very large to compute this becomes a problem right. If D is 1000, 10000 to the power n is also very large, you start to looking at very large numbers right.

Then your system starts running starts running into either overflow or underflow errors if, D is very small you handle to underflow errors; if D is very large you handle to overflow errors. Especially for large values of n, but this is the formula right, this is proved and. So, now, I can use this to right figure out  $(())$ . So, then define my G of N. So, G of N is a... So, normalizing constant, I need this to ensure that all the probabilities are summing up to 1 right, and they are less than or equal to one. So, this is simply sigma over all possible values of n right, every computation of n.

(No audio from 06:54 to 07:23)

So, to avoid this computation problem, we can look at two ways right, if D i is very large then, but a good thing is the D i's are again appear in the denominator right.

(Refer Slide Time: 07:41)

To reduce computation overflows, etc.  
 select some scaling factor  $\alpha$ , ( $\alpha < 1$ )  
 $\downarrow$   
 If  $D_i > 1$

Let  $y_i = \alpha D_i$

eg.  $\alpha = \frac{1}{D_{avg}}$  or  $\alpha = \frac{1}{D_{max}}$

$P(n) = \frac{\prod_{i=1}^M y_i^{n_i}}{G(N)}$ ;  $G(N) = \sum_n \left( \prod_{i=1}^M y_i^{n_i} \right)$

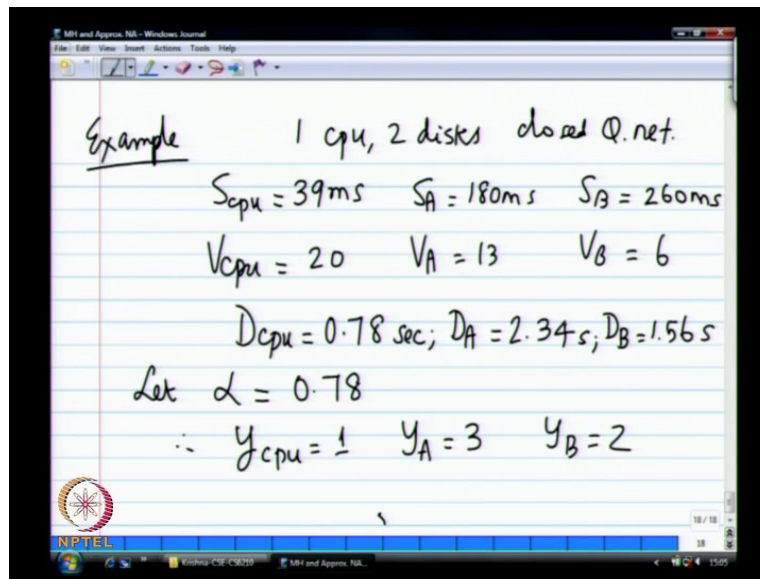
So, I can simply scale the  $D_i$  to whatever value that I want right. So, one way to reduce the computation right, to reduce the computation errors right overflow etcetera. I will simply choose a select right, some scaling factor  $\alpha$  -  $\alpha$  usually less than 1 right, and then rewrite. Are of course, if the  $D_i$ 's are very small then it will scale them up right. So, depends right, if all  $D_i$ 's are right; are all greater than or  $(\alpha)$  scale it some number less than 1 right. And if it is a too small than scale it up the property. For example,  $S$  in a service service time right in a  $(\alpha)$  system, your packet transmit time will be  $r$  of nana seconds.

So,  $V_i S_i$  will be very, very, very small, and then you try to that case you want to just scale it up to right  $(\alpha)$ . So, we now express everything in terms of this  $y_i$ ,  $y_i$  is nothing but  $\alpha$  into  $D_i$ . And  $\alpha$  can be anything, you can just pick the largest you know value of  $D$  whatever it is, this is even if it is not greater than 1 you can still use that for in in some situation. But any or if you want to just normalize everything to integers also you can try to do that if; things are  $(\alpha)$  set up multiples of integers. So, for example,  $\alpha$  I can use this so called  $D$  average, I can use that right or I may just say that I will scale everything with respect to  $D_{max}$ . So,  $D_{max}$  closed to 1, and everything else is less than one. So, something of that is all or any other arbitrary value right. So, in the now my  $p$  of  $n$ , which is the same expression that we saw, before is nothing but so, instead of express you need terms of  $G$  of  $D$ , we are expecting expressing in terms of  $y$ .

(No audio from 10:17 to 10:45)

Questions alright.

(Refer Slide Time: 10:56)



Example 1 cpu, 2 disks closed Q.net.  
 $S_{cpu} = 39ms$     $S_A = 180ms$     $S_B = 260ms$   
 $V_{cpu} = 20$     $V_A = 13$     $V_B = 6$   
 $D_{cpu} = 0.78 \text{ sec}$ ;  $D_A = 2.34s$ ;  $D_B = 1.56s$   
 Let  $\alpha = 0.78$   
 $\therefore y_{cpu} = 1$     $y_A = 3$     $y_B = 2$

So, now, let us go back to our the same system with queue right 1 cpu, 2 disks closed network. And let us assume that S cpu is 39 milliseconds, S A is 180, S B is 260 seconds, and then V cpu is again given to you.

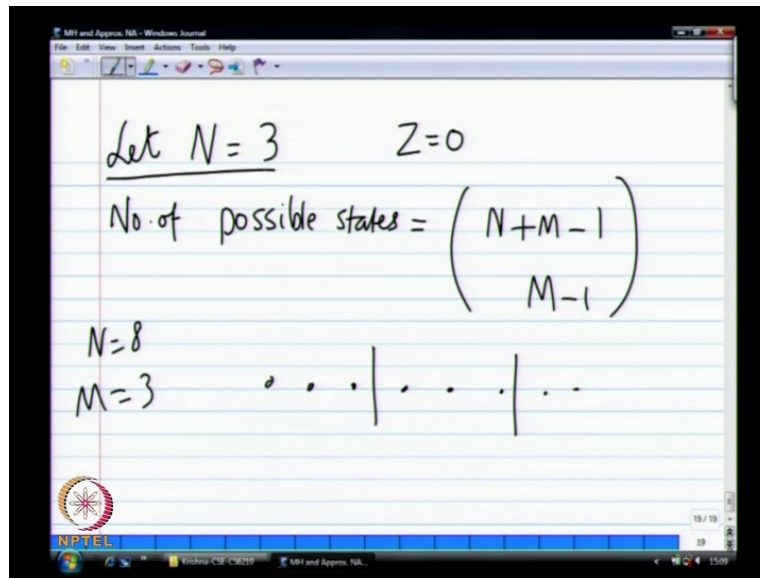
(No audio from 11:36 to 12:06)

So, your D cpu is now, this is in milliseconds. So, we will just put everything in seconds, D A is 2.34, (()) some 13 hiding right. So, so let alpha be 0.78.

(No audio from 12:40 to 12:51)

So therefore, y cpu equals 1, y A an equals 3, y B equals 2 right. y A is alpha D I. So, divide everything by 0.78, we get y i equals no I am sorry 1 over 0.78; (()) correct or not.

(Refer Slide Time: 13:59)



So, now, let us assume that there are is the close networks right. So, let us see that there are total of three jobs in the system at that is all I am considering right; close network with 3 jobs; so what the probability of right (( )) one job in this queue, 2 in the other queue and so on. So, because there are three jobs (( )) been assume that there is no terminal here, that is no thinking time.

So, z equal zero. So, there is there is no terminal at all; it is just the 3 queues, they job there is (( )) just switching between these 3 queues right. So, now, I can enumerate right, if we want to find out. So, what is the probability? What is the different ways in which these 3 jobs can be distribute among these 3 queues, what's that. So, what are the totals number of what are the total combinations for this n right, I said n is the set of all possible right state combinations;  $4 C 2$  that equals  $4 C 2$ ,  $4 C 2$  for a 3 jobs, 3 queues; 3 jobs, 3 queues one before  $C 2$ . So, I have 3 jobs distributed among 3 queues right. So, what is the common into get this look at it somewhere right.

So, the number of possible states.

(No audio from 15:24 to 15:33)

So, I have n objects right, that can be in m queues that can be distributed among m queues. So, what is the number of ways in which we can do that.

5 C 2.

5 C 2 good. So, how do we get that.

Put two partitions in 3 plus 2, 6  $\binom{6}{2}$ .

So, that is basically can you give that in terms of N and M.

M plus n minus 1 n minus 1 plus n; are of course, n right M M minus 1, if it is easy or if you will think about in terms of the partitions right. So, this is the number of ways in each which number of possible states and yeah. So, if again I found is explanation. So, if you represent your. Set of jobs as dots right. So, this is the N dots that you have right, and then you are basically partitioning this into M sets right. You can you can put so, basically bars represent here, boxes I have m minus 1 boxes right.

So, I can put let us say m equals 2 right or M equals 3 in our case. I need draw 3 bars right to basically split; I have in this case N equals eight and M equals 3 right. So, now, I have to look at these 2 bars as where I can place them, I can place them in a right anyone of this it is N plus M minus 1 positions are there right, I can and then we have M minus 1 choices. We have to select M minus 1 position out of the N plus M minus 1 positions possible. So, let us so we get that; that you can checked it out later right, just take this for  $\binom{6}{2}$ . So, there are so these many combinations. So, what is this of the order of what is the order complexity for this; if you have to compute this right, this many states a number of states is how much the order of N to the power M minus 1. So or N to the power M at many states on the then again you can verify.



(Refer Slide Time: 18:11)

For  $N=3, M=3$ , possible states:  $5C_2 = 10$

cpu	DA	DB	$\prod y_i^{n_i}$	$P(n)$
0	0	3	8	$8/90 = 0.089$
0	1	2	12	$12/90 = 0.133$
0	2	1	18	.
0	3	0	27	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
3	0	0	1	0.011
			$G(N)$	90

So, in our example right, for N equals 3, M equals 3, possible states  $5C_2$ ; this is 10. So, you have to enumerate 10 possible states. Why would have to enumerate all 10, because I have to compute G of n right; G of n is the summation of overall n, which is a the  $y_i$  to the power  $n_i$ . So, have a basically have a have one big table, that actually does that. So, let us compute some parts of this table. So, let this is the number of items in the cpu, number of items in disk A, number of items in disk B, and right this is your  $y_i$ , (( )) sorry  $y_i$  is we already known. So, this is this, and then right, for this combination we can look at the...

So, for every combination I would gone figure out this  $y_i$  to the power n again right, and  $y_i$  (( )) figure out to be 1, 3 and 2. So, when this is one possible distribution right, all the 3 jobs are in the disk B. So, what is the corresponding product here 8, and we can just do this for the rest of this class, but then I put 0 here; 1, 2, 0, 2, 1, then I have 0, 3, 0. And this is now 3 to the power 1, 2 square this is 3 the power 2, 9 into 218 yeah. And so forth until we look at the final combination is 3, 0, 0, because the  $y_1$  equals 1 this just sums up to be 1.

So, remember my G of n is summation of all this (( )) which is 90. Once I compute G of n, then my formula says right p of n equals this product divided by G of n. So, what is this? This is 8 by 90 that equals 0.089; this is 12 by 90 and so forth, dot, dot, dot. So, that is have I computed yes. So, given a system I would to go back write a small program to do this right, only worrying about overflow or underflow, but I can do that.

Multiplied by p of n  $(\alpha)$ .

I am sorry.

Multiplied by divided  $(\alpha)$ .

No, alpha does not come in this picture; thus alpha is already in the denominator on the numerator.

p of n divided by  $(\alpha)$ .

y i is reduced, and likewise yours. So, both numerator, denominator both reduce by alpha right. So, you do not need to reduce this; we use alpha later on when you come to the actual through metrics will have to use alpha to scale it back to the original values. So, that is one way of computing right; there is another way of computing g of n, m. Do you know that is. So, given n and m, we can compute g(n,m) faster.

(Refer Slide Time: 22:49)

Let  $g(n,k) = \prod_{i=1}^k y_i^{n_i}$

Then  $g(n,k) = g(n, k-1) + y_k g(n-1, k)$

$g(n, 0) = 0 \quad \forall n=1,2,\dots,N$

$g(0, k) = 1 \quad \forall k=1,2,\dots,M$

Table with  $(N+1)$  rows &  $M$  columns.

So, let us do that. So, let us right g(n, k) is defined this way, then they can actually prove this, but I will keep that for some other time. Basically, I will get why. So, the number your g(n,k) right.

(No audio from 23:24 to 23:46)

So,  $g(n,k)$  this is the right expression that takes of these two. So, this simply says that in the particular  $k$  queue  $k$  that you are looking at, there is no job. So, all the  $n$  jobs that is being in the  $k$  minus 1 queues right, that is 1 tag queue  $k$  that you looking at. So, it is either all the jobs are in the other queues  $k$  minus 1 queues or there is 1 job, which is as  $y_k$ . And then the remaining  $n$  minus 1 jobs are still across all the  $k$  queues right. So, one of there is jobs that we are looking at putting them in these queues. So, we are just one job you are tagging, and then that is even in this  $k$  th queue or not in this  $k$  the queue, and this is this is the property. So, this was  $g(n,k)$ . So, why this is use for us, now I can actually compute my  $g$  of  $n$ 's much faster right, with the simple set of multiplications and additions.

So, now, let us define some boundary conditions right. So, what is  $g$  of  $n$  comma 0 equal to how many ways can we distribute  $n$  objects in 0 queues 0. So, this is boundary, and then how many ways can we distribute 0 objects in  $k$  queues; exactly one way right, all 0 for all  $k$  equals 1 to  $m$ . So, in our constructor table with  $N$  plus 1 rows and technically  $M$  1 plus columns, but you do not to worry about  $M$  columns, and proof is there on the book right, how we can do this grouping for a simple  $g(n,k)$ , this 3 queue, 3 customer case you can actually come up with that is all .

(Refer Slide Time: 26:09)

	$(y_1)$	$(y_2)$	$(y_3)$	$\dots$	$y_k$	$\dots$	$(y_m)$
0	1	2	3		$\cdot k$		$M$
1	$y_1$	$y_1 + y_2$	$y_1 + y_3$		1		
2	$y_1^2$						
$\vdots$							
$n$					$g(n,k-1)$	$g(n-1,k)$	$g(n,k-1) + g(n-1,k)y_k$
$N$	$y_1^N$						$g(N,M) = G(N)$

So now, my table essentially looks like this a.

(No audio from 26:08 to 26:30)

And this is basically. So, they are assuming that the table has this row values. So, the number of ways of distributing 0 jobs right. So, we set that 0 this is 1, and then so we will initialize these 2 initial first row and the first column. This is y 1 sorry.

(No audio from 27:04 to 27:30)

So, this will be 1 queue right, the first column is simply say that only 1 queue, and there are N elements. So, the therefore, all if are all n i right, the corresponding value simply y at the power n I; there is no other queue to be consider in the system, Right that is your these are your initial conditions right, we know that 0 customers in N queues, wherever be the number queues is the probabilities or g of n right; g is 1 and if there is only 1 queue, it is simply y 1 raise to that power n 1 right.

So, that is have these 2 columns are generated. So, now, we just simply use our previous expression right. So, you look at the it is a.

(O)

Yeah. So, let us look at some. So, there is y k here, right this is your k th column; that is also one. So, what you have you have g this is n eth right. So, your g(n, k minus 1) comes from your previous column, and then you have g(n minus 1, k) from your previous row right. And this is may, this is simply that is alright. So, how many computations now N, m right, whether every computation is simply an addition and multiplication. So, that is all we can build my, this is I think (O) algorithm. And then finally, you compute g(N,M).

So, without having to enumerate, we simply have N, M computations. And the previous case you are do all the n plus m minus 1 choose n options; now you do not have to do that you can just compute this, this one right. So, if fill up this one, then you can fill up this one. So, you can fill row by row. So, as it go along g of 1 or a g of 1 comma M right, g of 2 comma M, M is everything is getting generated. And we also will find that g N minus 1 is also right, this g(N,M) is basically what we are looking for G of N right.

Sir, g n minus 1 k (O), y n minus 1 to the power...

So, this will be y 1 plus y 2 here.

y n minus 1 to the power k.

Not power simply  $(())$  this  $y_1$  plus  $y_2$  into  $y_2$  plus  $y_1$ ; this will be it is this simply comes as only the one coming above is gets multiplied by another value of  $y_k$ . You get  $(())$ , because you are multiplying you are raising things to the power. So, your  $y_k$  is getting  $(())$ . So, we will have one more step, and then we will go to that reaction that people wanted.

(Refer Slide Time: 30:57)

$n$	$y_{cpu=1}$	$y_A=3$	$y_B=2$
0	1	1	1
1	1	4	6
2	1	13	25
3	1	40	90

$$U_i = y_i \frac{G(N-1)}{G(N)}$$

$$U = 1 \times \frac{25}{90}$$

So, can we compute this for N equals 3, M equals 3.

(No audio from 31:00 to 31:30)

Since, this is one, this is very simple right. So, 1 plus 4 into 3, 13, 13 plus 6 into 2, 25 then 1 plus 39 is 40, 40 plus 50, 90. So, in the process, we also computed. What a 2 jobs are placed in 3 queues; that  $G$  of 2 is basically 25, and 1 job placed in 3 queues also simply, 6 different ways of doing that  $(())$ ;  $G$  of  $n$  is basically 6.

(No audio from 32:20 to 32:38)

So, we will come back next class, and look at actually how can derive some all the terms, but since there was a request to see whether  $(())$  you runs or not. So, next 20 minutes I am going to just check out the  $(())$ , hopefully it runs right. So, just as a right these are so, if you want to find out utilization of  $q_i$ ; that is simply given by we will we can derive this from one known expression right, so for example. So, if I am looking at the  $u_{cpu}$  utilization of this  $cpu$  is how much,  $y_i$  is 1 right and  $G(N-1)$  is 25, 25 by 90. So, that is your  $u_{cpu}$ , we can go back

and check with  $m V a$  if you want the other calculation. So, essentially everything is proportional to  $D_i$ , and this ratio of  $G(N-1)$ ,  $G(N)$ ; all utilizations, queue utilizations. So, once you get  $u$  then, you can compute other things right. Then  $S_i$  can compute which is  $S_i$  by  $1, 1 - u_i$  right, and so on.