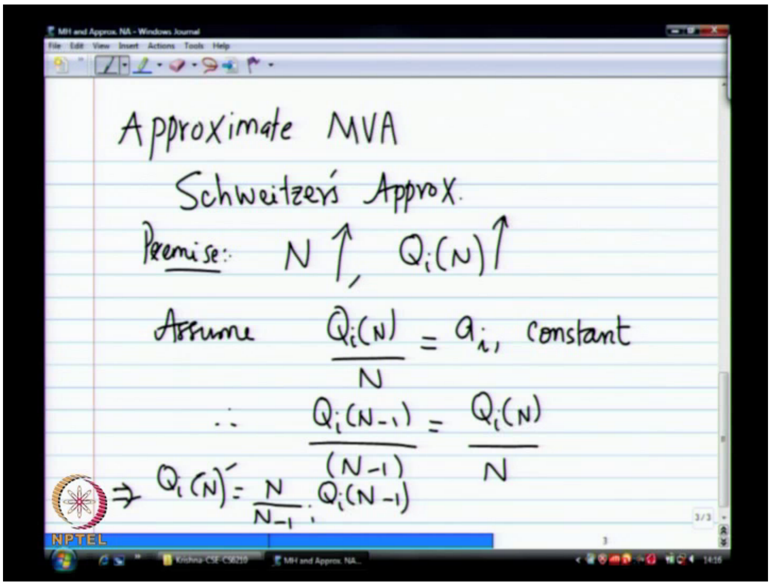
# Performance evaluation of computer systems

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

## Lecture No. # 27
## Approximate MVA

(Refer Slide Time: 00:10)



Approximate MVA

Schweitzer's Approx.

Premise: $N \uparrow, \; Q_i(N) \uparrow$

Assume $\frac{Q_i(N)}{N} = a_i$, constant

$$\therefore \frac{Q_i(N-1)}{(N-1)} = \frac{Q_i(N)}{N}$$

$$\Rightarrow Q_i(N) = \frac{N}{N-1} \cdot Q_i(N-1)$$

So, last class we looked at MVA right, which has computational (( )) that depends upon N, the number of customers in the system; and as we said, as we saw that can be very time consuming especially, for a very large systems, then this becomes a problem right. So, there have been several approximations, and this is an old approximation called Schweitzer's approximation. There are several others that have been proposed; but since this is there in the text book, we just used this as an introduction to how we can do approximate. So, here we replace that a recursive definition with iterative definition of (( )).

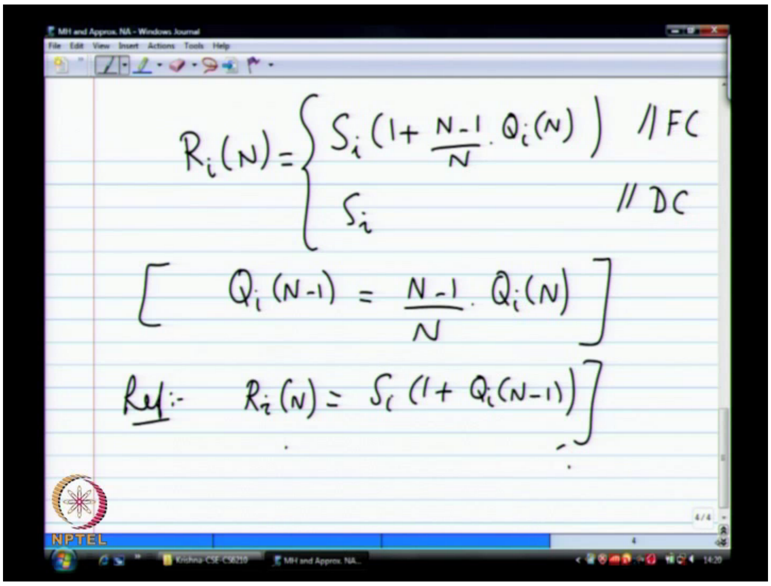
So, this is the so-called Schweitzer's approximation. So, the basic premise is that, which we can argue right; as N increases right, then the Q i the number of customers queued in a particular queue is also going to increase right. And then what he is assuming that that this is that the rate of increase right, so Q i by N is basically assumed to be a constant; from constant a i right. So, it is a assuming linear relationship between the numbers of customers queued at a given queue to the total number of customers.

So, if I make this assumption then, I can simplify and then right then we try to (( )). So, therefore, Q i, so with N minus 1 customers, this is the ratio, and then with N customers, this is the ratio right; so, that implies Q i of N equals N by…

(No audio from 02:34 to 02:58)

So, that is the basic expression that you are going to use for Q i of N. The previous case it was, we use that a Q i is in a right at the end of the expression right, we computed a R of N, then we computed R, then we recomputed the Q is based on the that (( )). So, let us see how it looks like.

(Refer Slide Time: 03:27)


MH and Approx. NA - Windows Journal
File Edit View Insert Actions Tools Help
NPTEL

$$R_i(N) = \begin{cases} S_i\left(1 + \frac{N-1}{N} . Q_i(N)\right) & // FC \\ S_i & // DC \end{cases}$$

$$\left[ \quad Q_i(N-1) = \frac{N-1}{N} . Q_i(N) \right]$$

$$\underline{Ref}:- \quad R_i(N) = S_i\left(1 + Q_i(N-1)\right) \Big]$$

So, now our response time right R i is now S i into, this was Q i of N right, so instead of Q i of N, I am going to replace this with…

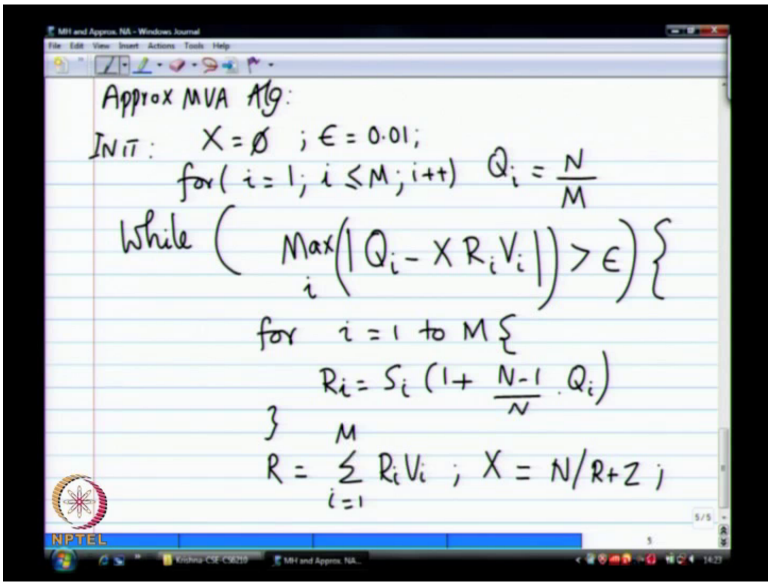
(No audio from 03:37 to 04:06)

That is I had written that the other should have been Q i of N minus 1 equals right, because in the original expression, we add Q of N minus 1, I am simply expressing that in terms of N again right. So, there, so R i of N simply depends on Q i of N that this approximation N minus N by N right. Previously we had equals 1 plus Q i of N minus 1 right, that is a number of customers that I had with N minus number of jobs queued at Q i when there are N minus 1 customers right.

So, again refer your previous your R i of N was S i right; this is what we used last time. So, that is (( )) recursive. This is again for your fixed capacity, and it is simply S i in the case of a delay center. So, instead of this Q i N minus 1, I am using this expression. So, now N depends only on N does not depend on N minus 1 right, R i of N does not depend upon Q i of N minus 1 anymore simply Q i of N itself. So, this is the simplification.

So, what we do is we start with some random value for Q I, and then it with this value until you converge it some point. And again this proof of convergence is not there. So, and also depends on the initial value the time to converge depends upon your initial value for Q of i; and convergence also not guarantee, but what people have done is you just trusted it empirically with different systems, and found out whether this is converging for large N is (( )), if you use N equal say 10000, if I can converge a 100 iteration is still better than converging a requiring 10000 iterations with the original, with exact MVA. So, that is right previous one is the exact MVA, this is your approximate MVA. Is it clear? What we trying to do?

(Refer Slide Time: 06:23)



Approx MVA Alg:

```
INIT:   $X = \emptyset$ ; $\epsilon = 0.01$;
        for( $i = 1; i \le M; i{+}{+}$)   $Q_i = \frac{N}{M}$
While ( $\max_i (|Q_i - X R_i V_i|) > \epsilon$ ) {
    for $i = 1$ to $M$ {
        $R_i = S_i \left(1 + \frac{N-1}{N} . Q_i\right)$
    }
    $R = \sum_{i=1}^{M} R_i V_i$ ; $X = N/R+Z$ ;
```

So, the actual algorithm will now look something like this. So, your approximate MVA. So, initialization is that you know, your throughput is 0, and then for since, there are M queues, we just need Q i, and I am simply assuming that the N jobs are evenly distributed among the M queues. So, this is my initial assumption, I could use anything else, but I am just you could put all the N jobs in Q 1 if you want and let others to be 0, but I am simply assuming that they are all evenly distributed across all the queues, which is incorrect right, because you know that some queues will be larger will be having more customers than other queues.

And then, so now it is a big while loop, and then we will initialize this source some, initialize epsilon to some number right, this is your termination condition.

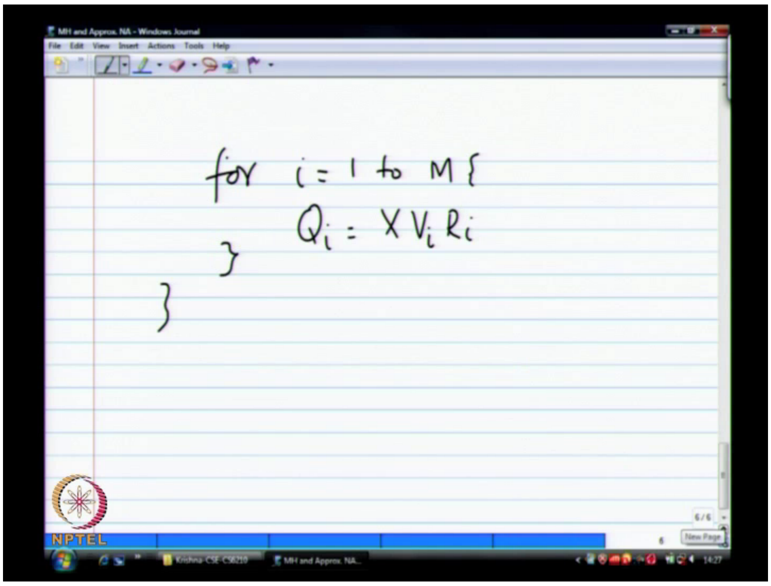
(No audio from 07:30 to 07:46)

So, when you converge, your Q i should equal to your X R i V i. What was previously… So, in this case, Q i is initial value right, that will certainly in R d equal to X, because X 0 initially right. So, assume keep updating. So, this is the case for the differences between your actual queue right, whether it is computing, and that is with two different expressions, you are computing a queue values; and I will look at the max value right. So, the max difference between these two expressions for queue length, if that is that is greater than epsilon, I keep looping.

I am sometime using C syntax, sometime using simpler syntax. So, so R i again these are all arrays right, you know is simply 1over... So, this is your updating of R i based on the previous value of Q i; and then I compute R, which is that; then I go for the run compute X, which is X N by R plus Z, then I have to update my Q i right.

(No audio from 09:38 to 10:14)

(Refer Slide Time: 10:22)



```
    for i = 1 to M {
        $Q_i = X V_i R_i$
    }
}
```

So, look funny to you. So, I am recomposing Q i, which is equal to X V i R i. So, where is the condition will terminate? Another something (( ))

(No audio from 10:56 to 11:54)

It should be the other way around right.

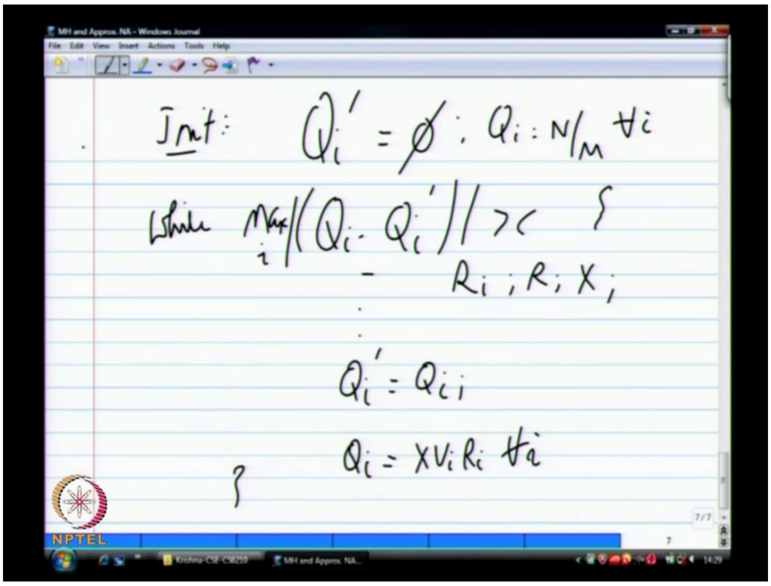
Sir, why…

It should be Q i, it Q i and then the in the previous iteration of Q i right. The previous iteration Q value and the current iteration Q value should be convergence. So this I am just assigning Q right. If it is a do-while, then even then you would have the update here. So, unless I have Q dash and then Q i right, so the previous computed queue, and then this one should be different. That is what the example is trying to do, but the code is that (( )).

Sir, why do not we take…

And I want to measure that the largest difference between the previously computed right. So, if you were to update this, I would keep that as a Q i right, and then Q i will be, I would basically store that previous value of Q i in something some temporal variable right. Safe I keep this is Q i, then I should replace this with Q i dash. Before I calculate, I should before. So, let us try to do this, a let us do this as a Q i dash I do not have here right, if I replace this with I need something for Q i dash. So, let me just you know, what changes I am trying to do.

(Refer Slide Time: 13:47)

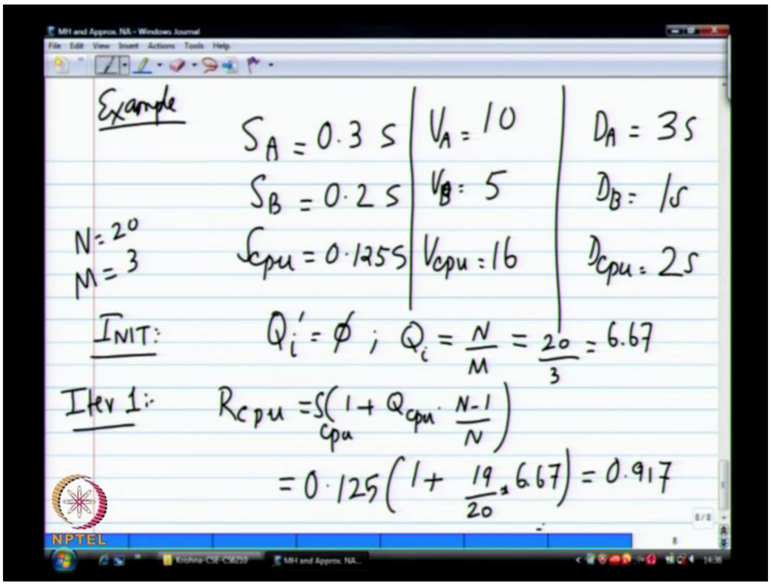


```
Init:  $Q_i' = \phi$ ;  $Q_i = N/M \ \forall i$
While $\max_i \|(Q_i - Q_i')\| > \epsilon$ {
        -      $R_i$ ; $R$ ; $X$ ;
        :
        $Q_i' = Q_i$ ;
        $Q_i = X V_i R_i \ \forall i$
}
```

So, I will assume that Q i dash equals 0 right in the init, and then I simply search right this is what max of i, while this is greater, this condition we will have. And then later on at the end, before I do my… Thus it make sense, so this keeping the previous value of Q i, and then looking at where the Q i are converge right. I am doing everything as before right, the same while loop, I had my Q i equals right N by M for all i, this is what I had before, we computed R i, we computed R we computed X right. Once we do all these computations, then this saving at storage block; if you not convinced, implemented (( )).

So we look at the example one, this will be clear right. The example has it correct or this will be support. So, let us go back to our previous example, the book example by the way for this is actually, all the values are incorrectly computed. So, you should make a note of that.

(Refer Slide Time: 15:24)



Example

$S_A = 0.3\,s$ | $V_A = 10$ | $D_A = 3s$

$S_B = 0.2\,s$ | $V_B = 5$ | $D_B = 1s$

$S_{cpu} = 0.125\,s$ | $V_{cpu} = 16$ | $D_{cpu} = 2s$

$N = 20$
$M = 3$

INIT: $Q_i' = \phi$ ; $Q_i = \frac{N}{M} = \frac{20}{3} = 6.67$

Iter 1: $R_{cpu} = S_{cpu}\left(1 + Q_{cpu} \cdot \frac{N-1}{N}\right)$

$= 0.125\left(1 + \frac{19}{20} \times 6.67\right) = 0.917$

So, example of the same example we saw before right. So, we had S A was a service time of the CPU was that sorry that is the tool.

(No audio from 15:37 to 15:48)

So, what were the (( ))? 10. So, this is 10, 5 and 16. Remember 10 visits to disk A 5 to disk B and therefore, CPU were 16, the same thing we did last class; and the D A is where 3, 1 and 2. This is S have in seconds. So, mostly we just look for this right. This is (( )) S V and after that you can get the D. So, now the first step is to find out R right Init Q before I can find R. So, initialization, I said we will use Q i dash equals 0, and then Q i was N by M and N and we said was 20 right M equals 3, 20 by 3. So, therefore, it pretends that each initially each queue has 6 customers, 6.67 customers that is our start for the iterations.

So, first iteration - So, what is R cpu equal to S cpu into 1 plus Q cpu into N minus 1 by N right.

(No audio from 17:46 to 18:22)

So, what is this equal to?

(No audio from 18:24 to 19:15)

Is that right?

Sir, can we calculate of the R Q CPU we are got S A V A and d a instead of taking 1 by N calculate also…

From how (( ))…

(( ))

This This is this is all that you have. So, if compute your Q, you need to know X right, we have X R i V i right. So, you do not know the throughput not you know the response time that is why to step by step right.

Stronger assumption Q i will be N by M

That is that is we are simplifying a assumption, we if it converges, it will converge regardless converge you got the initial value you have. You can also like this a right queue N equals a, put all N jobs in the first queue, and then see what happens right, the system really converge after that or not? But at some points, it should converge. So, what is the answer?

(( ))

0.91

(No audio from 20:16 to 20:34)

So, the book was wrong, and I was also wrong in my computations. I also ignore the 19 by 20 was ignored. So, the book actually has this as 1 plus 6.77 can be 6.77 right, 19 by 20 into 16.67. Let us compute the next one R A is…

Sir when we are taking this Q cpu was 6.67, we are using Q of i,

Q of i.

Q of i.

This one.

We are not taking N minus 1.

It should be…

It should be N equal to 1 right.

It should be…

N equal to 2.

N equal to…

See, the original recursive form was taking just keeping on adding one more one more customer, one more customer.

The previous one yes.

So, in this case, it is used Q i of, this is 1 plus Q i of N minus 1.

Accommodated and we are trying to compute.

No I am resuming that out of the all the 20 right, there are 20. I am starting off by saying that they are evenly distributed among all the three queues. And then I am using that value to keep on recursive, so that right. So, instead of using Q of N minus 1, I am using Q and itself as my recursion for iteration sorry right.

(( ))

No no that is, that is input, no that is input in such cases, you want to look at the throughput, if there are 20 customers in the system.

(( ))

The previous case, I am again find for N equals 20, but I have to start from N equals 1 and step by step right

(( ))

Until N equals until N equals 20.

(( ))

So, that is that is if we are when very large, then you will have order and learning time right. So, therefore, you want to reduce that by iterating on the with some approximate value, and then you keep updating this value until you converge on some Q i values that do not change beyond a point.

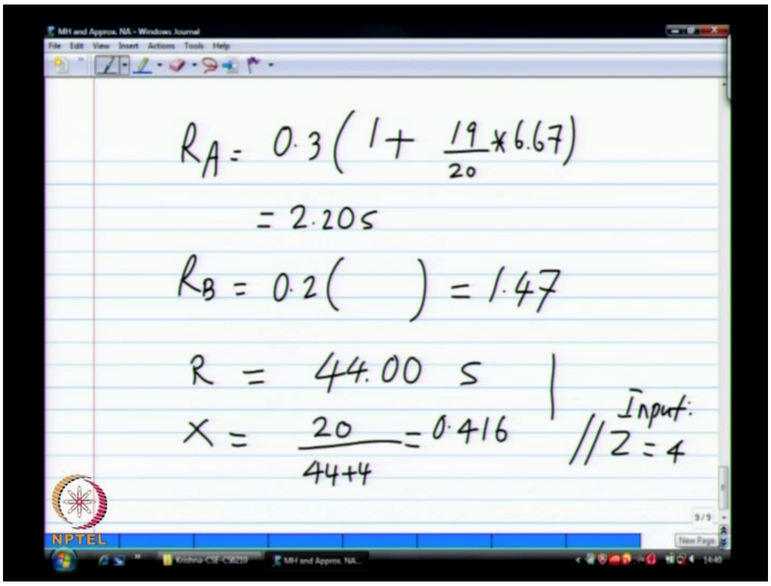
(( ))

So, some initial value of initial queue; no N is an input, N is not initial value, N is an input.

Some initial value, where we start?

Initial value for the Q i, you nead some starting value.

(Refer Slide Time: 22:54)


MH and Approx. NA - Windows Journal
File Edit View Insert Actions Tools Help
NPTEL

$R_A = 0.3\left(1 + \frac{19}{20} \times 6.67\right)$

$= 2.205$

$R_B = 0.2(\quad) = 1.47$

$R = 44.00 \text{ s}$

$X = \frac{20}{44+4} = 0.416$

// Input: $Z = 4$

So, this is going to be S A, which is 0.3 1 plus…

(No audio from 23:04 to 23:15)

And what is this going to be…

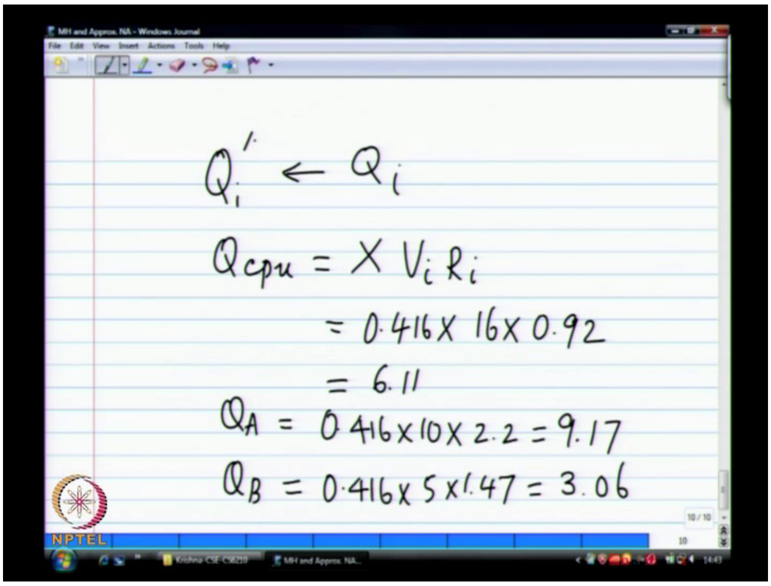
(No audio from 23:18 to 23:40)

2.2 (( )). So, in the book values are correct, but expression is not correct. The same as before, so what is this going to be 2.2 into 2 by 3, 4.4 by 672 1.

(( ))

1.467 to 7.47. R - 44. 0085, so we will stop 44. So, this is the overall response time of the system right. So, now the throughput is 20 by… I did not specify Z, Z was 4 last time right. So, remember that Z equals 4. So, 20 by 44 plus 4… 416.

(Refer Slide Time: 25:30)


MH and Approx. NA - Windows Journal
File Edit View Insert Actions Tools Help
NPTEL

$$Q_i' \leftarrow Q_i$$

$$Q_{cpu} = X V_i R_i$$
$$= 0.416 \times 16 \times 0.92$$
$$= 6.11$$
$$Q_A = 0.416 \times 10 \times 2.2 = 9.17$$
$$Q_B = 0.416 \times 5 \times 1.47 = 3.06$$

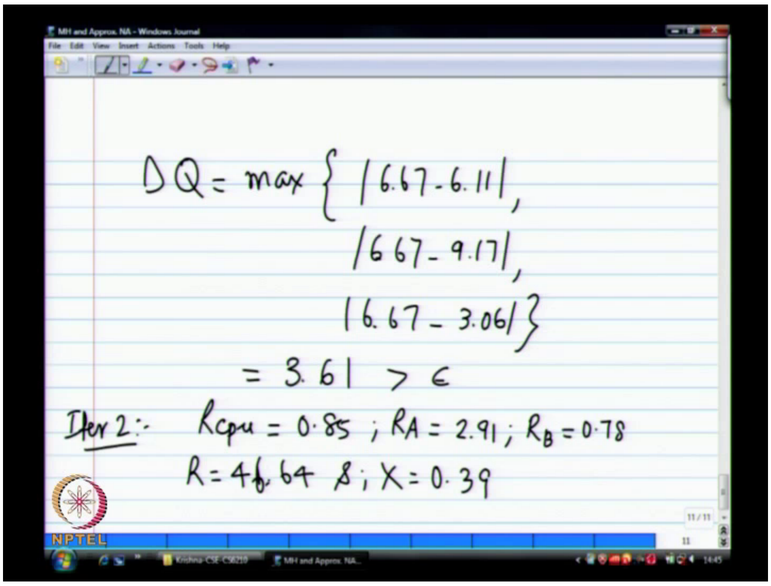
So, the now I am going to store my… So, we just store the current values of Q i, and this Q i dash; and then I compute the new Q is right. So, now, I know my X, we know we are we have if computed R i right. So, I can go back and recompute this, so this is point Z - 0.416 into 16 into sorry R i was 0.92. So, we have, from 6.67 we have recomputed Q CPU to be 6.11, and Q A is likewise 0.416 into 10 into we computed this to be 2.2. So, this was 9.17.

So, that is our first update right. So, we vary started with N by M 6.67 for all of them; now because of the fact that the disk A is got a higher service time, you are finding that the system is saying that there will be more jobs in disk A than in the other two queues right.

(( ))

Does it compute 20? Some approximation, some things are missing, but should come roughly 20. Is it 9 is coming to… So there is also some amount of time in thinking right. So, there is that equals 4 right. So, and what happening is your probability of jobs being in think state is 4 divided by 44 plus 4, R is 44, 4 is right. So, probability of job being that is 4 by 44, 4 by 48 that is 1 over 12, and that is 0.8 that is the missing 0.8.

(Refer Slide Time: 28:38)



$$\Delta Q = \max \{ |6.67 - 6.11|, |6.67 - 9.17|, |6.67 - 3.06| \} = 3.61 > \epsilon$$

Iter 2:- $R_{cpu} = 0.85$ ; $R_A = 2.91$; $R_B = 0.78$

$R = 46.64\ s$; $X = 0.39$

So, now, I look at right the delta Q, which is max of 6.67 minus 6.11, and then we had 6.67 minus 9.17, and then 6.67 minus. So, this happens to be, which is still greater than epsilon.

(No audio from 29:04 to 29:15)

So, these are the new queue values right. So, one more iteration, with the new queue values recompute R of i recompute everything. So, after this second iteration, I will just give the results (( )) right. So, R CPU gets recomputed to be 0.85 R A gets to be actually, R A is upward revised R b is 0.78, and then R itself is 44.64 seconds, X becomes 0.39. So, this is, the same computations repeated in the new values of queues. Questions, 0.39 that is the revised text sorry, this is 46.64.

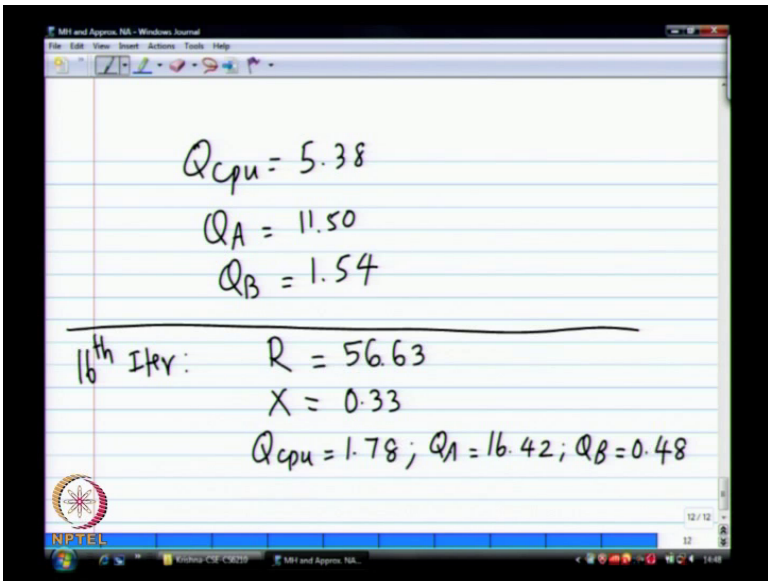
(( ))

But if we simplify Q i minus X V R i, then since we just computed Q i by simply use my program as such right, I just updated Q i equals X V R I, then I cannot go back and say Q i minus that this going to be a greater than epsilon.

(( ))

That is that is we need something to store the old value yes. Mathematically you can say that, but when you write, it is the program we have to make sure that expression.

(Refer Slide Time: 31:12)



$Q_{cpu} = 5.38$

$Q_A = 11.50$

$Q_B = 1.54$

16th Iter: $R = 56.63$

$X = 0.33$

$Q_{cpu} = 1.78; Q_A = 16.42; Q_B = 0.48$

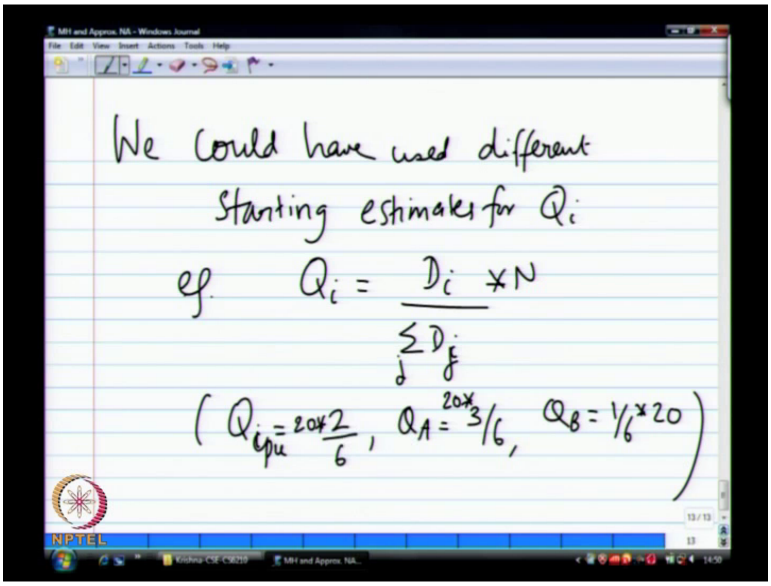
So, after this X computation, we now recompute the Q CPU to be 5.38, Q A to be 11.5, then Q b to be see notice that right, there is more more Q build up that is applying in Q B are which means their initial estimate was totally way off right. So, this is your second iteration. So, you repeat this the little program, and the book finally, shows for N equal or for the 16 eth iteration actually, we are sort of converging on the queue lengths, which is a kind of silly, because my N was 20 right; so, but if you choose may be N equals 100, you will probably might have converged in the 16 eth iteration. So, it is so really there, then unless you have solve it proves that it convergence with significant less time, this might end up taking much longer than simply running at N times right.

So, so finally, if you look at the 16 eth iteration, then the R is going to 46.63, X is settling at 0.33 right, this we knew from last time that is our upper limit 1 over d max. So, X is finally, converged. And then the queue lengths actually it is…

(No audio from 32:34 to 32:49)

So our total initial estimate was totally way off right; assuming that that is equally distributed across all the queues, because Q B is actually 0.48. So, took a while for Q B to come down to 0.48 and Q A is 16.42. So, the initial value makes a difference in your computation right. So if you want to be really more efficient, you could probably look at the service times or demands right for those three queues and use that as an (( )) you simply a D A by sigma D i right would give you a better approximation. We say that if it proportional to the demands right, the amount of time, number of customers queued is going to be a proportional to the demand on that queue itself. So, you can, may be use that right, because the demands are 3, 1 and 2. So, you know that the queue and this case is going to be larger than the others right. So, there is simply set 3 by 6 and so on right.

(Refer Slide Time: 33:48)



We could have used different starting estimates for $Q_i$

eg. $Q_i = \frac{D_i}{\sum_j D_j} \times N$

$\left( Q_{cpu} = \frac{20 \times 2}{6}, \; Q_A = \overset{20\times}{3}/6, \; Q_B = 1/6 \times 20 \right)$

So, we could have used different start estimates for all the Q i(s) right. So, for example, I could have said that Q i simply right Q j. So, in this example, this would have been Q cpu would have been 2 by 6…

(No audio from 34:29 to 34:57)

That could still be incorrect; it should be 40 by 6, which is actually (( )) 1.78, but anyway may be this will give a less iteration, so that we can verify with a computers. That is a approximate MVA. So, this is exact and approximate MVA. Questions, before we go to something even more. So, this is (( )) mean value analysis right; only getting the mean queue length for example, if I ask you what is the distribution of the queue lengths right, number of customers at a particular queue. We know that for MM 1 right; we know that for MM 1 queue, the number of customers probability that there are 4 customers in the queue is what? rho i into 1 minus rho. So, we know that from over MM 1 results. We want something similar in this case. So, that is the next step now.

We now move to what is called (( )) we will right; more detailed analysis, not just the mean analysis, we would like to know exactly the number of customers, the distribution of the number customers at queue and then do lots of analysis based on that. So, I am skipping 1 part of the chapter 34 right now that is the balance job bounds. We saw the bounds last time right 1 over d max and so on we can improve that as I said, but I will skip the derivation of that to after I finished this computation algorithm.