Performance Evaluation of Computer Systems

Prof. Krishna Moorthy Sivalingam

Department of Computer Science and Engineering

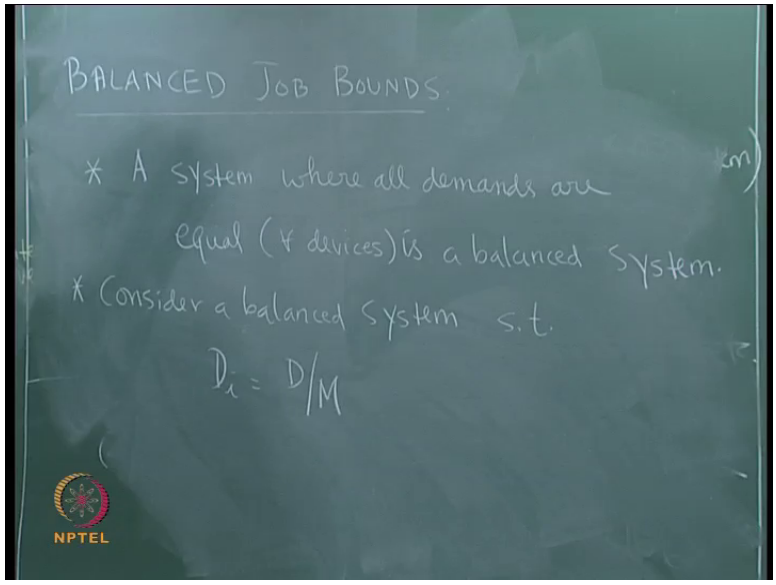
Indian Institute of Technology, Madras

Lecture No. # 32

Balanced Job Bounds

Balanced Job Bounds.

(Refer Slide Time: 00:10)

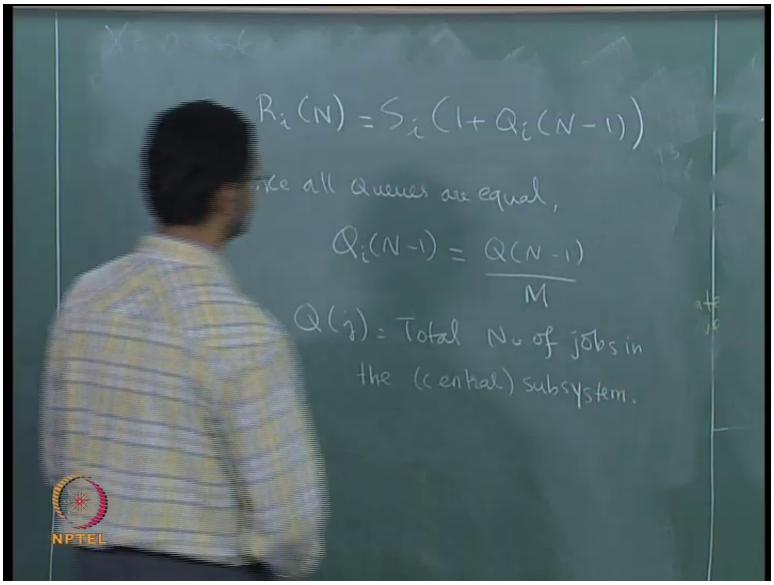


We saw before those R of N bounds. R of N is greater or equal to D, but there is no real lower bound, except when it is less than or equal to 1 over D max at… But, then what is the corresponding bound on the other side? That is what was attempted to solve. Again, this was back in the 80s with the help of this so-called balanced job bounds, which had a flipped back quite a few pages.

What is the balanced job system? There is a definition for that. A system with technically no bottleneck device, there is enough… where all the demands are equal. A system where all demands are equal for all the devices is a balanced system. So, that is the definition. So, all the D's are equal; D is are equal. Therefore, there is no single bottleneck device. Basically, all devices are bottlenecks. Whereas, if there is one device,

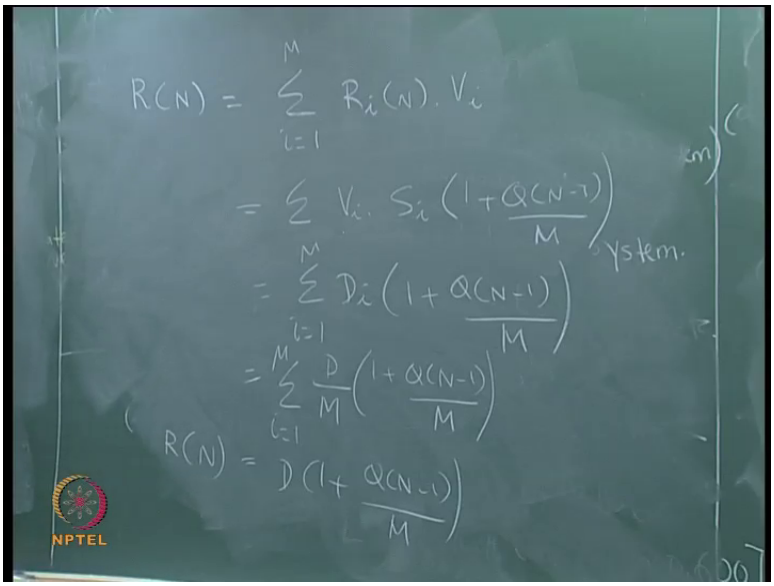
which has D max, then that tends to be one the dictates the throughput. Consider a balanced system such that D i is equal to D by M. So, this is a system, where there is a no terminals; only their c p u discs and so on. This is the central subsystem as you call. So, let us not get look at. So, there are M devices in this queue; and, the total demand is D. Remember, D was sigma D i. And therefore, D i equals D by M for all i.

(Refer Slide Time: 03:11)



Remember R i of N. So, R i of N is basically S i into 1 plus Q i of N minus 1. This is from before, where in the ith queue, if there are N minus 1 customers already waiting, then the demand that response time for the Nth job is just this one. So, since all the queues are equal, that (( )) Since all the demands are equal, their expectation is that all the queue lines will be the same. So, Q i of N minus 1 is simply Q of N minus 1 by M; where, Q of j is the total number of jobs in the central subsystem.

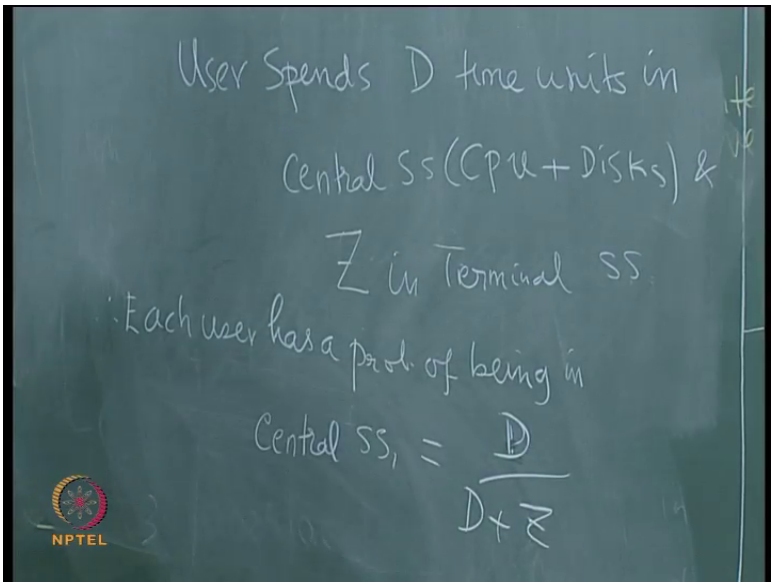
(Refer Slide Time: 05:03)



Now, R of N is summation R i of N into V i for i is equal to 1 to M. Write this we know from before. This again we write this as summation V i into S i into 1 plus Q of N minus 1 by M; equal to summation D i into 1 plus Q of N minus 1 by M for i equal to 1 to M. So, this is the R i – S i into 1 plus Q N minus 1 by M. And, V i S i is D i. So, that is that. And, this is now equal to summation i equal to 1 to M; D i is defined as D by M. So, this is D by M into 1 plus Q of N minus 1 by M. Since there is no need for i anymore here, simply multiply this whole thing by M. So, this is D into 1 plus Q N minus 1 by M. This is R of N. So, for a balanced system, R of N equals D into 1 plus Q N minus 1 divided by N, which we can again expand this as D plus D by M into Q into N minus 1. That is step 1; long derivation. So, this is part 1 out of the four steps. So, this is fair enough.

Now, let us assume that… Here we have a shared system. There is a central subsystem that is shared among these n users in the system.

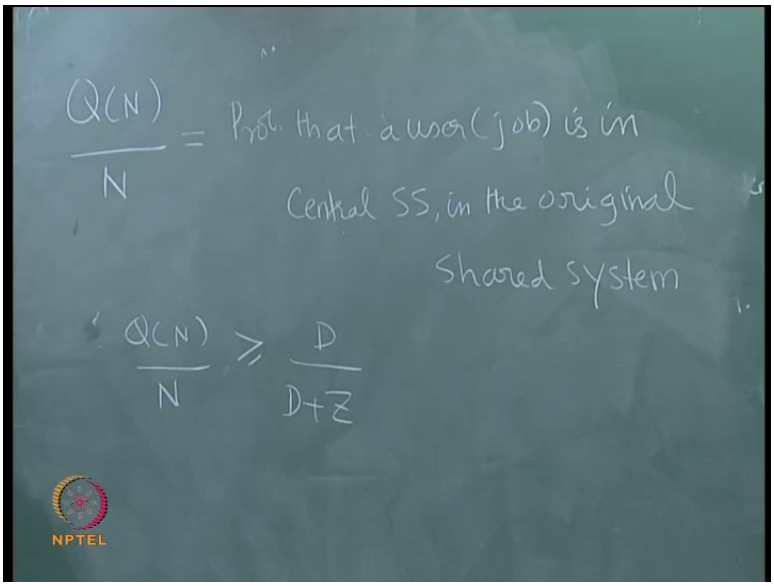
(Refer Slide Time: 08:18)



Let us look at the system, where let each user have his or her own system. So, there is no queuing as such here anymore; I am giving like a PC that is given to each of the n users in the system; total of n users out of which I am giving each user their own system. So, there is no queuing here. This is like zero-queuing system, because here everybody has their own system. So, what is the total time spent? The user spends D times… Write D in the central subsystem. This is the CPU plus disks. And then, Z in the terminal; where we saw that D is the total demand by any job on the entire system. Therefore, D in a (( )) Therefore, if you look at this, each user has a probability of being in the central subsystem that is equal to… So, it is a probability that given user will be in the central subsystem. Either you are in the central subsystem running the computation or you are in the terminal subsystem, where you are simply processing and idling.

What is the probability? D by D plus Z. That is the probability (Refer Slide Time: 10:12). Total time is D here, Z there. Therefore, given probability is just that. Then, if you look at our original definition, we have a system with N users; out of which, some of the users are in the central subsystem; others are in the terminal subsystem. So, the total number of users in the central subsystem is given by Q of N. Q of N is the sum of all the device queues. Therefore, the total number of users in the central queue would basically for processing either disk or CPU is Q of N.

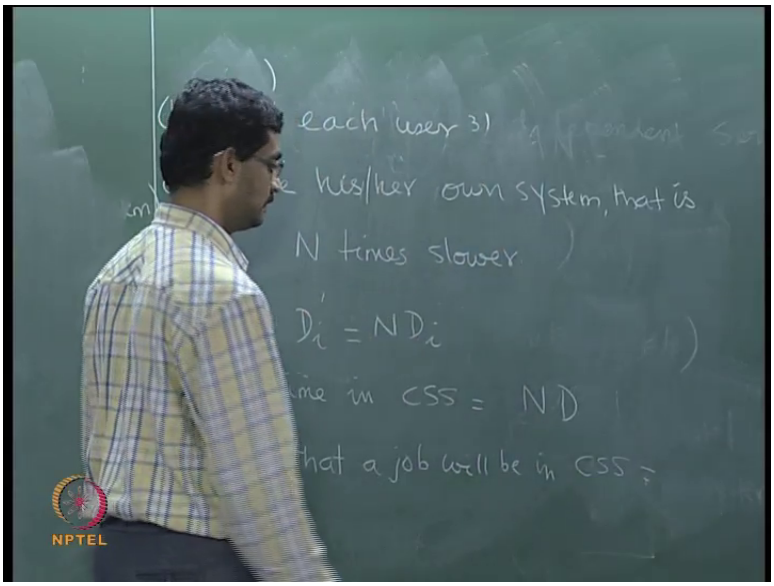
(Refer Slide Time: 10:48)



Now, Q of N by N; out of the N users, queue of N are in the central subsystem; the others are in the terminals, where there is no processing; simply the delay center. So, Q of N by N represents what? Probably that user is in the central subsystem. Out of the N users, queue of N are in the central subsystem, the remaining are in the terminal subsystem. So, Q N by N is the probability that the job is in the central subsystem in the original shared system. In the original system, Q of N by N is the probability that the given job will be in the central subsystem. So, which should be more? This one or that one? This is the probability. And, you have nobody else sharing your system. So, the probability for queuing will be more in the other system. The probability that…

Here there is no queuing at all (Refer Slide Time: 12:10). This is therefore, Q N by N should be greater than or equal to D by D plus Z. So, one system with queuing; one system without any queuing. And therefore, this should be the lower bound. We have to be at least as large as that depending on the value of N. If N is very small, N equals 1, then there is (( )) queuing in the system too. If N becomes larger, then the probability that you going to be Q is larger. Therefore, the probability of being in the central subsystem is greater than the probability that you will have in a system, where you have your own unique system assigned to you. So, that is a part a.

(Refer Slide Time: 13:05)



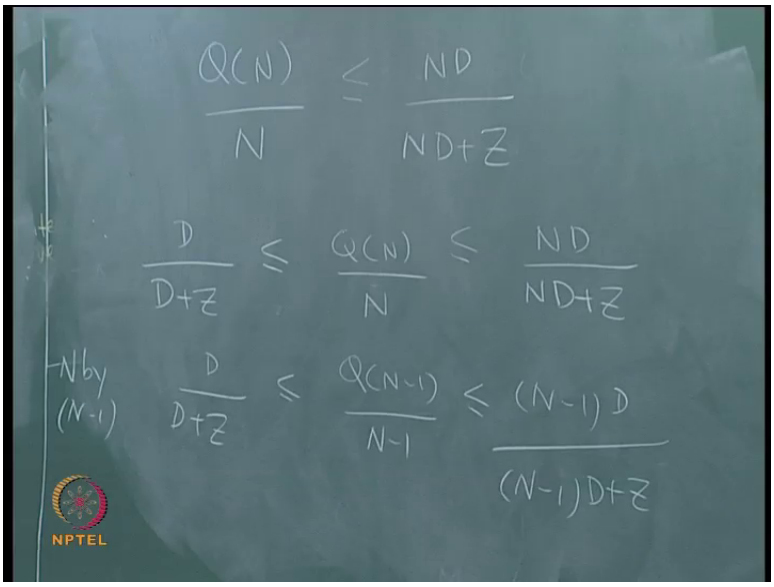
Then, will go to part b. Now, we say that instead of every user having their own system, they still have the same system, but it is N times slower than the original system. So, the capacity of this new system… This is one kind of system. This gives you this bound. We will look at a different kind of system, where every user has their own individual system. But, that is N times slower. Therefore, the effective capacity is same as the original system.

Let each (Refer Slide Time: 13:35) user have his or her own system that is N times slower; which means each device service time S of i is going to be N S i in the new system. That is what it means by being N times slower. If I had S i service unit, S i service time in a particular device in the original system, then in the new system, I have N into S i as the service time. That is what I mean. Therefore, what happens if D i is now N into S i into V i? Therefore, all the D i's become multiplied by a factor of N. So, D i dash becomes N into D i in this new system, because each device is N times slower. And therefore, S i is also lower. But, V i's does not change; the number of (( )) to every device is the same. Therefore, it should be N times D i. The total time taken to process will be N times the original time. That is what (( )) Therefore, time in the CSS equals N into D.

What is the probability that a job will be in (Refer Slide Time: 15:16) the central subsystem is N D by N D plus Z? N D is time spent in the central subsystem; Z for terminal processing. Therefore, this is another bound for the same probability of finding

job in the central subsystem compared to the original subsystem. For Q N by N in the original system, will this be more or less than that? This is N times slower system; whereas, that is N times faster.

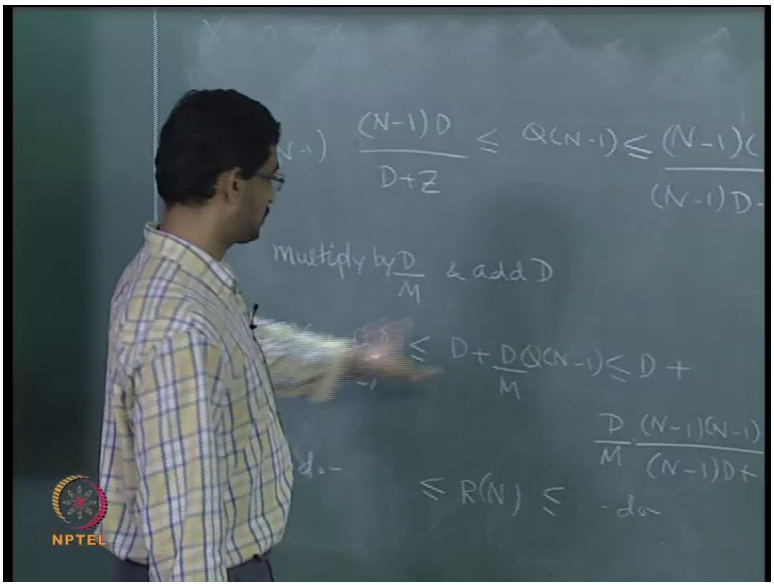
(Refer Slide Time: 16:11)



Therefore, this should be less than or equal to ND by ND plus Z, because the system is N times slower. But, there is only one user in the system at a given point of time. Other system is N times faster, but you will have in the worst case, all N users in the particular system. Therefore, this should be the other end of the bound. Therefore, combining these two fellows, we have D by D plus Z less than or equal to Q of N by N less than or equal to ND by ND plus Z. So, for a balanced system, this is one step. You have some more steps here. So, we will now write the same thing with N minus 1.

Replace (Refer Slide Time: 17:05) N by N minus 1. What do you have? D by D plus Z is less than or equal to Q N minus 1 by N minus 1 less than or equal to N minus 1 divided by N minus 1 into D plus Z. That change.

(Refer Slide Time: 17:54)



Then, we will multiply this entire expression by N minus 1. So, what we will have? N minus 1 into D divided by D plus Z is less than or equal to Q N minus 1, which is less than or equal to N minus 1 into N minus 1 into D divided by N minus 1 into D plus Z. Here we multiply by N minus 1. Now, you add D and multiply by D by M. So, multiply first everything by D by M; then, simply add D for this entire expression. So, what do we have? This is D by D by M into N minus 1 into D divided by D plus Z. This is less than or equal to D plus D by M into Q N minus 1 less than or equal to D plus D by M N minus 1 into N minus 1 into D divided by N minus 1 into D plus Z; just routine algebraic manipulations. So, multiply by D by M first everywhere; then, add D to both sides. So, what do you get know? So, what is this expression finally? R of N. So, that is what we wanted to get to. So, for a balanced system, R of N is given by this (( )) where all the demands are equal. And therefore, all the queue lengths are equal. So, for this system, it is given by this expression. That is step one.

Now, we have to do two more experiments to finally come to the bounds that we want for an unbalanced system. For a balanced system, this is good enough. Then, two more (( )) strips later will come with some better bounds. In fact, you will find that the bounds are actually really much stronger compared to what we saw before. So, I am not sure how much enthu you have for knowing those bounds, but we will anyway proceed with those bounds. But, it will be useful at some point in time when somebody says, here is a queuing network; just give me the bounds; I do not care about running these MVA and all

that stuff; just tell me what the bounds are; give me better bounds. In that case, you can simply just run this formula in two seconds and tell them these are the bounds. That is enough; exact values are not needed; just upper lower bound, which is fairly close to the actual…

After all this computation that we do with x n and all that, we do not have to do that if I can actually use this. That is the reason for going through this. We did all these calculations and finally found that it will give you a fairly narrow set of bounds, which is adequate in most cases. If you are talking to your customer, customer says, tell me what will be the delay for this particular system. You are going to sell them your web processing system from transaction processing system. And then, that they says, there was so many steps in the transaction; give me bounds on this performance. And then, you do not have to go – let me write my axiom code and come back to you in three days from now. You do not have to do that. Or, let me check my spread sheet to do this MVA calculation and come back to you. We can take this as a shortcut and get back to you. That is the reason for going through this.

We will stop here; come back and then use these two stages of the expression that give better delays or better bounds for both R of N and then X of N.