

**Storage Systems**  
**Dr. K. Gopinath**  
**Department of Computer Science and Engineering**  
**Indian Institute of Science, Bangalore**


**Storage Reliability, Performance, Security**  
**Lecture - 28**  
**Performance Modeling of Storage Systems**

Welcome again to the NPTEL course on Storage Systems. Today, we will take brief look at some aspects of Performance Modeling of Storage Systems.

(Refer Slide Time: 00:26)

**Performance Modelling**

- Storage Systems often the slowest component
- Often determine the speed and responsiveness of the system
- Modelling useful
  - Simple models sometimes very useful!
  - Effectiveness of, say, caching may/may not be useful
    - Two zones of operation



We can talking about how storage is often the slowest component, and we looked at disks and the most faster variety SSDs or flash and the future what is called phase change memory PCM - etcetera right; we did not really study PCM, but these are considered featured type of non-volatile storage memories. And when we are designing any large system, it turns out the slowest component determines the overall speed of the system; in our cases slowest parties the disk or the storage system. So, it turns out that it is a slightly good thing for us because often times something is really slow that drives everything. So, the analysis can be somewhat simpler.

So, often times in the storage system kind of situations, you do what is called back of the envelope calculations. The good example is if somebody says I have a disk and it can

only do any random access about 10 milliseconds max, they cannot do anything better than that that itself tell you that there is no way in the system you can do more than 100 transaction per second. Assuming that the transaction CPU processing is within milliseconds, some millisecond that itself will tell you that the disk since it is the slowest part, it will limit the speed of the whole thing, and you can easily save without any hesitation without doing any complicated analysis. We can just say very easily that we have to let us say at the most say that you can handle 100 transactions.

If somebody says 100 is too slow, give me 200, then you say go and get two disks. So, again as long as the processing speed is reasonably fast that will take care of it, of course, you can go so much because finally, those systems also will start other systems also will start becoming what will next. So, what I just what to mention is that often times simple modeling is very useful in this area, because of this wide disparity in the speed. And other thing about this system is that often times because of the slowness of the system, we use caching etcetera, caching, buffering all those things are widely are used in storage systems. And sometimes it so turns out either because of the bug in the system, the caching might have to be disable or because of the work load, the caching does not work very well.

In which case what happens is suddenly you will start seeing that all the optimizations that you are seeing with respect to caching no longer work and you will find that your system is really crawling. Really, really crawling and that is because the wide disparity in the cache bits in the disk bits, if the caching somehow is off we can disk explore. And this happened quite regularly or essentially regularly it happened on and off sometimes you might have heard about there was some time back problem with Facebook, where because of some errors some important cache was turned off, because the error processing or error management systems they shut up the caching for some time. And the minute the caching is turned off what will happen is that it has to go through everything has to picked up from the disk. And since disk is so slow compared to memory that means, that it was barely able to do the processing let us say- barely even less than a percent of its older capacity.

If you are doing as slow as 1 percent of the previous capacity that means, what will happen the request are coming in as usual as before, the queues build up like anything. Once the queues build up like anything the system looks completely hang, there is no

chance of you are being able to done. Everybody is looks as though nobody is able to get any service done. And I think this also is probably might have heard about it. If you want to get a appointment for the passport, you have to get some appointments and it is a totally big lottery. You have to get it in a small window everybody is trying to access at very same time and mostly likely it is a disk limited system, I am that very clear why that is the case. So, basically you can barely get in and almost nobody is able to get through system looks completely overloaded that is basically because somehow the transactions, they are unable to do fastener. And caching that something is not useful or it is a database kind of system where by definition you are accessing randomly various parts of the disk where caching will not be useful.


So, you will find that we are hit with this kind of phenomenon. Any time you make a transition from memory to disk for example. And the reason why systems like Google etcetera work very well is because they are vey, very clear about disk part, and they ensure that the most important things are always a numeric. And but as I mentioned sometimes there are errors like Facebook support they also tried to do the same thing, but the errors prevent the caches being probably being used, because somehow the information is somehow invalid, some part of the subsystem has decided that the information the caches invalid. Once you say that it is invalid then we are forced to go to the more let us say trusted part of the data. And if it is on the disk then again your system is going to just crash, because there is no way the disk is going to able to service them. So, this kind thing happens.

So, it is useful to think about let us say what is this what is going to be the worst situation for you if you are using this kind of systems, so that if you can tell people I cannot do better than this in case some bad things happens, so that you can start putting that into your models.

(Refer Slide Time: 07:10)

### What/How to model?

- CPU utilization for processing storage requests
  - Compression, Encryption, Deduplication, QoS for mm
- I/O Completion Processing and Interrupt handling
- Network packet processing
  - QoS for packets
- Scheduling groups of processes across nodes:
  - distributed search
- Two main types
  - Operational models: limited assumptions
  - Stochastic models: mathematically tractable (eg. Queuing theory)



Again if you are thinking about storage system, there are various aspects that come into picture. Again we looked at device drivers there is some processing going on. You can interrupt somewhat not right, but in addition that is what I am I O completion processing and interrupt handling this is something which you are already aware of. And if you have lots of disks you will have lots of these things.

And since you are also doing storage across the networks, there is also network packet processing that can come in. And at a high speed networks the gigabit Ethernet etcetera, they can be pumping packets at a high rate, because they are working at gigabits speed and Ethernet typically is that 1500 bytes or 9,000 bytes what I call jumbo prints. At gigabit they are actually able to pump it at very small intervals of time - microseconds. So, if you are able to pump at microsecond levels, if you send huge numbers of them then every time you process the interrupt it might take. So, much amount of time it actually causes what is called cache pollution because it displaces the previously cached information etcetera, etcetera.

Therefore, what happens is that this interrupt handling, network-processing etcetera can create lot of disturbance in the system, and they essentially cause lot more cache misses of the CPU etcetera, and things will be taking much longer than expected. So, there is lot of open times in very high end storage systems, there is dedicated hardware for interrupt handling and network packet processing. The simplest thing of course is what is called

we already discussed what is called host bus adapters. Host bus adapters are those proxies for the CPU which will handle all this business about interrupt handling, they are the people who do just like in the networks, they are something called segmentation and reassembling SAR. There is a similar thing in storage systems which do SAR and also interrupt handling these are called host bus adapters. They are essentially used to shield the CPU from all this interrupt activity.

In addition to these things, you will notice that CPU utilization is also required for processing storage requests that is in many situations like in suppose use compression, we have to decompress it. Those encryption and you have to decrypt it, this can also be quite costly in if you done in software that is why you will see high end equipment, they will not do encryption decryption end software, they do it in hardware itself. And there is also something called deduplication. For example, the idea is that often times there are lot of copies of the same object going around that I think we already discussed I think once.

We have mail objects for example, which is I send it to I want to send a document to somebody, there is a document in my file system. But I am also sending the copy and I keep the copy of mail, I sent then there will be a copy of that same document in my sent message right now two copies are there. So, it turns out that there will be lot of copies building up because of various aspects from the application point of view. And you want to find a way of keeping only a single copy or as many as copies required to ensure that your error management is handled properly.

For example, if somebody decides that because I can suffer failures of sectors and all those things, if I keep two copies if one goes bad, other I can use it. So, you might have a minimum I just want I have exactly not more than two copies or three copies. So, whereas, if you do not do this deduplication, the number of copies can be unlimited, for example, I work with the students I give a copy to one particular paper, next year another student comes around I mail it to him also, another student comes around I mail another copy. So, I can my copies can just keep on building up in my sent messages.

So, the deduplication is the technique by which you try to reduce the let us say the number of copies on system, but that means that any time there is an object I need to be able to figure out if the object exists in the system or not now that is a non trivial object.

It something like again we are searching on a round disk essentially and you know that disk is very slow. Therefore how do you search when you have terabytes of because nowadays we have two-terabyte and three-terabyte and four-terabyte disk. How do I search given that I am trying to do something some activity?

I instantaneously should see there is a copy that I am generating any time a copy is being generated I should see if it is already there right and then say that do not mail a copy like that put a pointer to the already existing data. So, the deduplication can be very heavy with respect to processing. So, that all this kind of things that come to picture again if you are doing a distributed search for example, like what Google is doing right you might want to have some performance model for multiple nodes doing something together.

So, given this simple background, it turns out if you think about it there are various models of performance, I am just going to talk more simple varieties; normally as I mentioned what is called back of the envelope calculations are reasonably good for fairly simple storage systems, you do not need really complicated models. So, you can do this back of envelope calculations, and usually you essentially figure out what is called the bottle neck device.

To find out what is the bottle neck and that will try everything else. If something is a bottle neck nobody else can go faster than that particular bottle neck. So, you look at how fast that bottle neck device is doing something that will decide everything else. So, it is what is called bottle neck analysis. So, it turns out there are two main types of there are many more, but I am just giving you here some just for simplicity what are called operational models here they make very limited assumptions.

And then we have slightly more detailed models where you assume certain distributions. And usually you assume these distributions to be mathematically tractable. So, queuing theory is a good example of this there are many types, but I am just in this particular talk I will just mention these two. So, we will just quickly take a look at each of these things and see how they can help us.

(Refer Slide Time: 13:36)

### Operational Laws


Little's Laws: true if during observed period T,

- arrivals (a) ~ completions (c)
- ie: (a-c) small compared to c

Let J = time in system  
Mean time spent in system = J/N  
Mean # in system = J/T = J/N \* N/T = response time \* throughput  
Or:  $Q_i$  (mean # in device i) =  $X_i R_i$

Usually written as  $L = \lambda W$ :

Mean number in system = arrival rate \* mean time spent in system



A good example of an operational law is what is called Little's laws. So, usually if the arrivals and completions are exists about equal during some observed period then this Little's laws usually are quiet accurate. So, let us just see how this works. Suppose, there is a particular job and it spends J is the timing that is spent in the system. So, since we have n jobs the mean time spent in system is J by N, basically this is the time in the system there are n jobs, therefore the time for each job is J by R. Again you can see mean number in system is J by T, and you can easily write it as J into N into N into T, it is very simple this I am just doing simple algebraic manipulation. It is just you see N, N cancels therefore, it is same as the J by T, but J by N is what number of it is the time taken divided by number of jobs in the systems that is response time.

What is N by T, is the number of jobs divided by time, now this basically is the throughput. So, essentially the mean number in the system is basically response time to throughput. This is an example of an extremely simple analysis will tell you that under these kind of conditions this will better. Or you can often times written as the mean number of customers in a Q for device i the mean number the Q size are device i will be same as X i into R i, where X i is the response time and R i is the throughput. This is the sorry I made a mistake throughput is X i and response is R i.

So, throughput is what basically number of jobs per second, and response time is how long you take it. Often times this is written also as L equal to lambda W, where L is the

mean number in the system, and  $\lambda$  is the arrival rate and  $W$  is the mean time spent in system. We can say here its talking about response time here showing the arrival time basically if you are in a stable situation, reasonably the stable situation which is basically arrivals approximately completions, then response time and arrival rate basically there is some corresponding between these two. If your arrival rate and the way the jobs are departing system of highly unbalanced, then you will start seeing (Refer Time: 16:28) developments.

As long as I am not in the really pathological situation right then it turns out that there is a corresponding there is these two are essentially equivalent. So, often times they can here slightly stable kind of situations without any assumptions about distributions. This particular role is often applicable. Of course, there is something you have to be careful about when you look at the systems we have to make sure that the assumptions of some of these loss are valid luckily for us Little's law is valid in many, many different context it does not have very, very few restrictions it has got. But generally whatever I am talking about today we have to be careful about the assumptions; make sure that the assumptions are valid then many simple rules of this kind can be done.

So, basically what we had saying I can just repeat the  $Q$  size in front of device  $i$ , there could be lot of devices in the system, it is given by the throughput or the device multiplied by the response time were job at that particular device that there has to be I think. So, again you have spent time in thinking about it, some of them look a bit non intuitive, but if you think carefully and look at especially this kind of thing, we will see that it is fairly straight forward to derive.



(Refer Slide Time: 17:54)

### Poisson Arrivals

Assumes that in a small interval  $\delta$

# of arrivals:  $\lambda * \delta$

Prob of more than 1 arrival in  $\delta$ : negligible

Arrivals in nonoverlapping intervals statistically indep


Expected arrival time =  $1/\lambda$

Probability of an arrival in time  $t_0 = 1 - \exp(-\lambda t)$

Probability of no arrivals in time  $t = \exp(-\lambda t)$

Probability of  $k$  arrivals in time  $t = \exp(-\lambda t) (\lambda t)^k / k!$

Similarly, Poisson departures



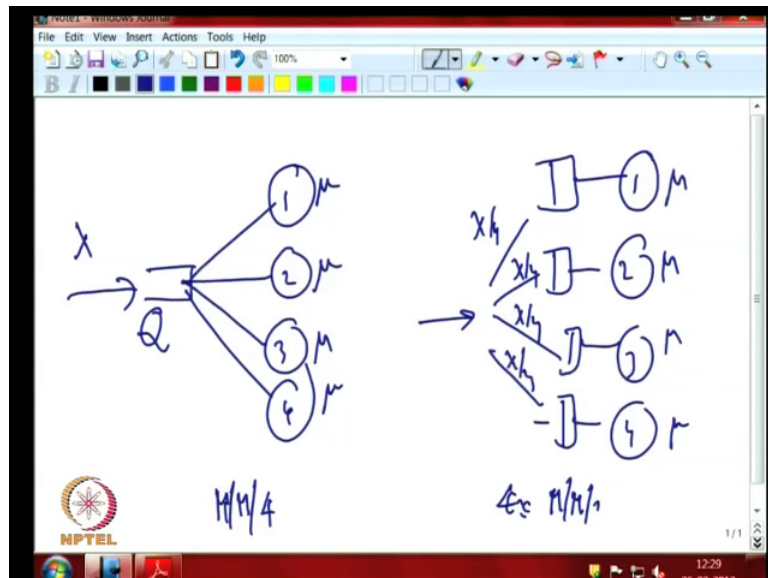
Often times we assume what are called Poisson arrivals, I will not go too much into details, but basically essentially it means that is the rate at which it is coming and every arrival is independent of the previous one, there is no correlation. And if you take a small interval, the arrival rate is lambda. So, in that small interval, lambda into lambda star delta. And if you take a small interval delta probability of more than one is arrival is negligible.

So, basically arrivals in non overlapping intervals statistically independent you take any two non overlapping intervals then the arrivals are statistically independent, there is absolutely no information flow from this interval to this interval or vice versa independent, complete independent often times called memory less. So, the expected arrival time is because if lambda is arrival rate is expected arrival time is 1 by lambda. And you can the probability of an arrival in time t given this Poisson can be derived as one minus exponent I am not go into go into it, but they basically 1 minus exponential of minus lambda t.

Therefore, no arrival is exponential minus lambda t. And you can also derive probability of k arrivals in time t is basically same as this, but lambda t to the power of k divided by the factorial k. So, I am not going to go into too much as details, but generally Poisson arrivals are assumed and similar we can also basically because we have this

independence of non overlapping intervals and that is simplifies analysis similarly you can think about departures also.

(Refer Slide Time: 19:48)



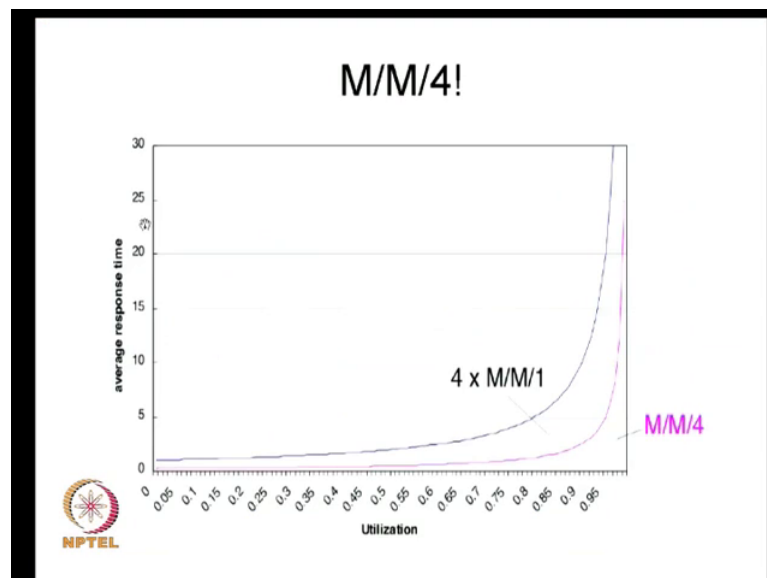
See what we are looking for is the following. Suppose, I want to analyze some of these kind of situations there is one single bit  $Q$ ; and from this  $Q$ , as CPUs here CPU 1, 2, 3, 4 and people are coming at arrival rate  $\lambda$  and their service rate is  $\mu$  mean everybody CPUs. The second thing is suppose the other possibility is I have a queue here, and what I do is I have four subsidiaries queue. And I put one crossing there. I think I can avoid this basically in some stages I think I will remove this part it is not necessary. So, somehow basically instead of  $\lambda$ , I have  $\lambda/4$  in order to say. So, this 1, 2, 3, 4, again this is doing at  $\mu$ .

Now, the question for us is which one is better, is the left one better the right one better. You will often see this in practice, for example, you go railway reservation counters in some places you will find that there is one central queue. And then from there you go to various thing there will be some let us say an electronic indicator which says as soon as I get in you get some number and then you all sit in one common area. And then you will get an electronic saying this number go there and you will forget of which one which have this and it will tell you where to go also that is this one.

The other model which is much more common because it requires less infrastructure, somebody comes in he looks which is the one which is for most likely available he goes there. The question for us, which is better this is better or this is better. Here basically the idea is that there are four independent queues and then once you get in you cannot any other queue of course, real system people do that also they go into this and decide this is moving faster and then move all those things. But let us say for this particular problem we do not do that the question which is better.

This turns out to be what is called M slash M slash 4 and this 4 into M slash M slash one I will tell you what is this one, these are two types of models or systems. And here you will see that there is a total processing power or single queue is feeding four of these things. Whereas here each of them being is separate there is essentially four units of processing power for all the queue that is why is 4, here it is one unit of power is per each one that is M 1. Sorry, there was a slide here, but somehow.

(Refer Slide Time: 23:08)



Question is which one is better, it turns out if you do it carefully this is what you get. I just want to check whether it is a. You will see that if you look at average response time and utilization if you do analysis then M M 4 will be always better than 4 into M M 1. And you can see why also basically because a in practice also you will notice that if you look at this, it is because if for some certain reason there is one job which takes too long. People are behind this particular queue, they are stuck. Whereas, if you are here the

particular jobs takes too long, you are going to be you still have a opportunities to go here, this guy taking too long. But the guy (Refer Time: 24:02) because he can still go to any of these guys, he still get a chance of t being serviced by any of these guys, whereas here you get stuck.

So, if it turns out on an average this has a implication with respect to what is the typical rating time for somebody to be served in this queue that is why this one. Normally, the idea is that your serving capacity should be as far as possible aggregated together as far as possible, do not split into pieces. But a good thing about splitting into pieces is that you get sometimes dedicated processing power. Here what is happening is that I have to share all these across all these guys, sometimes you are in a situation where you want this particular set, you want it, and once you get demand hold on to it. There was a situation that happens where this kind of thing model management also for especially for guys who have long let us say requirements for processing anyway.

So, the question for us is how do we do these things. So, I will just show you simple M M 1 and then I will not do M M 4 I will just mention the result for M M 2 and show that there is some how you can compare these things.

(Refer Slide Time: 25:12)

### M/M/1 Analysis

- Arrival rate Poisson  $\lambda$
- Service rate exponential  $\mu$
- Model as birth-death process
- Steady state ( $\lambda < \mu$ ): Probability stable that  $i$  customers in system ( $\pi(i)$ ). Given by
  - $\pi(i) \cdot \mu = \pi(i-1) \cdot \lambda, i = 1, \dots$
  - ie.  $\pi(i) = \rho \pi(i-1), i = 1, \dots$

```
graph LR; i-1((i-1)) -- lambda --> i((i)); i -- mu --> i-1
```

If you have a M M 1 system; I think I seem to a missing out some things, two or three slides are missing. Sorry. So, basically what is happening is that I have certain arrivals,

people were looking for service, and then there is some service. Now, the arrival rate is  $\lambda$  and the rate at which I am able to service is  $\mu$ . Now, what we are looking for is how many consumers, how many customers are there in system, either you have customers zero number nobody is there, 1, 2, 3,  $i - 1$  arrival or  $i + 1$  or  $n$ . So, if I am at  $i - 1$  consumers customers, if a new customers comes that is arrival, I go to this state. If there are  $i$  customers in the system and one service gets completed I go to  $i - 1$ .

Now, we can see that if the arrival rate is much bigger than the service rate then queues will build up it is unstable situation. Whereas, if the arrival rate is smaller than the service rate then there might be there will be some queue because of stochasticity, there will be some queues building up now and then they will get crushed out. So, if  $\lambda$  is less than  $\mu$  then you can get a steady state and basically you can compute a steady state follows. What I am interested is finding out the probability that I am in this particular state that is  $i - 1$  customers in the system or  $i$  or  $i + 1$  or whatever I want to you able to compute the probabilities.

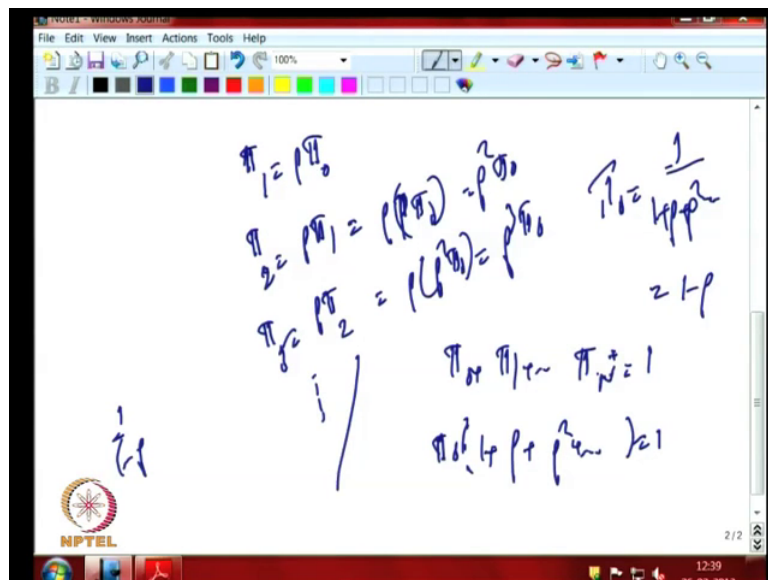
So, if I am interested what I can do is let us say that I call the probability of bringing this state  $p_i$  of  $i$  this is  $p_{i-1}$ . So, now, from here I go with at rate  $\lambda$  from here to here I come with  $\lambda$  rate  $\mu$ . So, if something stable at each point suppose I draw a line like this, if it is the situation is the system is stable, there I have to be the I am going from this radius to this radius should be constant should be the same. Therefore, they will again be the probabilities will be stationary; they will be at the same they will not change.

So, basically the probability that I am here is  $p_{i-1}$  and multiply that with  $\lambda$  that should be equal to with the number of guys, who are here or the probability of the system all right here multiplied by  $\mu$  that is basically what it is. So, basically I take if  $n$  is the number of guys,  $p_{i-1}$  is the probability of being at that one, then we  $n$  into  $p_{i-1}$  is the number of guys here this will be  $p_{i-1} \cdot n$ . So, basically we can see that this equation is has to be valid  $p_{i-1} \cdot n \cdot \lambda$  should be equal to  $n \cdot p_i \cdot \mu$  that is the number of guys here number of guys. Here the chances of anybody making transaction with  $\lambda$ , which has anybody make a transaction with this. Or you can just look at from the point of probabilities  $p_{i-1}$  going to this into  $\lambda$  is this that should be same as this part  $p_i$  and job getting finished and

that is  $\pi_i$  is equal to  $\lambda/\mu$ , which  $\rho$  this is called utilization,  $\lambda/\mu$  by  $\rho$ .

And  $\lambda/\mu$  is basically less than 1, because I am assuming  $\lambda < \mu$ .  $\lambda/\mu$  will be less than 1. So,  $\pi_i$  is equal to  $\rho$  of  $\pi_{i-1}$  and this is true for  $\pi_i$  starting  $i = 1, 2$  etcetera, is this hopefully is this clear. So, it turns out from this you can compute  $\pi_i$  equal to  $\rho$  to the power of  $i$   $\pi_0$ , how does it happen you can just unroll this. So,  $\pi_1$  is equal to  $\rho$   $\pi_0$ , then  $\pi_2$  will be  $\rho$  of  $\pi_1$  then  $\pi_3$  will be  $\rho$  of  $\rho$  into  $\pi_2$  etcetera. So, essentially you will get this.

(Refer Slide Time: 30:04)



You will get something like this  $\pi_1$  equal to  $\rho$  times  $\pi_0$ ;  $\pi_2$  will be  $\rho$  of  $\pi_1$ ;  $\pi_3$  will be something like this  $\pi_2$  etcetera. So, if you just keep substituting, we put into this for example, this will be  $\rho$  times  $\rho$  times  $\pi_0$  nearly  $\rho^2$   $\pi_0$ , this will be again same thing  $\rho$   $\pi_2$  is  $\rho^2$   $\pi_0$  equal to  $\rho^2$   $\pi_0$ , this (Refer Time: 30:45). So,  $\pi_1$  is  $\rho$   $\pi_0$ ,  $\pi_2$  is  $\rho^2$   $\pi_0$ ,  $\pi_3$  is  $\rho^3$   $\pi_0$ , but then the probability should all sum to 1, so that means, that  $\rho$  of  $\pi_0$  plus  $\rho$  of sorry let me write it. So,  $\pi_0$  plus  $\pi_1$  plus  $\pi_2$  all these things should be equal to 1. So, we can this equation equal to 1. So,  $\pi_0$  is  $\rho$   $\pi_0$  sorry let me, so it is  $\pi_0$  is common  $1 + \rho + \rho^2$  etcetera equal to 1.


So, I think we can add  $\pi$  of we assume if we do not have any limit on the number of things it can be any number. If there is a bound then we have to solve it more carefully, but if it is no bound on the queue, it will go out. So, it can go all the way. So,  $\pi_0$  will be equal to from this we can say  $\pi_0$  cannot writing what is happened,  $\pi_0$  is equal to  $1 / (1 + \rho + \rho^2 + \dots)$  this is basically  $1 / (1 - \rho)$  by what is this equal to  $1 / (1 - \rho)$ . This is basically equal to  $1 / (1 - \rho)$  did I make a mistake  $\pi_0 + \pi_1$  instead of  $\pi_0 / (1 + \rho)$ .

So, what is this  $1 + \rho + \rho^2 + \dots$  it is going to be equal to  $1 / (1 - \rho)$  right. So, it is going to be  $1 - \rho$  that is basically what you get.

(Refer Slide Time: 32:55)

### M/M/1

- $\pi(i) = \rho^i \pi(0), i = 1, \dots$
- Since sum of all  $\pi(i), i = 1, \dots$  should be 1
  - $\pi(i) = (1 - \rho) \rho^i, i = 1, \dots$
- Mean number of customers  $E[N] = \rho / (1 - \rho)$
- Expected Waiting time  $E[W] = E[N] / \mu$
- Expected service time  $E[T] = E[W] + 1/\mu = 1/(\mu - \lambda)$



And therefore, that says  $1 - \rho$  that is  $\pi_0$ . So,  $\pi_i$  of  $i$  is equal to  $1 - \rho$ ,  $\rho$  to the power  $i$ . And you can also compute mean number of customers, how will you find the mean number of customers.

(Refer Slide Time: 33:12)

The image shows a whiteboard with handwritten mathematical equations. At the top, it lists the probability mass function:  $\pi(0) \cdot 0 + \pi(1) \cdot 1 + \pi(2) \cdot 2 + \dots$ . Below this, it defines  $S = \rho(1-\rho) + 2\rho^2(1-\rho) + 3\rho^3(1-\rho) + \dots$ . Then, it shows  $\rho S = \rho^2(1-\rho) + 2\rho^3(1-\rho) + \dots$ . A horizontal line is drawn between the two equations. Below the line, the subtraction is shown:  $(1-\rho)S = \rho(1-\rho) + \rho^2(1-\rho) + \rho^3(1-\rho) + \dots$ . The NPTEL logo is visible in the bottom left corner of the whiteboard.

Again by let us say the probability of 0 guys being there is pi of 0 multiplied by 0 plus the probability of being with only one customer is one multiplied by 1, pi of 2 into 2 etcetera. And pi of 0 is what it is pi 0 into 0 is 0 that is that is gone sorry. And pi 1 is rho of pi 0 which is 1 by rho into 1 minus rho plus 2 into rho square 1 minus rho etcetera, 3 rho cube 1 minus rho etcetera. And if you calculate this, this is s. So, again if you multiply by rho, rho s equal to rho square 1 minus rho plus 2 rho square rho cube minus 1 minus rho etcetera. So, again 1 minus rho s equal to rho 1 minus rho plus this will be rho square 1 minus rho plus rho cube 1 minus rho etcetera.

Now, this can be summed up, just sum of all these things because this is basically a geometric progression and you can compute s from here. It is very straight forward manipulation. If you compute that you will find that to the mean number of customers is rho 1 minus rho, rho 1 minus rho. We can see that if rho becomes close to one the mean number of customers starts going to infinity. So, rho should never be close to 1. One of the most important things that you have to do in a system design is ensure that your utilization, rho is utilization is not close to one it should be closer to 60 percent or 75 percent do not ever try to get to 90 percent etcetera, you will start seeing b queues development at that time.

So, the expected waiting time is basically this is the number of customers and this is the rate at which you are processing them the service rate. So, basically E of N by mu if you



talk about service time it will be not only the waiting time plus also your own service time that is  $E[W] + 1/\mu$ . This is a waiting time because other people are already in front of you and this is your own service time. So,  $1/(\mu - \lambda)$  again you can see that if  $\mu$  and  $\lambda$  are very close and  $\mu$  being greater than  $\lambda$  we are going to have very high service time. So, you should again try to ensure that the service rate is much better than the arrival rate.

So, basic thing is that the reason why queue is developed at all in first place is because even if you are in this situation where service rate is higher than arrival rate is because of the stochasticity in the system and some queues will build up here and there, but they will get flushed out fairly soon. And average will be in a stable situation, but you can have imbalances here and there now and then.

(Refer Slide Time: 36:31)

### M/M/1 vs M/M/2

⊙

Response time for separate Qs (M/M/1) =  $1/(\mu - \lambda)$


- with  $\rho = (\lambda/2)/\mu$ ,
- $E[W] = 2/(2\mu - \lambda) = (4\mu + 2\lambda)/(4\mu^2 - \lambda^2)$

Response time for combined Qs (M/M/2):

$E[N] = 2\rho/(1 - \rho^2)$  where  $\rho = \lambda/2\mu$

$E[W] = E[N]/\lambda$  (Little's Law) =  $4\mu/(4\mu^2 - \lambda^2)$

Less than that for separate Qs !



So, if it turns out if you do this M M 1 that is how we did M M 1 here. So, now, if you take the response time for separate queues, this is basically this one, expected service time is  $1/(\mu - \lambda)$  that is what I am putting here it is this is as same as that  $1/(\mu - \lambda)$  by  $\mu - \lambda$  into  $\rho$  is basically  $\lambda$ . Now, if  $\rho$  is equal to now situation right now is what is  $\rho$  for the separate queues, if you remember this remember that there is for separate queues right we have  $\lambda$  by here I set 4, but actually I am doing M M 2 there. So, it is  $\lambda$  by 2.

So, the actual rate at which things are coming is  $\lambda^2$  for  $M/M/2$ . These  $M/M$  these  $M/M/4$ , therefore, I divided by 4 here to make sure that the rate same for everybody, but if I am using  $M/M/2$  then  $\lambda^2$  will be there. So, we will see that  $\rho$  will be same as the  $\mu \lambda$  by 2 which is half of the other configuration divided by  $\mu$ , because if the service rate is still the same the same CPUs are there. So,  $E$  of  $W$  is basically the same as this,  $1/\mu - \rho$  we substitute  $\mu - \rho$  of this one you will get this  $2/2\mu - \lambda$ .

And then if you  $\mu$  just for comparison with the next one I am coming I will multiply this with  $2\mu - \lambda$  on top and bottom that is the. So, I will get  $4\mu$  s. Basically, I am multiplying bottom by  $2\mu + \lambda$   $2\mu + \lambda$  and both numerical denominator, so that if I multiply the top by  $2\mu + \lambda$  the two times there will be this. And bottom will come  $2\mu - \lambda$  into  $2\mu + \lambda$  that will become  $4\mu^2 - \lambda^2$ .

Now, I am not going into this  $M/M/2$ , but it turns out you can do something similar  $E$  of  $n$  will turns out to be this much, and  $E$  of  $w$  going to be this it turns out that will be  $4\mu$  times  $4\mu^2 - \lambda^2$ . We will see that the waiting time four separate queues is similar to this  $4\mu$ , but there is  $2\lambda$  is there here, but it is not there. Here the four same thing everything  $4\mu^2$  everything is same. So, we can say that this is it has to be in every single possible case this has to be bigger than this unless  $\lambda$  is 0 of course,  $\lambda = 0$  you do not have anything to talk about.

So, what this is showing is that if you do this kind of analysis you will be able to say that  $M/M/2$  is better than  $M/M/1/2$  of see  $M/M/2$  is better than 2 of  $M/M/1$  that is what it is showing. Similarly in the  $x$  the what I have said before in 4 of  $M/M/1$  is not as good as  $M/M/4$  what it is it saying it basically saying that put all your processing power in a single large capacity kind of system, do not split it up into pieces. If you split them into pieces, basically what happens is that sometimes there are short falls, sometimes there are over this thing and you cannot manage as well. Whereas, if you have very large capacity then it can processing so fast therefore, even if there is momentary imbalance it can cope with it very fast. Whereas if you split up the processing power when there is sudden momentary hence it will take longer time to fix it.

So, this is basically the intuition behind why you will see that you want to combine have a larger kind of memories larger kinds of let us say CPUs etcetera, which you want to put in front of processing and that large amount usually takes care of those momentary imbalances quiet well compared to split up the resources.

(Refer Slide Time: 40:30)

### Utilization Law

Utilization = Busy Time/T = (completions/T) \* (Busy Time/completions) = throughput \* service time


Or, at device i,  $U_i = X_i S_i$

Forced Flow Law: Device throughput of device i = completions(i)/T = completions(i)/completions of jobs \* completions of jobs/T = visit ratio \* system throughput

Or,  $X_i = X V_i$

Utilization of device  $U_i = \text{throughput}_i * \text{service time}_i = \text{visit ratio}_i * \text{system throughput} * \text{service time}_i = \text{system throughput} * \text{total service demand on device}_i$

Or,  $U_i = X D_i$



We will just also a just look at some just like Little's laws; there are few other types of laws which are quiet useful. These are called utilization laws. Again these are we can see they can be done with basically simple algebra here. What is utilization it is nothing more than busy time by time - the time taken. So, how much of the time you are busy by the total time. So, what is this? We can as well write it as completions by time into busy time the completion. Basically I am introducing these completions, we can see this completion and this completion cancel, it is same nothing more than busy time by T itself.

But if you look at completions by time what is that it is a throughput, how many things I complete in the particular time that is throughput. And what is busy time the completions it is basically service time there I am busy so much and I took this number of completion at it that means, these I am busy for every single thing I am busy that is a amount of service time. So, and this is true for any device in the system, it could be a disk whatever. So, at device i utilization is throughput i into service time this is thing you can derive.

Similarly, here something called forced flow law. What is it saying, essentially it is saying that if the throughput of the whole system is capital X system throughput that drives rest of the system. See, there is a bottle neck device that determines the throughput of the whole system, because that bottle like device drives the whole throughput of the whole system. Now, everybody is going to be affected by that particular rate at which the bottle neck device is driving the whole system. So, everybody else has to be corresponding to that that is what this is say.

Again let us look at it what is forced flow law device throughput of device i is completions of i by T at some particular device I have do so many completions, so many times I do something and I taking so much time T. Again you can write it as completions of i divided by completions of jobs into completions of jobs by T same like this completion of jobs can be canceled. This is a by b into b by c that is what I am doing completions of i divided by of jobs. So, this is cancels this into cancel.

Basically, actually completions of i by T same as this, but we say completion of I by completions of jobs what is it this is basically the number of time I am coming to this device to get my job done it is basically visit ratio. See, this is the number times what is completions of jobs, how many time I finish a job that completion raises how many times I finished doing some job at I did something for a job, how many times I have done it at device i. So, this is basically the visit ratio that is how many times if I want to get a job done how many times I go to a particular device that is what this visit ratios V.

For example, if you are going to a government office and you have to talk to some person ten times before your job is done that visit ratio is 10. So, again what is completions jobs by T that is nothing more than system throughput so that is why this what is called forced flow law basically your device throughput or device i given by the visit ratio multiplied by the throughput. So, essentially what it saying is that the system throughput forces how things happen at a each particular device with respect to its throughput. If your system throughput is i then we are not a bottle neck device then your thing will be correspond to that same system throughput how many times you are.

For example, if I increase the some throughput of particular system correspondingly everybody also in the system also has to somehow correspondingly increase also there ability to process things that is basically version. We can also have utilization of device,

now utilization of a device is basically this one. What we decide utilization is what throughput into service time that is what we said now but throughput is what we decided here is  $X V_i$  that basically throughput is same as visit ratio in system throughput. So, it is basically visit ratio into system throughput into service time. So, I am just comparing these two equations, first I start with these throughput into service time, but throughput is given by this, so I have put visit ratio in system throughput here.

Now, I can see this I can take out system throughput visit ratio and the service time I multiply because this visit ratio into system throughput service time. But if I multiply visit ratio  $i$  into service time that is basically the total service demand and service  $i$ , where is I visit a device visit ratio number of times every time I ask for system amount of time at the device. So, visit ratio  $i$  into service time is basically total service demand on device  $i$ . So, the utilization of defined is decided by the total demand on the device multiplied by the throughput of the system itself  $X$ . So, these are three types of utilization laws.

So, we have Little's laws and we have this things and this are usually similar to the back of the envelope calculations. And these are slightly better than one back of envelope calculations sometimes can create they might not work out as well as some of these also. Then in some situations where you will find this kind of rules quiet useful and storage systems typically whenever the things actually if there are bottlenecks they can drive this.


(Refer Slide Time: 46:34)

### Example (from Jain'91)

Each prog requires 5 secs of CPU time & 80 I/O reqs to disk A and 100 I/O reqs to disk B. Disk A takes 50ms; disk B takes 30ms. Total of 17 terminals with disk A's thruput 15.7 I/O reqs/s. Ave think time: 18s. What is the system thruput? Device utilization?

$$D_{\text{diskA}} = V_A S_A = 80 * 1/20 = 4s$$
$$D_{\text{diskB}} = V_B S_B = 100 * 3/100 = 3s$$
$$X_A = 15.7 = X V_A \Rightarrow X = 15.7/80$$
$$D_{\text{cpu}} = 5s; V_{\text{cpu}} = V_A + V_B + 1 = 181;$$
$$U_{\text{cpu}} = X D_{\text{cpu}} = (15.7/80) * 5 = 98\%; U_A = X D_A; U_B = X D_B$$

If  $Q_{\text{cpu}} = 8.88, Q_A = 3.2; Q_B = 1.4$ , what is response time?

 Use Little's law:  $R_i = Q_i / X_i$

And let us take an example. Suppose there is a program, which is requires five seconds of CPU time and for getting that program completely done it requires 80 hour requires to disk A, and 100 hour requires to disk B that it has to go. So, many time it has to go disk A to get something done. Disk A takes 50 milliseconds, disk B takes 30 milliseconds there are lot of terminals like your you know access terminals with so many with disks a throughput is 15 points whenever requires I have taken the example from a book.

And average think time is 18 seconds because people are sitting in front of a terminal before the ratio requires they typically take a 18 for their make a request. The question is what is the system throughput? So, basically what we say from this program from this particular problem he is saying that we have to visit disk A 80 times.

So, visit ratio for disk A is 80 and visit ratio for disk B is 100 what is the total demand on the disk A. What is the demand we are looked at demand right demand is basically we have want to find the total service of on demand you want to find the total service demand on device side that is basically one. So, basically visit a multiply by service time. What is the service time for disk A, 50 milliseconds 1 by 20 milliseconds, so that total demand for disk A is 4 seconds. Similarly total demand for disk B is going to be the number of times it visit the disk multiplied by the amount of service time we required on disk B now it is 30 milliseconds is 3 by 100 that is 3 seconds.

Now, you already know that disks a s throughput has been given 15.7 I requires per second somebody observed it that it is being 15.7 I O requests per second. So, that is basically the system throughput  $X$  of  $A$  that is same as because of the forced flow law right we say  $X$  of  $I$   $X$  of  $I$  equal to  $X$  of  $X$  into  $V$  i. So,  $X V A$ , so now, we know  $v$  a because that is a number of times you go to disk. So, we can see that the system throughput is basically  $X$  equal to from this 15.7 by eighty that is num that is a throughput of the whole system. So, essentially it is doing about it is taking about this amount of basically you are able to achieve a throughput given by this.

Similarly, if you look at the CPU, it says it takes 5 seconds of CPU time. Now, what is it visits the CPU, the basic idea is that the assumption is that the program requires 80 hour requires and 100 hour requires; that means, it has to do something then go to disk and do something go to disk for  $A$ ,  $B$ ,  $C$  etcetera that means, it has to keep visiting those things. So, again it has to once it has finish the disk activity has to go back to CPU again all. So, it is going to do number of visits the CPU will be given by number of times it goes into disk and come back to CPU again, it has a very scheduled again back to the CPU. So, the number of visits is going to be number of times it goes to disk  $A$ , number of times it goes to disk  $B$  and the last time it does it goes out, so 181.

Now you will notice that utilization of CPU is going to be given by the throughput of the system multiplied by the total demand on the system again it comes back from the same thing, same thing utilization of device because the CPU is also like a device. So, you know already  $X$  and you know that demand on the CPU which is 5 seconds. So, therefore, you can see utilization 98 percent that is; what is the system throughput we are calculating device utilization. We can also compute a device addition of disk  $A$  and disk  $B$  given that  $X$  is there and you know the total demand also. So, you can make this kind of calculations you can see from here that you have been able to figure out that the utilization of the CPU is 98 percent.

Now, this is very high that means, that if you are really trying to improve the performance of the system, you better find a way of putting a much faster CPU, because at 98 percent you remember the utilization is what is utilization the disk sorry because I am utilization is this point is basically  $\lambda$  by  $\mu$ . So, you are in a situation where your response time is going to be proportional to  $1$  by  $1$  minus  $\rho$  since it is 98 percent; that means, you are going to be if the range fluctuations. It can hit you very bad if things

are nicely coming with reasonable things people work out that, but suddenly the fluctuation there will be a problem. So, I will skip this part, but basically the thing is you can use measuring certain parameters in the system, and looking at the way the requests are flowing in the system, you can actually compute some of these details. But most importantly you have to remember this utilization is forced flow law and (Refer Time: 51:52).

(Refer Slide Time: 51:52)

**General Response Time Law:**  $Q = Q_1 + \dots + Q_n$ ;  $Q_i = X_i R_i$   
 (Little's Law);  $Q = XR = \sum Q_i = \sum X_i R_i$   $R = \sum V_i R_i$


**Interactive Response Time Law:** Z (think time) + response (R).  
 In time T,  $T/(Z+R)$  requests.

With N users, system throughput  $X = (N \cdot T / (Z+R)) / T = N / (Z+R)$   
 $\Rightarrow R = N/X - Z$

**Bottleneck Analysis:**  $X(N) \leq \min(1/D_{\max}, N/(D+Z))$  where  $D = \sum(D_i)$  (the sum of total service demands on all devices)

$U_{\text{bottleneck device}} = X D_{\max} \leq 1 \Rightarrow X \leq 1/D_{\max}$

With 1 job:  $R(1) = D_1 + \dots + D_M = D$   
 $R(N) \geq R(1) = D$ ;  $X(N) = N / (R(N) + Z) \leq N / D + Z$



There is a similar that we can do I think I will skip this part, I will talk about bottle neck analysis ok we will do this also. So, what is basically this is the number of sorry Q is the number of things in the system. So, basically at device one there are Q 1 guys waiting Q, Q 2 guys waiting at device 2, Q n guys the total number of guys in the system is Q. Now, we from the Little's law, the number of guys waiting in the system is Q. So, this that is same as throughput in the system multiply response time. And this is same as all the guys getting sigma Q 1, now this is per device. So, you can essentially per device you know what Q i is right it is basically X i into R i multiplied by R.

So, again this is we are using Little's law again one more time. So, basically we look at R into sigma sorry this R into X i becomes the number of digits. So, essentially you will find that Q is equal to sigma of V i R i. What it is saying is that if you look at the number of visit ratios visits or a particular device multiplied by the response the sum of all this



things that will tell you how did the how many people are there in the system. I am just using Little's law to derive this and we are using this  $Q_i = X_i R_i$ .

Again this basically we are using Little's law one more time. And there are some systems for which you have interactive systems that think time is given. And basically if think time is  $Z$  on the responses  $R$  then time  $T$  we can only make  $T$  by  $Z$  plus  $r$  requests because you have to add your think time and response and that is a number of times you can go back and ask for some service this is a maximum we can do. Again with  $n$  users what happens now the  $N$  of this guys, they can all make  $N$  into  $T$  and divided by  $T$  is because that is the total time we are thinking about. So, essentially this is the system throughput is given by this which is  $N$  by  $Z$  plus  $R$  is equal to  $X$  again response time is, now therefore, given by  $N$  by  $X$  minus  $Z$ . So, this is one if you have interactive system this will be this one (Refer Time: 54:34).

Then if you want to do bottle neck analysis basically if you have a bottleneck device the throughput of the system is decided by the bottle neck device, and then that has got a maximum demand of service. Basically that bottle neck device has got a lot visits or whatever it is and that is causing a system to be driving a system to what it can do. So, but the utilization is always less than or equal to 1 so that is why  $X$  should be always less than 1 by  $D_{max}$ . Since, there is a bottleneck device it is going to satisfy this property we already seen this one. So, the guy with the largest demand is the thing whose is going to set the stage for everybody also. So, since utilization can always cannot be more than one therefore,  $X$  has to be less than or equal to 1 by  $D_{max}$ .

Now, that is why your throughput of the system will definitely be equal to or less than this guy. There is one more parameter which comes on interactive thing basically if you take only one job, there are no queues right because only one job, therefore the service demand is basically the demand of at each station, so that is  $D_1$  plus etcetera to  $D$ . Whereas, if I have  $n$  jobs, the response time has to be certainly more than response time for a single time that is  $D$  which is  $D$ . So,  $X$  of  $N$  is basically  $N$  by using this particular thing  $N$  by this  $R$  of  $N$  plus  $Z$  this and  $R$  of  $N$  is basically greater than  $D$ , so is basically less than equal to  $N$  by  $D$  plus  $Z$  that is why you get this mean on this.

Therefore if you do this bottleneck analysis we can say that you look at the bottleneck device. And look at this demand total demand how many times some guys asking for

service how many times he is visiting that guy see how much time it takes that is the whole demand. And that guy will essentially set the stage or if there is from the interactive response time this guy will also can be there limit between the two things they will actually limit what is possible in the system.

So, you do some simple analysis of this kind and you can essentially get some very good insight into the system luckily. You can see this nothing complicate here everything is very simple algebra extremely simple algebra. And this is good for storage kind of systems because there is one guy who is really slow and that guy drives a everything else that is a bottle in some situation we might call the bottle neck guy. The minute everybody support the same speed then you need more complicated models. But, so far storage system have been slow that is why you can look back of the envelop calculations.

So, I just want to conclude here by saying that do simple models probably you get some insight, take that insight does not work with what you see then you try to come with something more always start simple thing first. That is all I have to say.