



**THINKING OF PERFORMANCE**  
(FORECASTING EXERCISES)

# WHOAMI

10 years experience with Oracle Database

Oracle Database & Oracle RAC fanatic

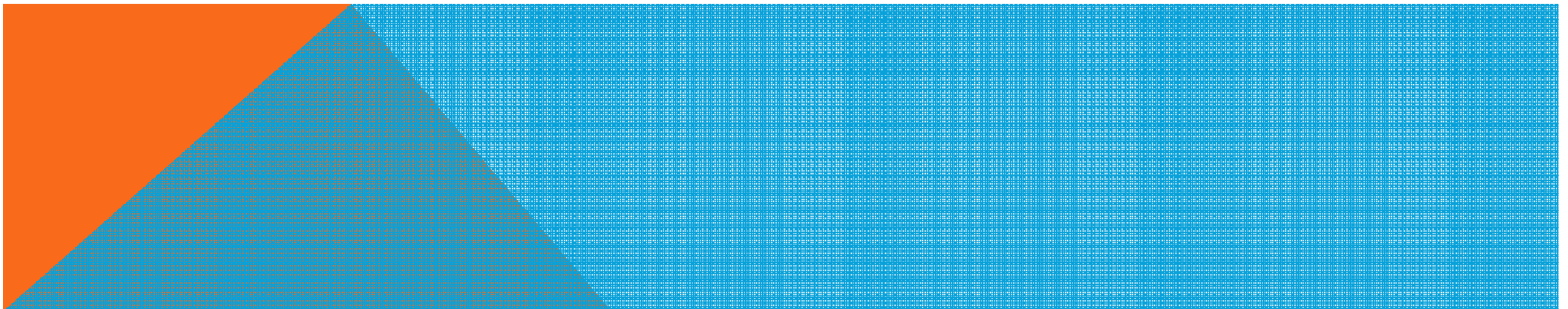
DBA at MoneyBookers

Oracle Certified Master

OCP DBA 9i/10g

OCE RAC

OCE SQL



# ABOUT THIS PRESENTATION

I will try to show some concepts and a model of thinking about performance

- What the is performance

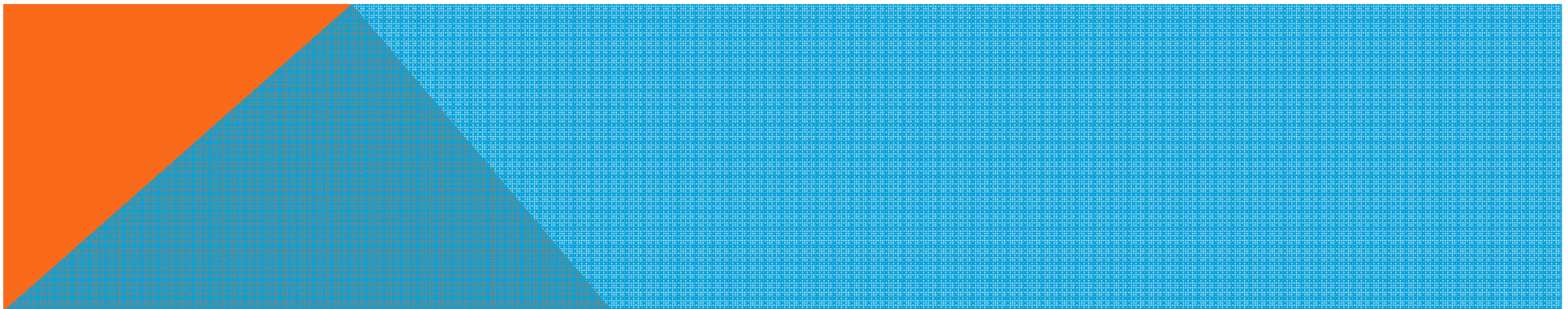
- What is response time

I will introduce you to some basics of the queuing theory

I will show some complex formulas, but only to show that it's all about math - there are formulas indeed. I will not explain the formulas in detail

I will not give oracle database specific things – no SQLs, no hidden v\$... or x\$... tricks

- Anyway, I will talk from a viewpoint of Oracle DBA



# WHAT IS FORECASTING

There is some real answer to the following questions:

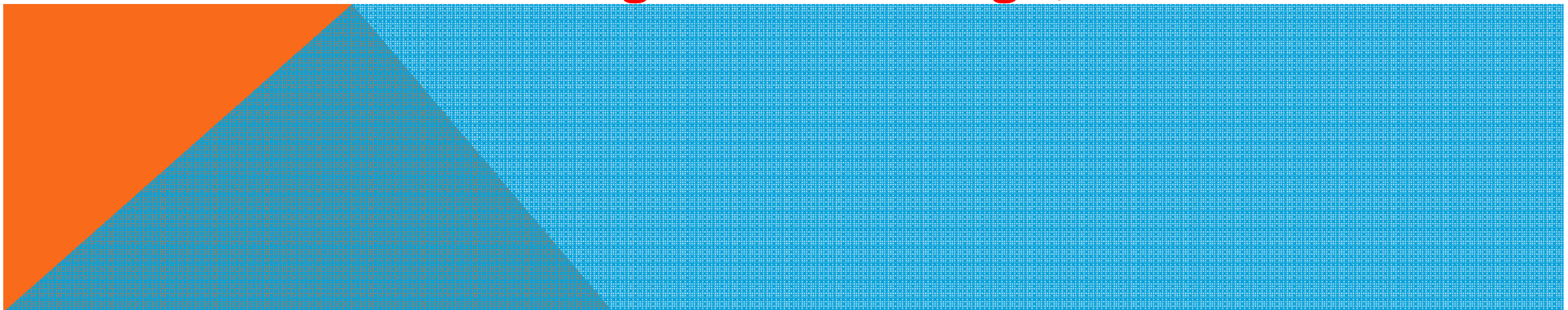
How many CPUs I need if I want to run one query from 100 users and need 1.5 seconds response time for 95% of them?

I have 800 users and the CPUs are loaded on 70%. How slower the system will get if I add 200 more users?

How much faster will our system get if we replace the current CPUs with 30% faster ones?

What is better for my workload: 4 fast CPUs or 8 slower CPUs?

**Performance forecasting is not black magic, it's science!**





# **PART 0**

STANDING ON THE SHOULDERS OF GIANTS

# STANDING ON THE SHOULDERS OF GIANTS

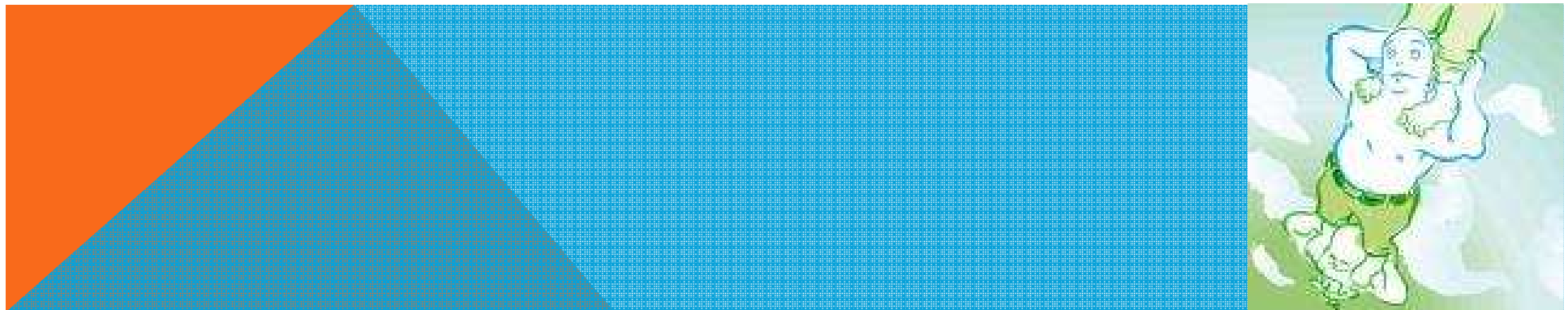
This presentation is highly influenced by the books:

*“Optimizing Oracle Performance”* by Cary Millsap

*“Forecasting Oracle Performance”* by Craig  
Shallahamer.

I will try to present some interesting things I found there.

All the formulas are researched by Cary and Craig in  
some tougher books.

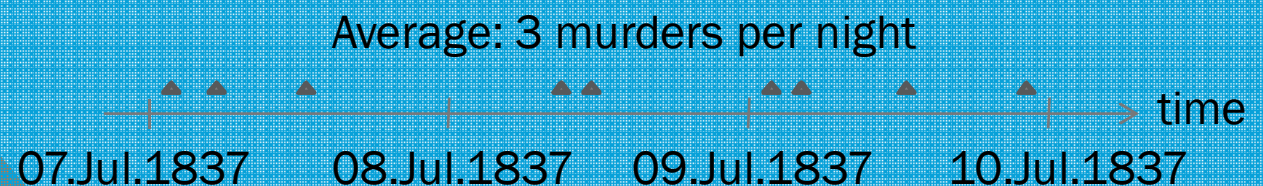


# SIMÉON-DENIS POISSON

Siméon-Denis Poisson (1781 – 1840), was a French mathematician, geometer, and physicist.



In 1838 he published his work "*Recherches sur la probabilité des jugements en matières criminelles et matière civile*" ("Research on the Probability of Judgments in Criminal and Civil Matters"). The work focused on certain random variables  $N$  that count, among other things, the number of discrete occurrences (sometimes called "arrivals") that take place during a time-interval of given length.



# AGNER ERLANG

Erlang worked for *Copenhagen Telephone Company (CTC)*.

There he was presented with the classic problem of determining how many circuits were needed to provide an acceptable telephone service. His thinking went further by finding how many telephone operators were needed to handle a given volume of calls. He developed his theory of telephone traffic over several years. His significant publications include:



In 1909 - "The Theory of Probabilities and Telephone Conversations" - which proves that the Poisson distribution applies to random telephone traffic.

In 1917 - "Solution of some Problems in the Theory of Probabilities of Significance in Automatic Telephone Exchanges" - which contains his classic formulae for loss and waiting time.

*from Wikipedia, The Free Encyclopedia*





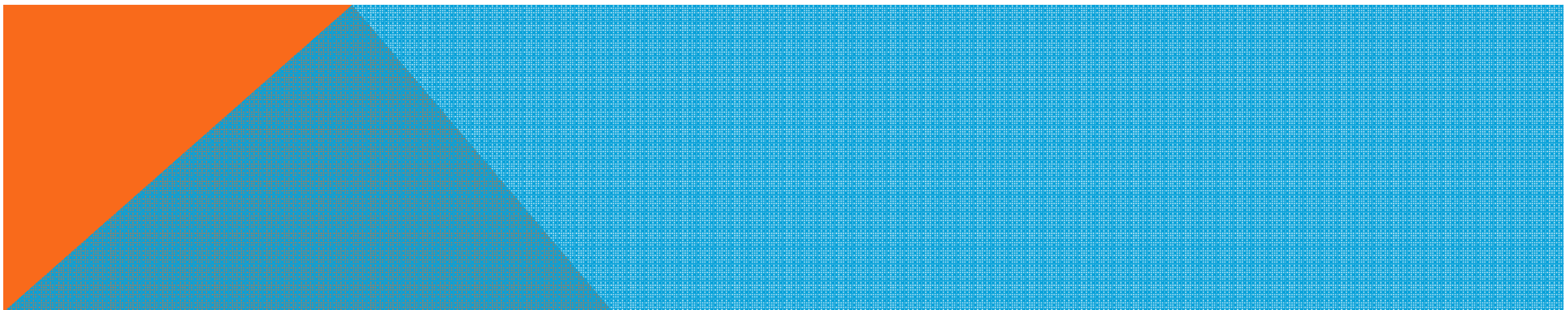
**PART 1**  
WHAT IS PERFORMANCE

# VIEWPOINTS TO PERFORMANCE

Objective: measured in transactions per minute, microseconds per call and so on

Subjective: the end-user's perception about the performance; cannot be measured

The performance analyst should work on both viewpoints



# TYPES OF PERFORMANCE

## Response time

Aims at getting one single task done as quickly as possible

Helps for better subjective performance

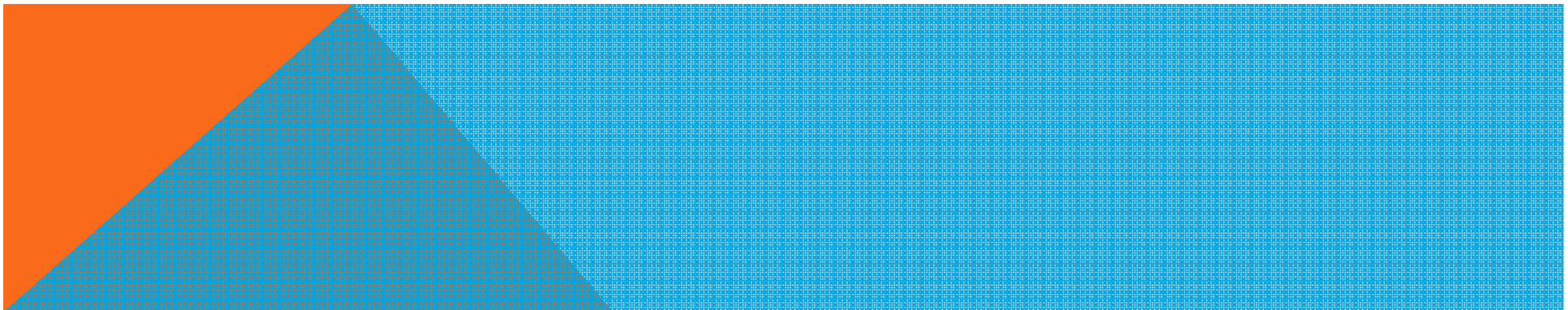
*Think of: FIRST\_ROWS*

## Throughput

Aims at doing most things for a relatively long period of time

Especially good for OLTP

*Think of: ALL\_ROWS*

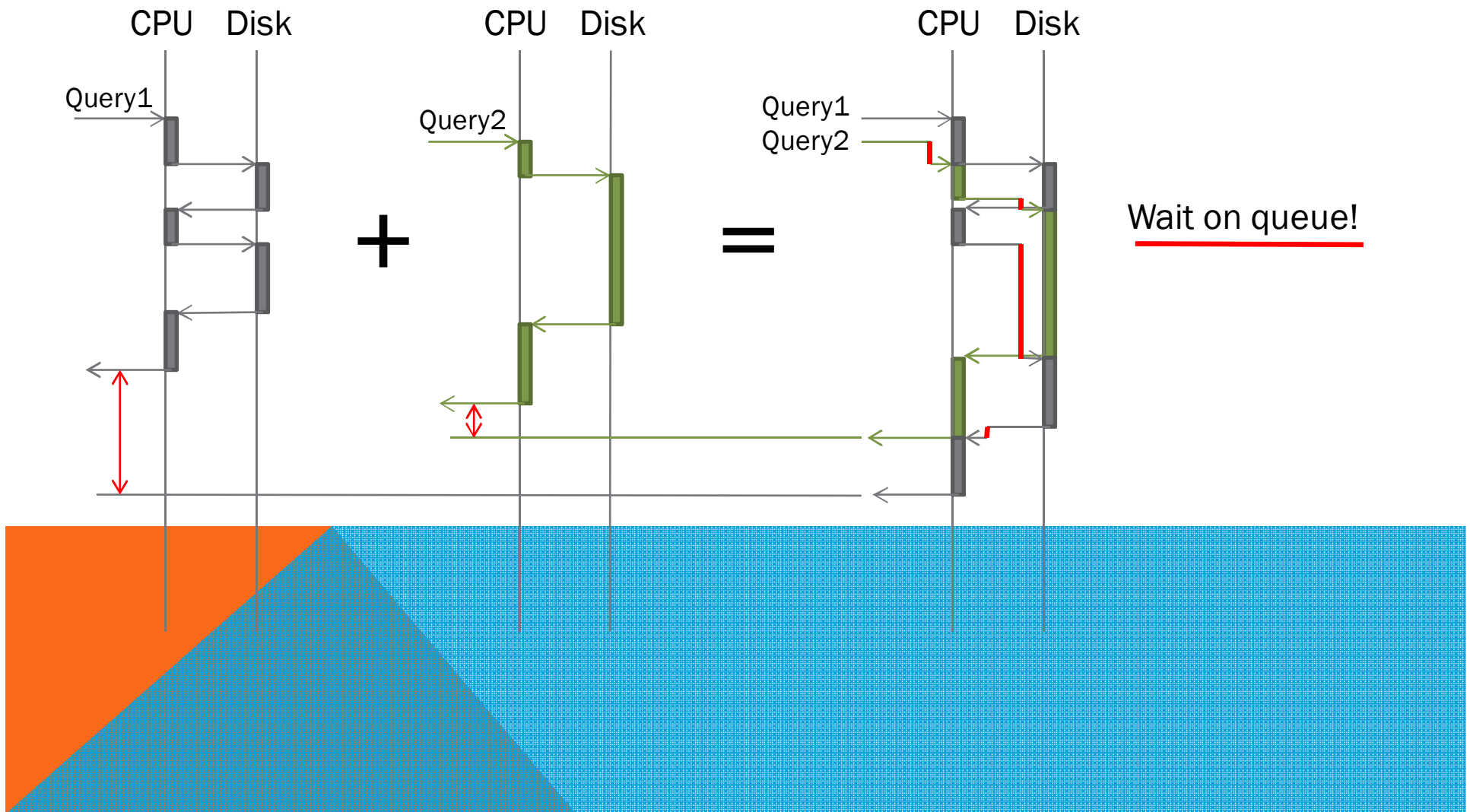




# **PART 2**

INTRODUCTION TO THE QUEUING THEORY

# WHAT IS QUEUING DELAY



# QUEUING THEORY

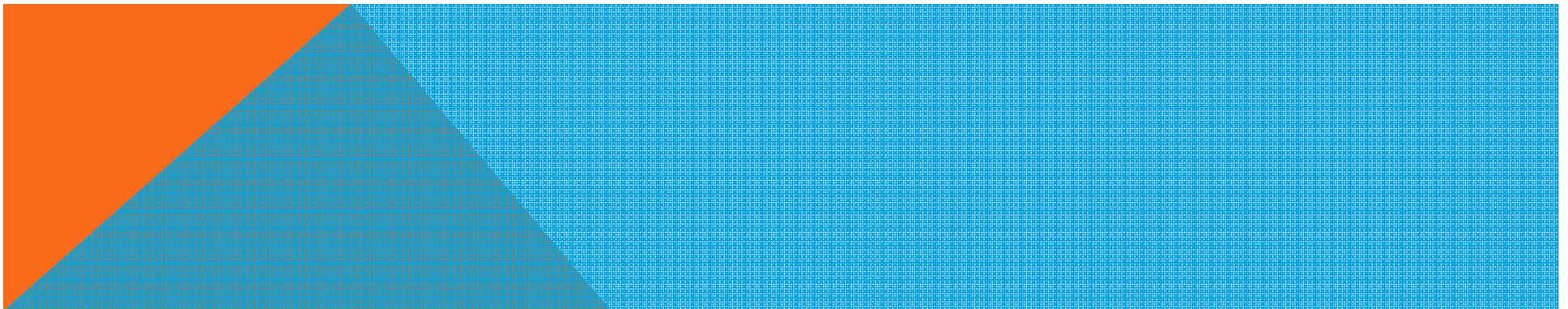
$$R = S + W$$

or

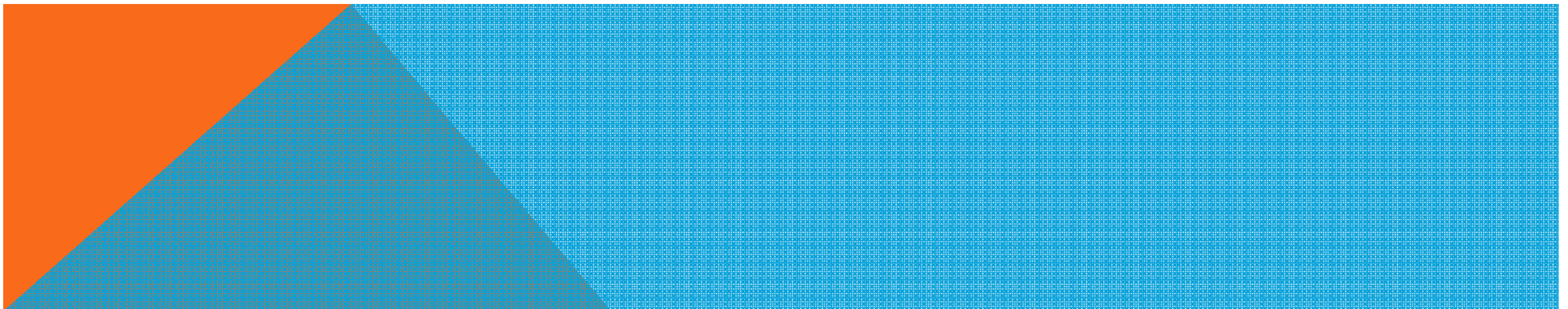
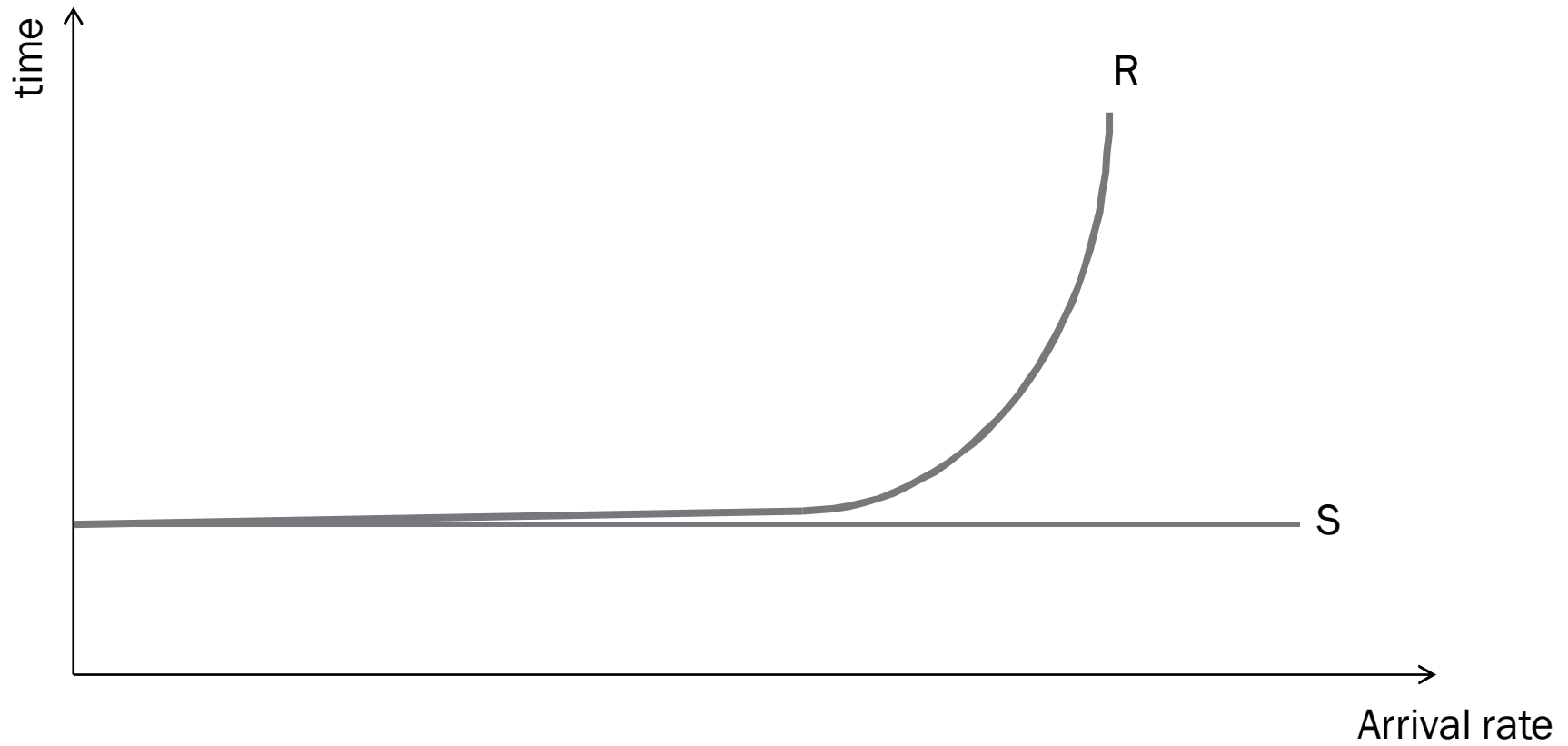
$$R_t = S_t + Q_t$$

or

Response Time = Service Time + Queuing Time



# THE HOCKEY STICK



# QUEUING THEORY TERMS

R – response time

S – service time

W – queuing delay

A - number of arrivals that come into the system

C - number of completed requests

T – time period

$\lambda = A/T$  - arrival rate (*e.g. 400 transactions per minute*)

$X = C/T$  – completion rate, throughput

$\tau = T/A = 1/\lambda$  - average duration between arrivals

Example:

T=2 seconds we have

A=4 requests arrived. In that case

$\lambda=2$  requests per second (average), and

$\tau = 0.5$  seconds per request (average)



## QUEUING THEORY TERMS (CNTD..)

B – busy time; the amount of time that service channels spent fulfilling service requests

$U = B/T$  – total system utilization for a period of time (e.g. 0.69 which is 69 %)

m – number of service channels (e.g. 8 CPUs).

When  $m > 1$ , U may be greater than 1

$\rho = U/m$  – average utilization per channel

S = B/C – service time; amount of time that a service channel spends busy per completion

$\mu = C/B = 1/S$  - service rate; the number of requests that a single service channel can complete per time unit

	Snap Id	Snap Time
Begin Snap:	55760	2-Anp. - 09 13:30:49
End Snap:	55761	2-Anp. - 09 14:00:50
Elapsed:		30.01 (mins)
DB Time:		215.54 (mins)

Event	Waits	Time(s)	Av
CPU time		6,641	
db file sequential read	792 112	3 95111	



# PART 3

FORECASTING PERFORMANCE

# QUEUING THEORY

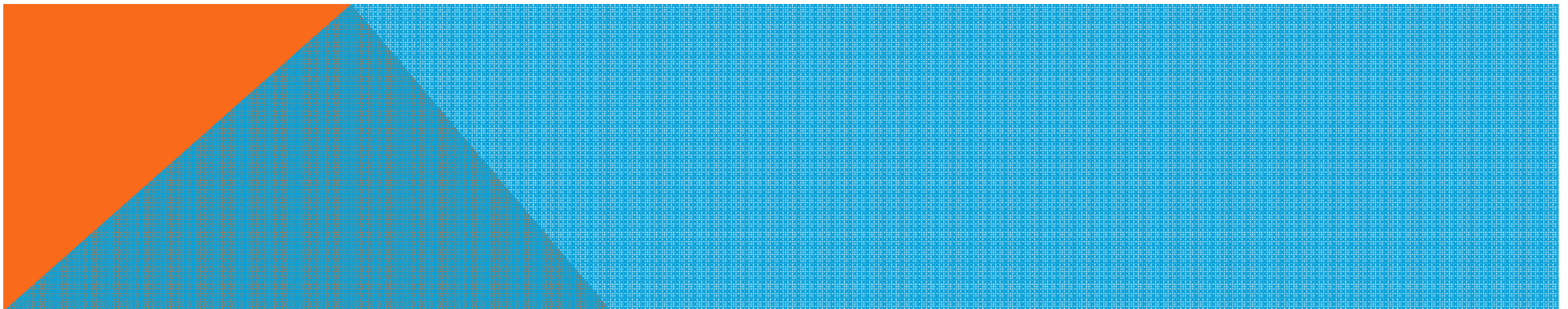
$$R = S + W$$

or

$$R_t = S_t + Q_t$$

or

Response Time = Service Time + Queuing Time



# CAN WE PREDICT THE RESPONSE TIME?

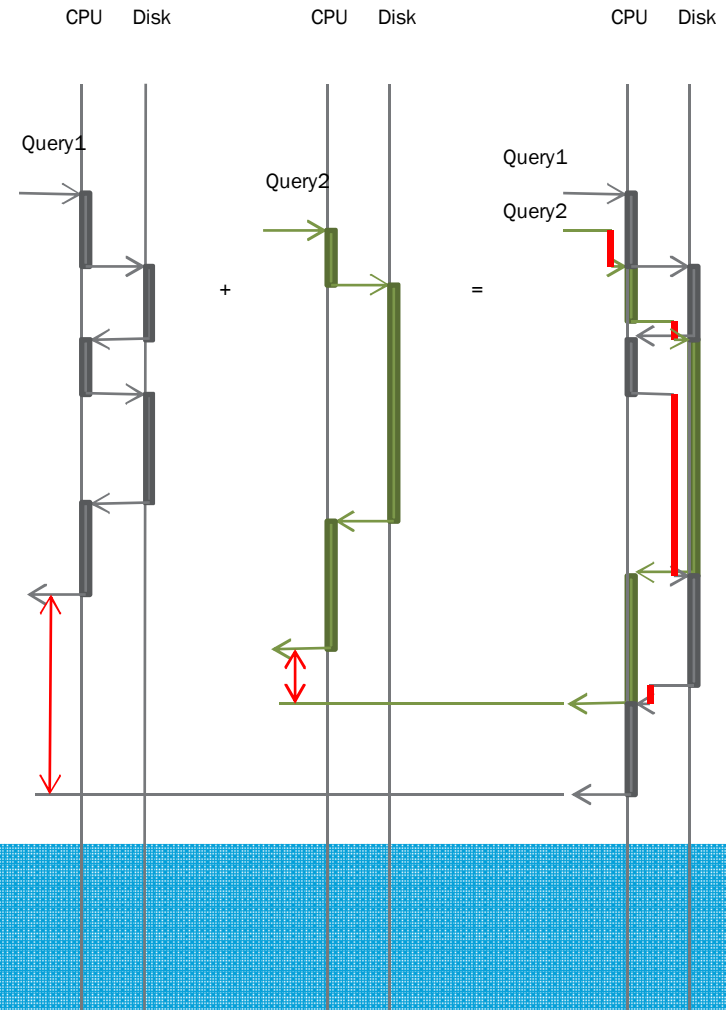
S (service time) can be measured

W (queuing time) is predictable.

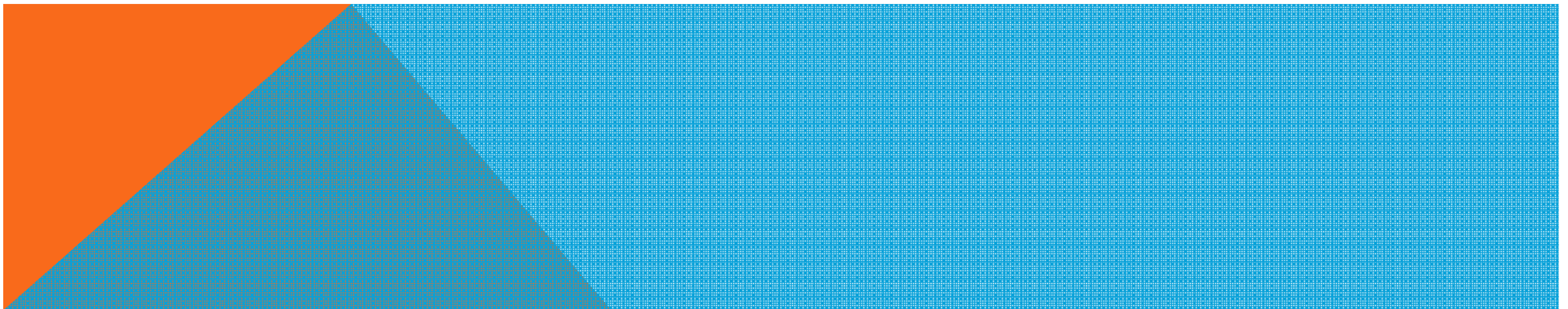
Just the math is more advanced

To find the waits (this is, the overlaps) on a resource we need to know:

- How often a new job unit arrives in the system ( $\lambda$  - *arrival rate*)
- How much time we spend completing every job unit ( $S$  - *service time*)



But those things look random!  
How can we calculate something?

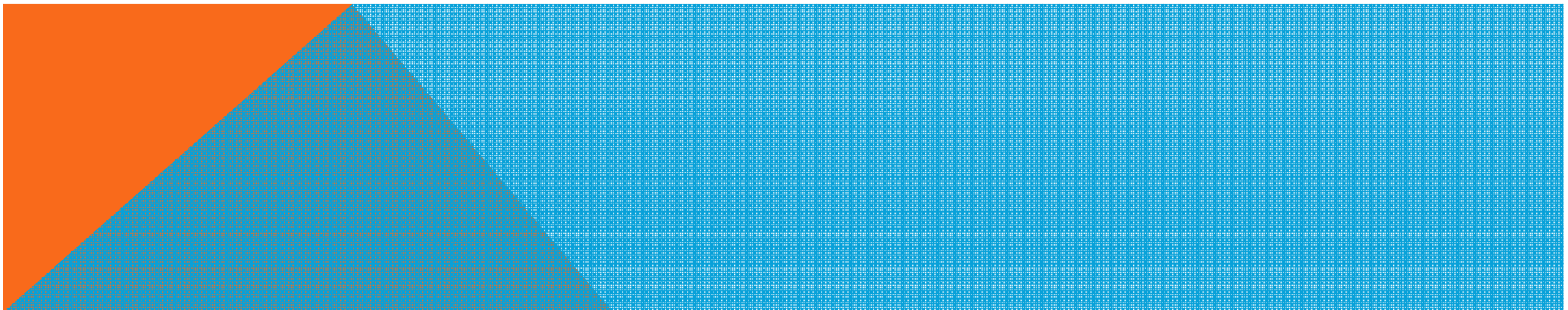


# THE POISSON DISTRIBUTION

In probability theory, the Poisson distribution expresses the *probability of a number of events occurring in a fixed period of time* if these events occur with a *known average rate* and *independently of the time* since the last event.



Average: 3 customers per second



# POISON PROCESSES

The number of soldiers killed by horse-kicks  
each year in each corps in the Prussian cavalry

The number of raindrops falling over an area

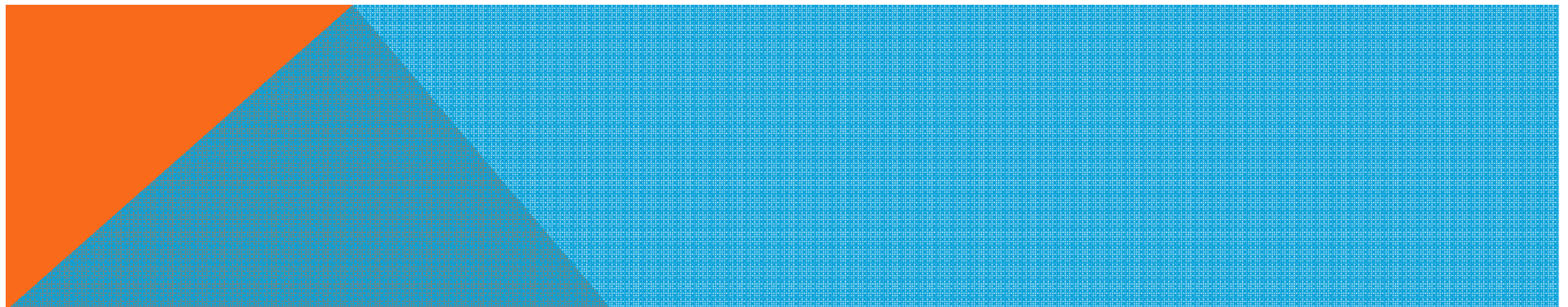
The arrival of "customers" in a queue

The number of photons hitting a photodetector

The number of telephone calls arriving at a switchboard

The long-term behavior of the number of web page requests arriving  
at a server, except for unusual circumstances such as  
coordinated *denial of service* attacks

The number of queries arriving in a stable oracle database system



# THE POISSON DISTRIBUTION

$$f(k; \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

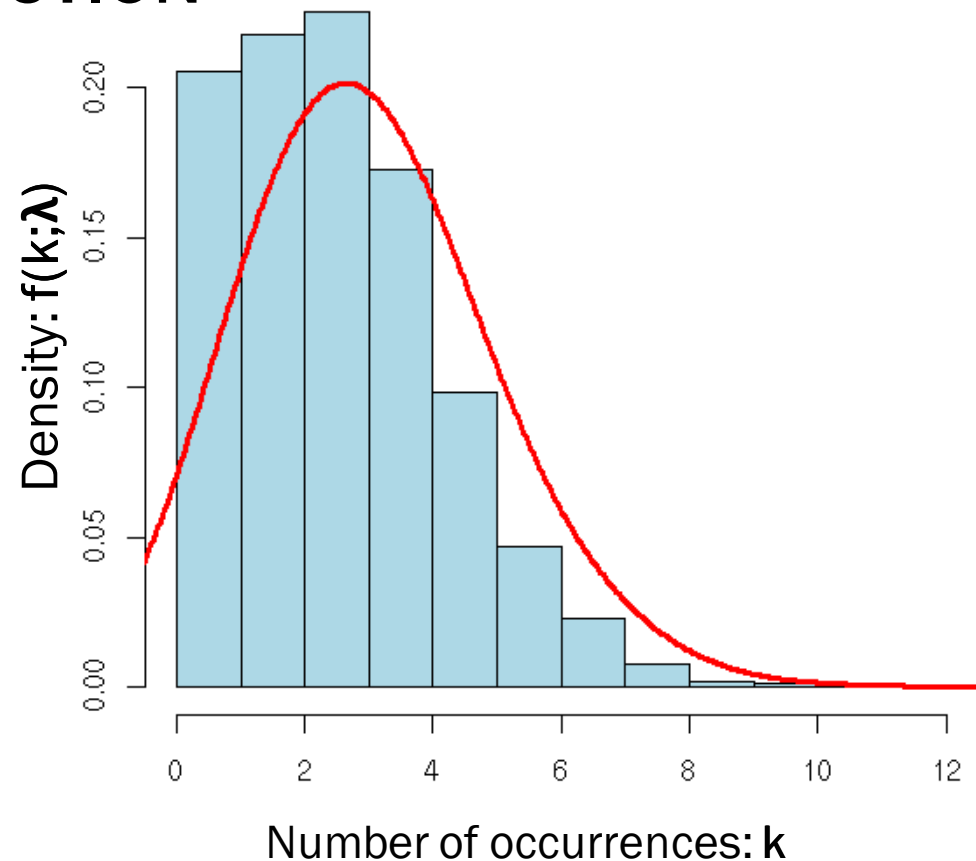
**e** - Euler's number =  $\lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n$

$e = 2.71828182845904523536\dots$

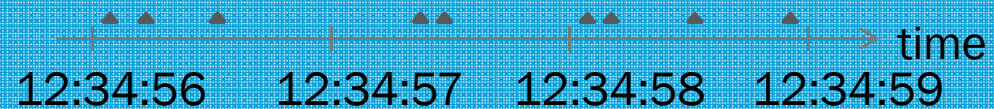
**$\lambda$**  - Mean number of occurrences

**k** - Number of occurrences (the function gives the probability for given **k** to happen)

Poisson distribution for  $\lambda=3$



Average: 3 queries per second





## THE M/M/ $m$ / $\infty$ /FCFS MODEL

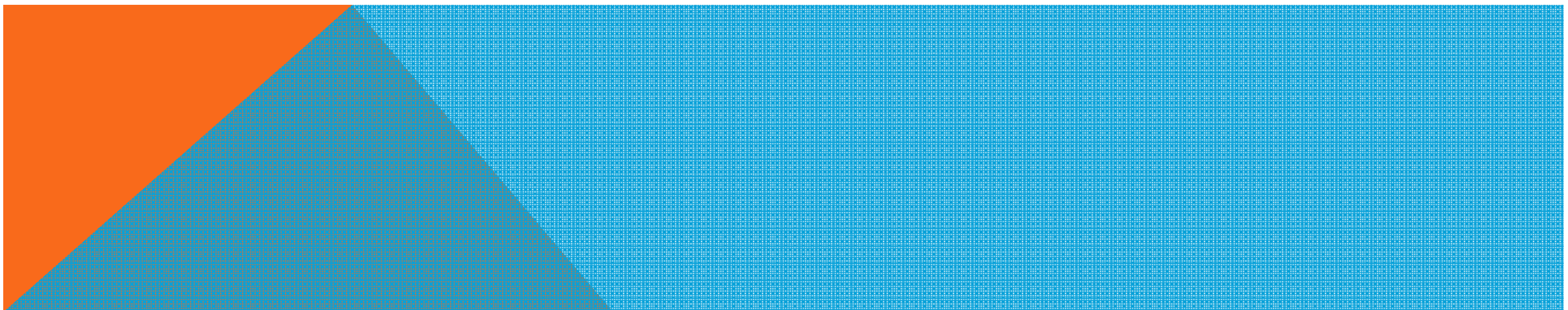
M - The request arrival time is a random variable with *Markovian* distribution

M - The service time is a random variable with *Markovian* distribution

$m$  - There are  $m$  parallel service channels, e.g.  $m$  CPUs

$\infty$  - There is no restriction on queue length

FCFS - The queue discipline is first-come, first-served



# CALCULATING THE RESPONSE TIME

$$R = S + W$$

$$W = \frac{C(m, \rho)}{m\mu(1-\rho)}$$

$m$  - number of service channels

$\rho$  - average utilization per channel

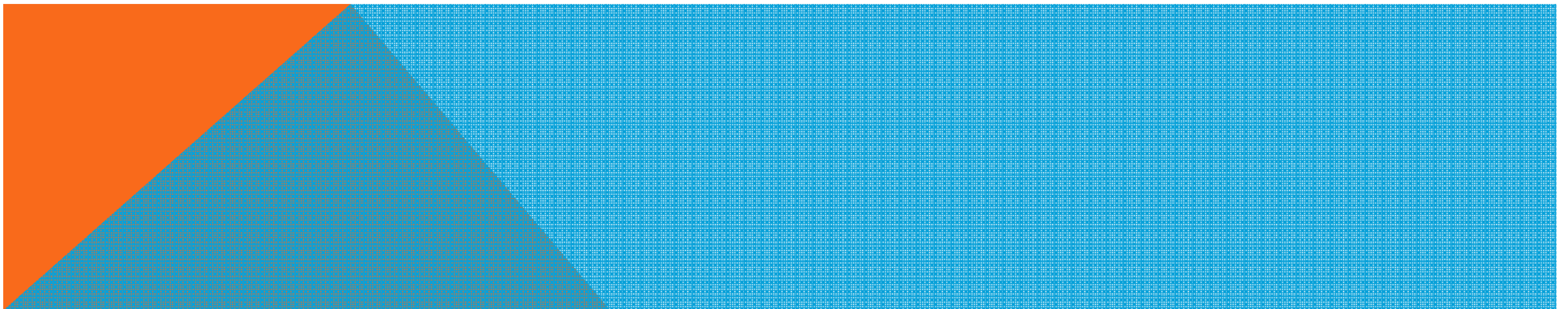
$\mu$  - the number of requests that a single service channel can complete per time unit

$C(m, \rho)$  is:

$$C(m, \rho) = \frac{\frac{(m\rho)^m}{m!}}{(1-\rho) \sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{m!}}$$

$C(m, \rho)$  is calculated in 1917 by a Agner Erlang and is called *Erlang C* (it is not the same as the number of completed requests, also noted with the letter C). There are some easier algorithms for calculation *Erlang C* with good precision.

***DEMO***



# MODELING SYSTEMS

When we want to model CPU

1 queue system

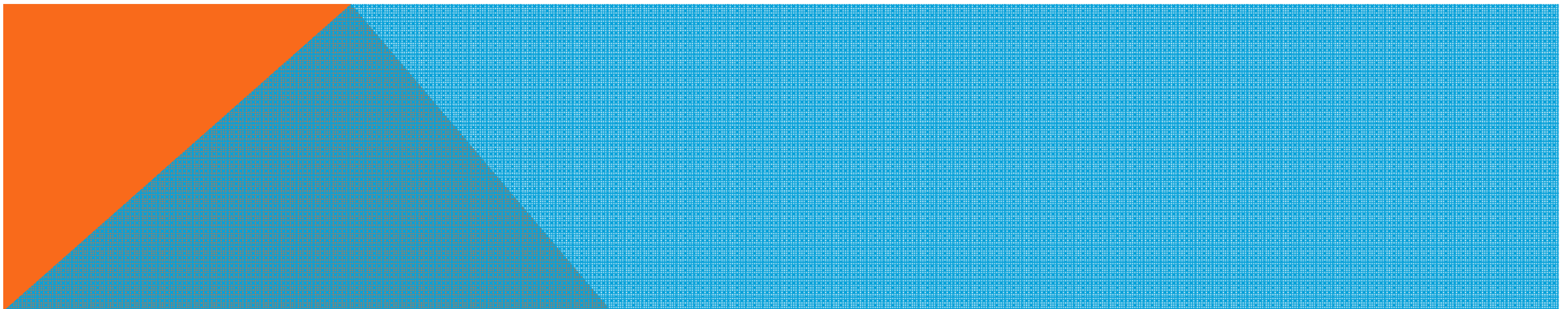
$m$  servers ( $m$ =number of CPUs/cores)

When we want to model disk subsystem

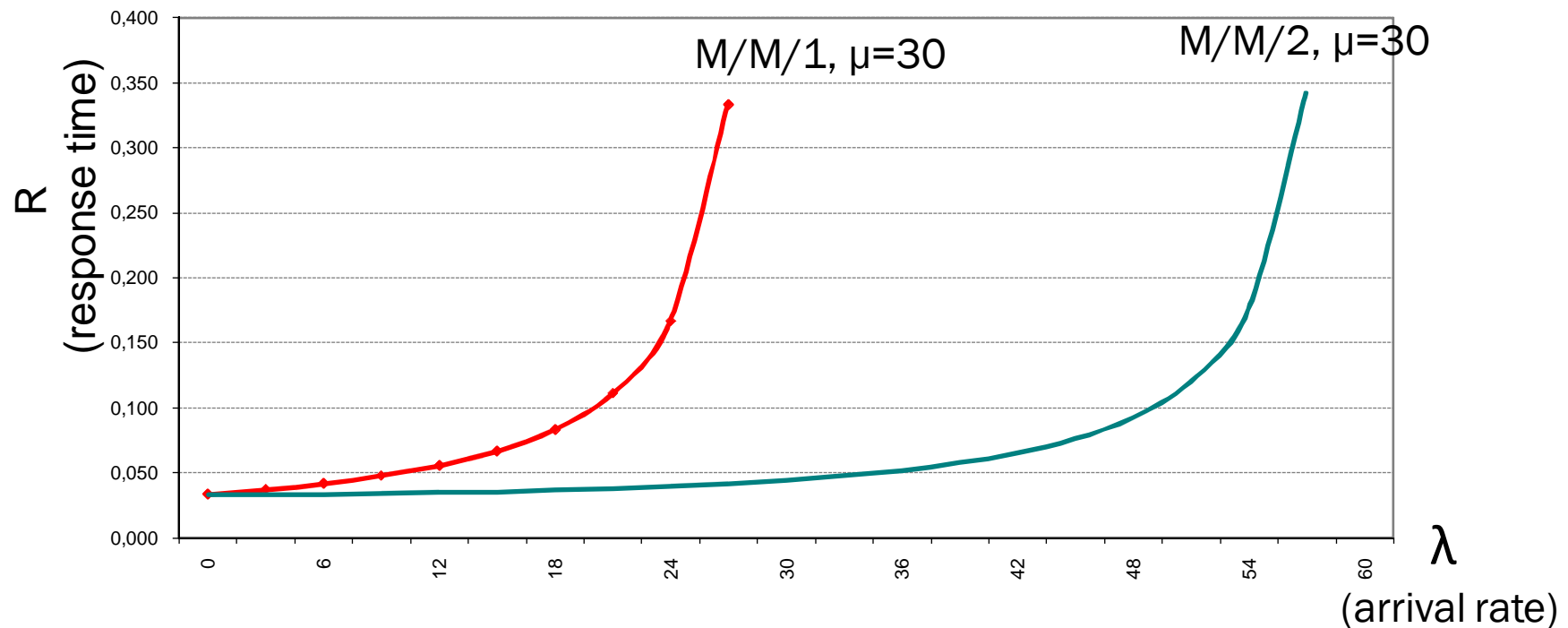
number of queues = number of  
disks/volumes/partitions

1 server per queue ( $m=1$ )

$$\lambda_{disk} = \lambda_{system} / \text{number of disks}$$

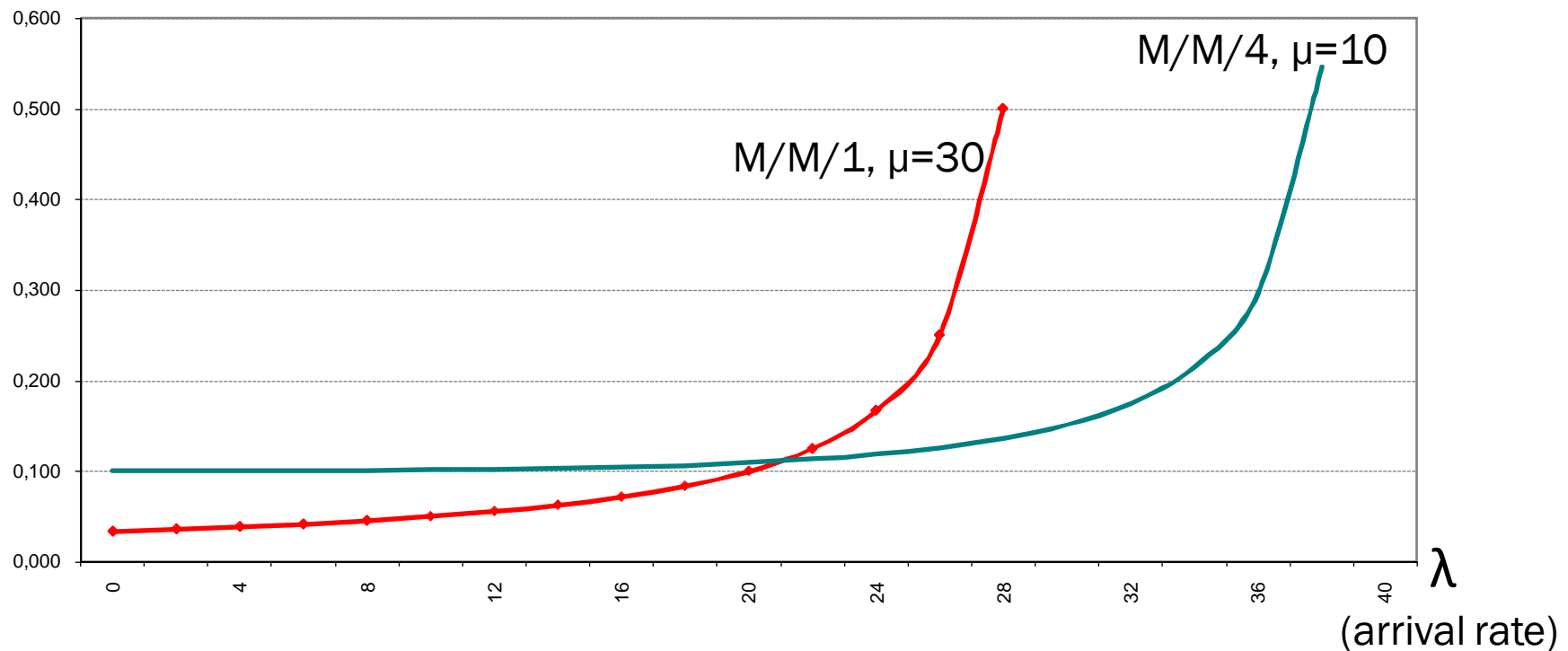


# WHAT GOOD CAN WE GET FROM THE FORMULAS?



At low arrival rates you do not gain anything from more serv

# WHAT GOOD CAN WE GET FROM THE FORMULAS?



A computer with a single fast CPU produces better response times for low arrival rates (*think of: Batch processes, DSS*), but a computer with four slower CPUs produces better response times for high arrival rates (*think of: OLTP load*)

# QUEUING THEORY

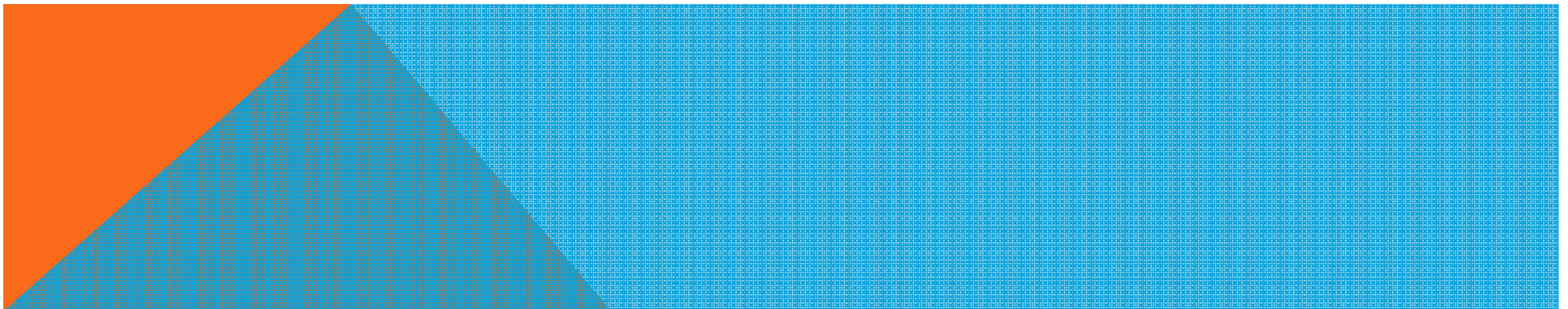
$$R = S + W$$

or

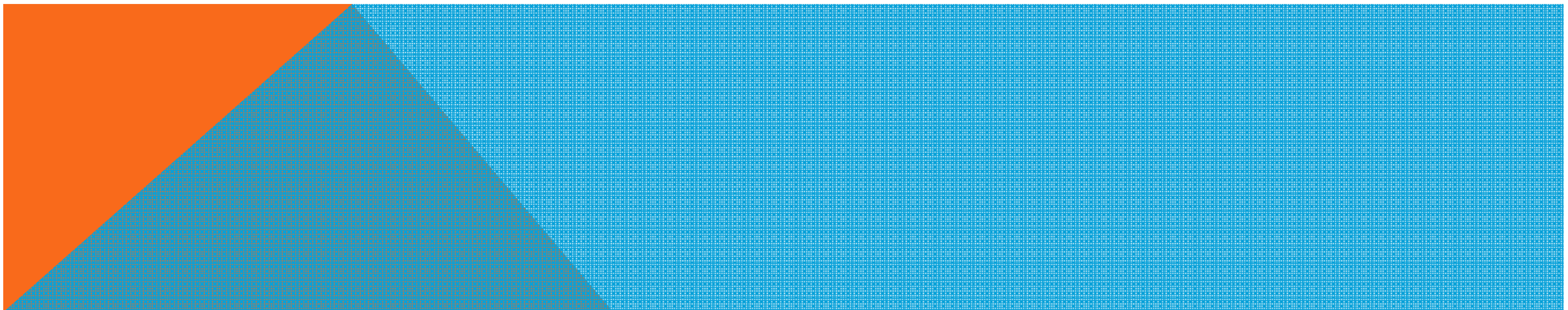
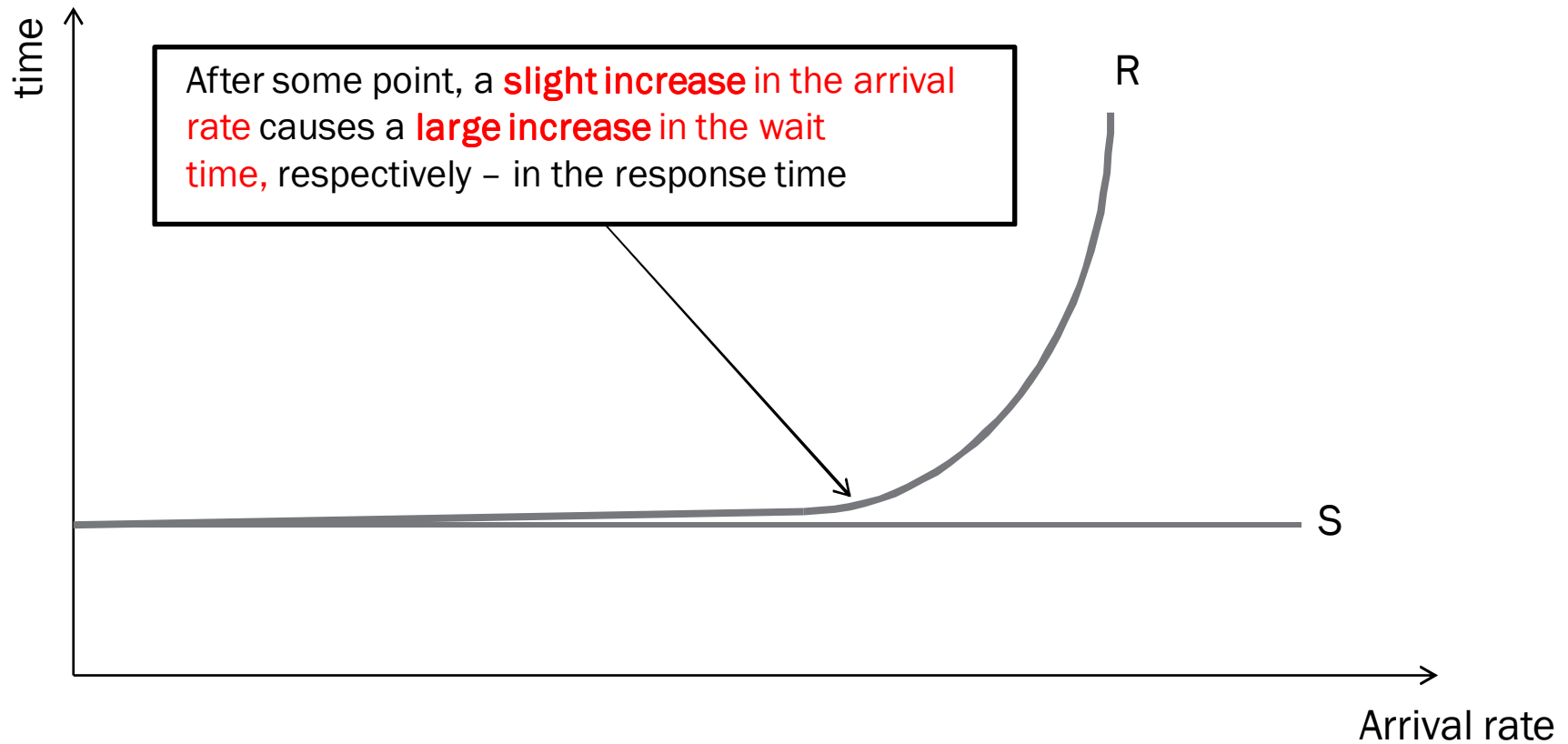
$$R_t = S_t + Q_t$$

or

Response Time = Service Time + Queuing Time

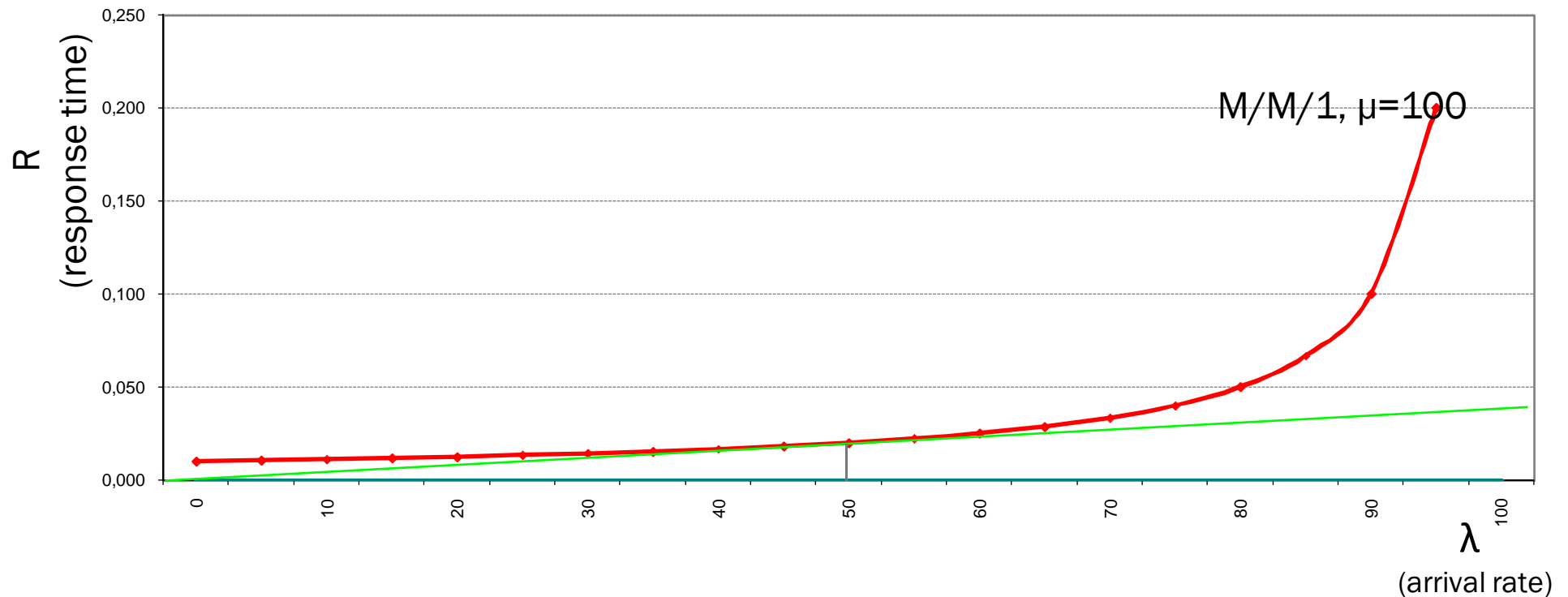


# THE *KNEE* OF THE HOCKEY STICK





# WHAT GOOD CAN WE GET FROM THE FORMULAS?



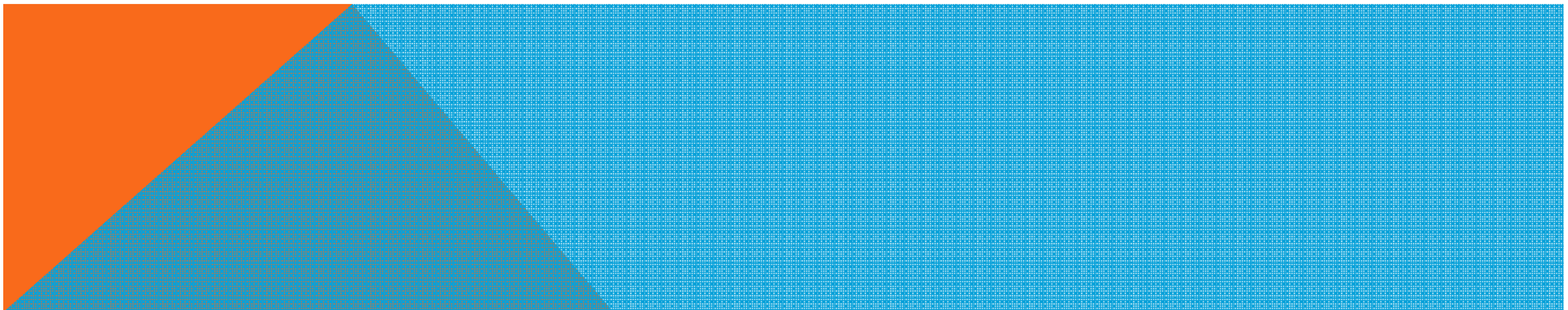
This point is called *the elbow* or *the knee* of the curve. It depends on  $m$  - the number of the *service channels* (e.g. CPUs)

## THE KNEE

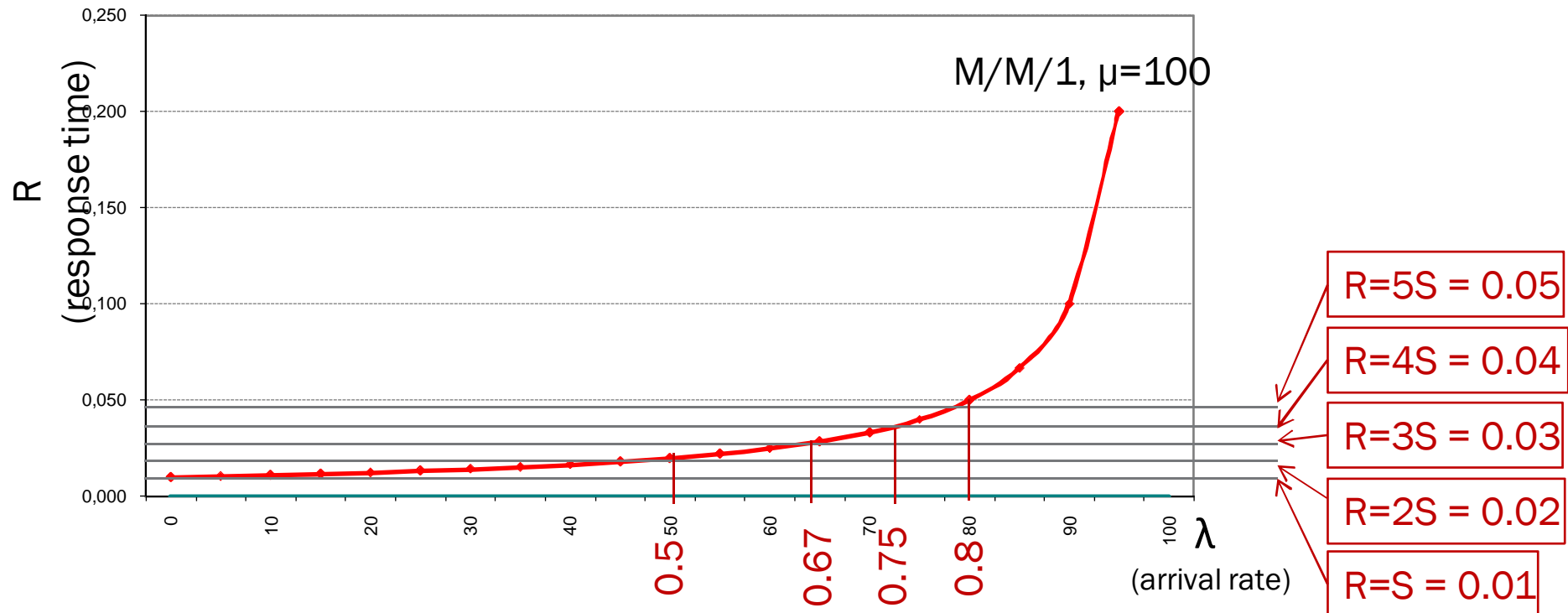
$m$	$\rho^*$
1	0.5
2	0.57735
3	0.628831
4	0.665006
5	0.692095
6	0.713351
7	0.730612
8	0.744997
16	0.810695

This table gives the utilization values  $\rho^*$ , at which the knee occurs in an M/M/m system, depending upon the value of  $m$  (*think of: number of CPUs*)

*Calculated by Cary Millsap*



# RESPONSE TIME DEGRADATION



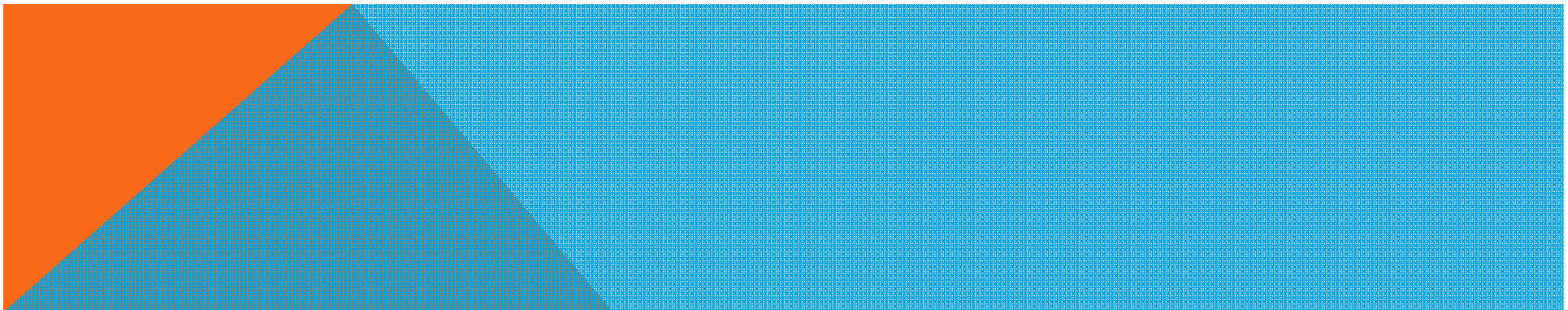
At given arrival rate (*load on the system*), the response time is double the service time. At the next point the response time is 3 times the service time, then 4, 5, etc.

It depends on  $m$  - the number of the *service channels* (e.g. CPUs)

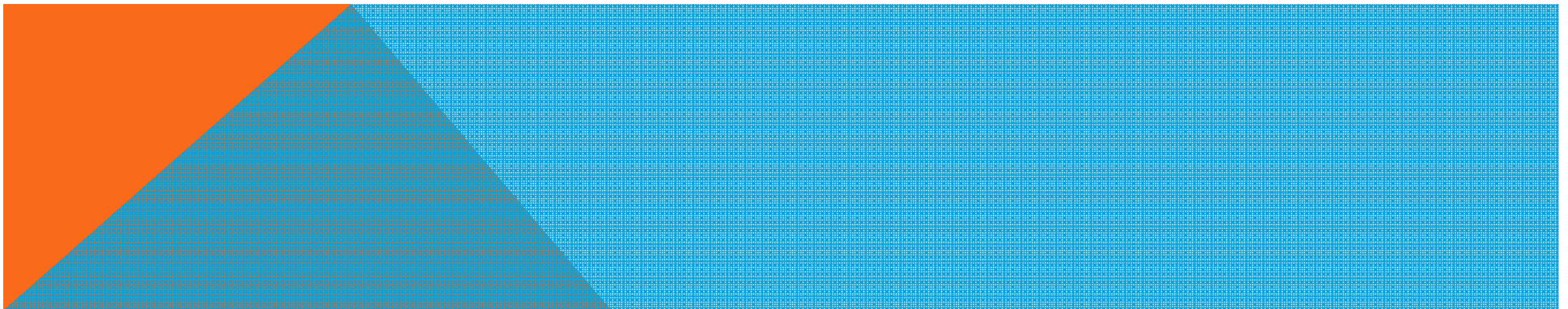
# RESPONSE TIME DEGRADATION

The following table gives for M/M/*m* system, at what utilization the response time *R* gets 2, 3, 4 or 5 times worse than the service time *S* (calculated by Cary Millsap)

	$R = 2.S$	$R=3.S$	$R=4.S$	$R=5.S$
<b>M/M/1</b>	0.5	0.666667	0.75	0.8
<b>M/M/2</b>	0.707107	0.816497	0.866025	0.894427
<b>M/M/4</b>	0.834855	0.901222	0.929336	0.944954
<b>M/M/8</b>	0.909166	0.947673	0.963169	0.971569
<b>M/M/16</b>	0.950986	0.97263	0.980984	0.985426



***THAT'S ALL ABOUT THE RESPONSE TIME***



## CUMULATIVE DISTRIBUTION FUNCTION (CDF) OF RESPONSE TIME

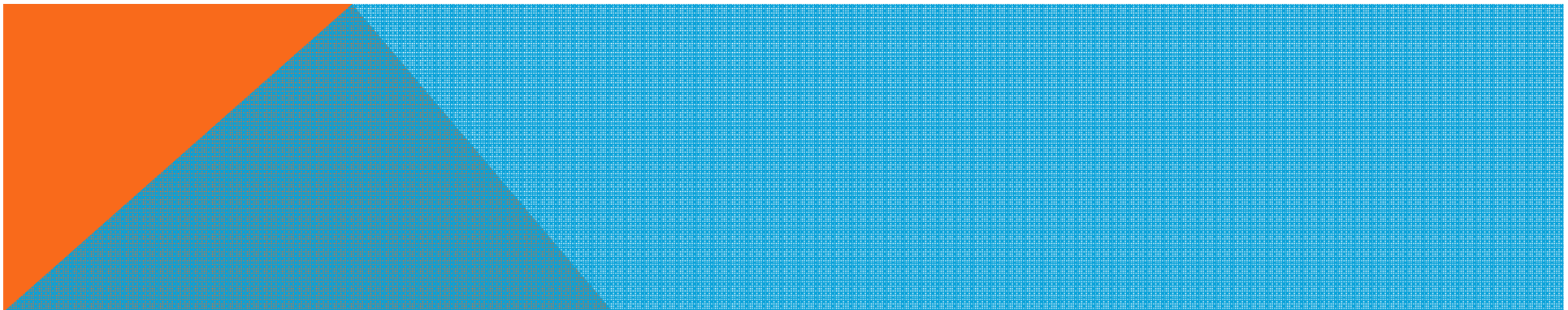
The formulas so far give the average response time of a system

The users usually care of the maximum response time

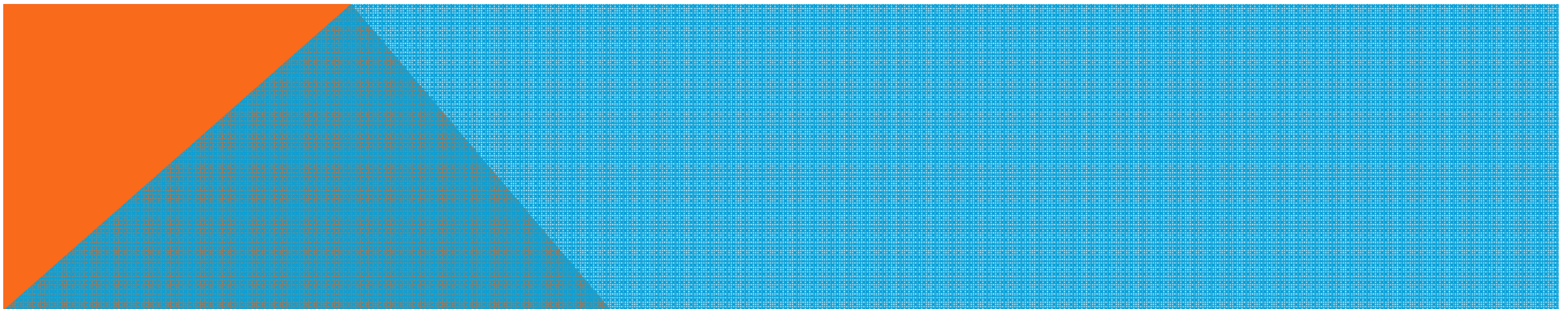
The SLA may state: *The data for building that particular page must arrive in no more than 0.3 seconds, for at least 90% of the executions*

The *cumulative distribution function (CDF) of response time* allows us to compute the probability

$P(R \leq r_{max})$  for a given request to be fulfilled with total response time less than or equal to some response time tolerance  $r_{max}$



***(PLEASE SIT BACK AND RELAX...)***



# CDF FOR M/M/M SYSTEM

$$P(R \leq r) = F(r) = \frac{m(1-\rho) - W_q(0)}{m(1-\rho) - 1} (1 - e^{-\mu \cdot r}) - \frac{1 - W_q(0)}{m(1-\rho) - 1} (1 - e^{-(\mu \cdot \rho - \lambda)r})$$

where  $W_q(0)$  is:

$$W_q(0) = 1 - \frac{(m\rho)^m p_0}{m!(1-\rho)}$$

where  $p_0$  is:

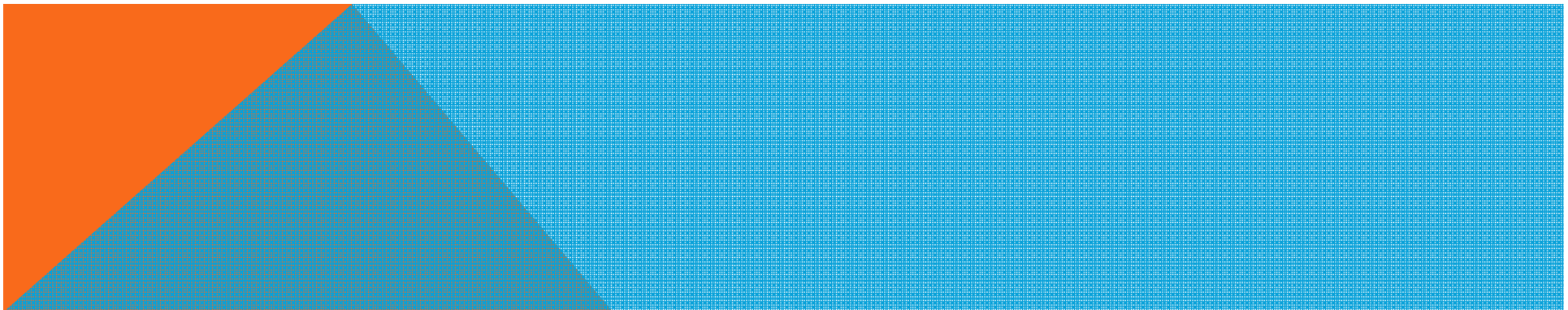
$$p_0 = \left( \sum_{n=0}^{m-1} \frac{(m\rho)^n}{n!} + \frac{(m\rho)^m}{m!(1-\rho)} \right)^{-1}, \rho < 1$$

$m$  - number of service channels

$\rho$  - average utilization per channel

$\mu$  - the number of requests that a single service channel can complete per time unit

$\lambda$  - arrival rate





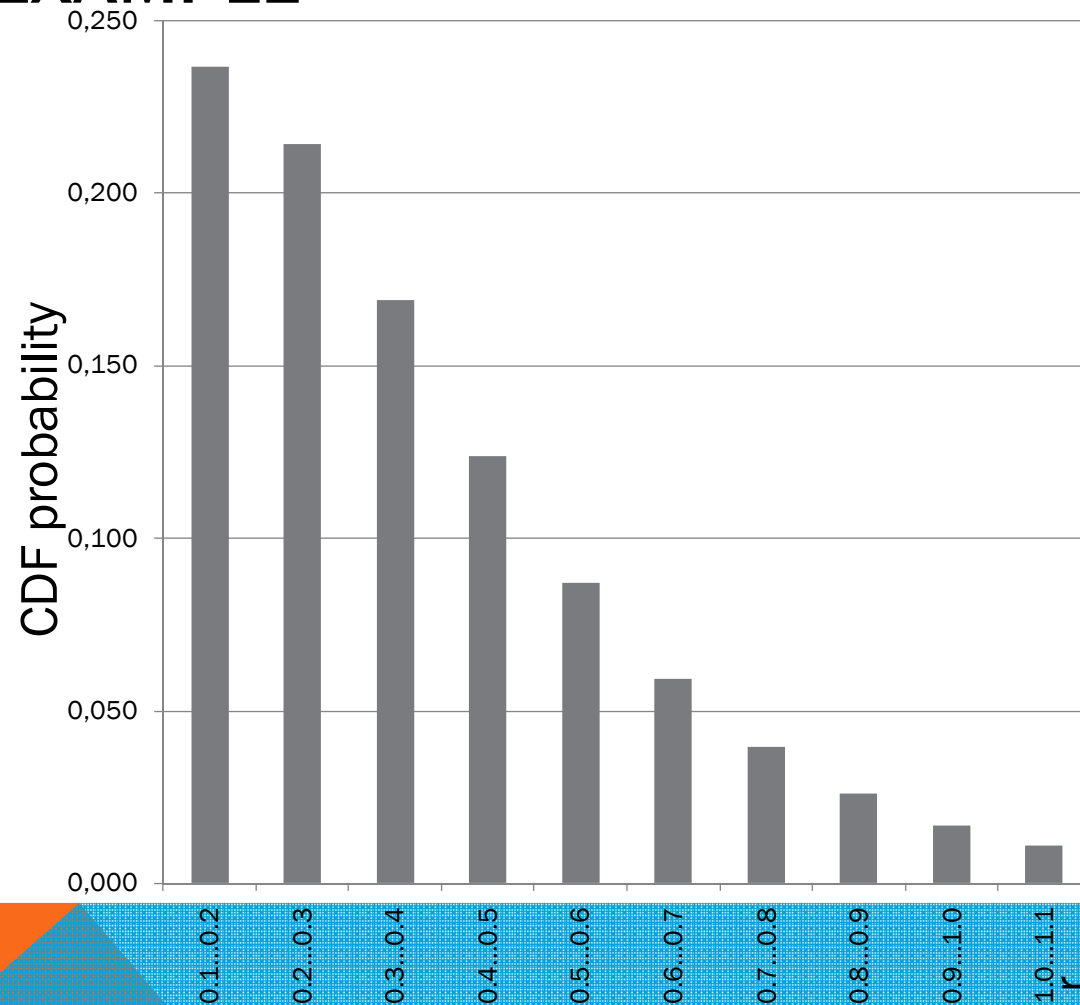
# CDF EXAMPLE

The system:

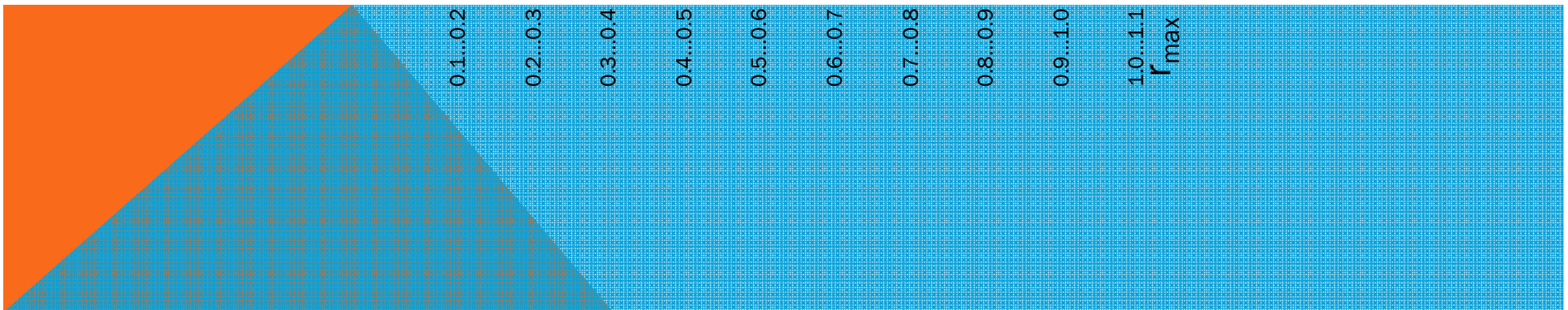
- M/M/5
- $\lambda = 25$  tx/s
- $\mu = 6$  tx/s

Average:

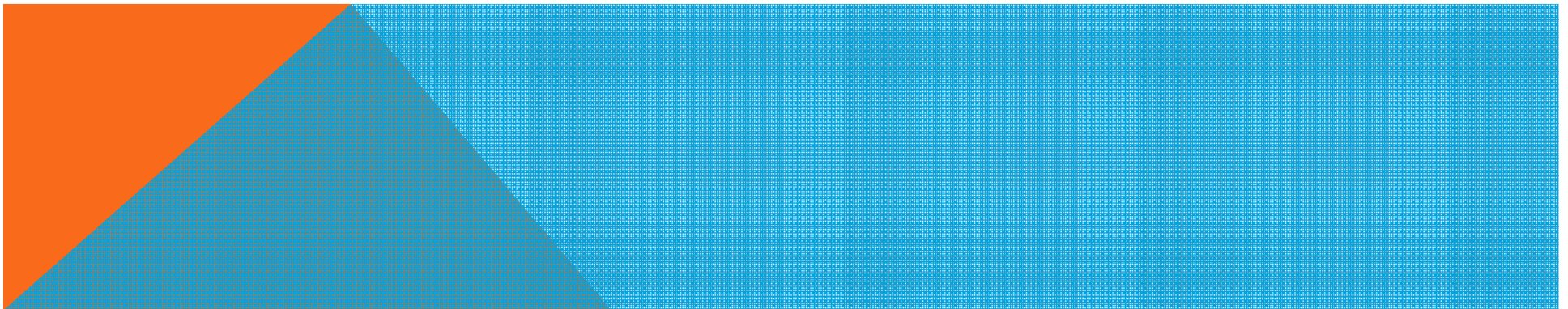
- $S = 0,167$
- $W = 0,124$
- $R = 0,291$



$r_{\max}$	CDF
0.1...0.2	0,236
0.2...0.3	0,214
0.3...0.4	0,169
0.4...0.5	0,124
0.5...0.6	0,087
0.6...0.7	0,059
0.7...0.8	0,039
0.8...0.9	0,026
0.9...1.0	0,017
1.0...1.1	0,011



***THAT'S ALL (THE THEORY)***





**PART 4**  
WRAP UP

## PRACTICAL QUEUING: WHEN THE RESPONSE TIME IS NOT GOOD ENOUGH

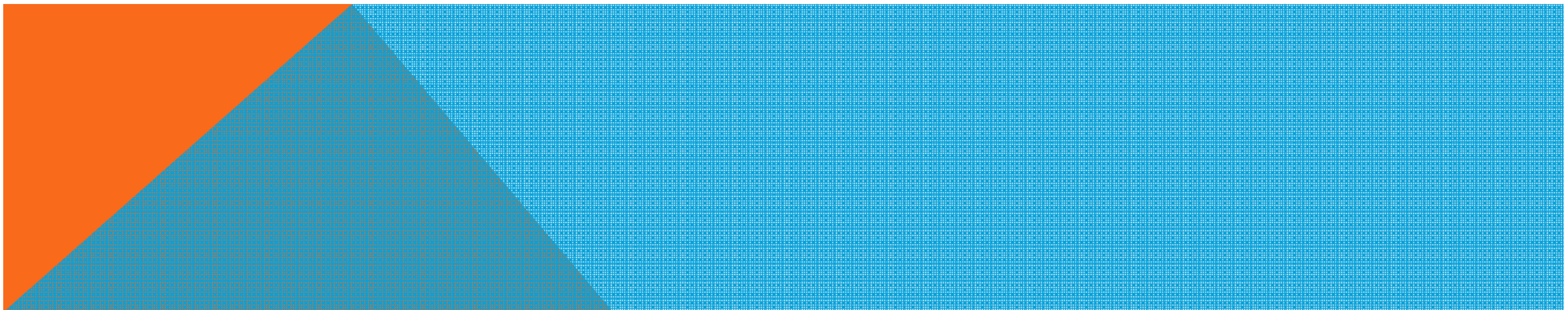
$$R = S + W$$

$\mu$  – you can increase the service rate by reducing the code path. This will decrease both  $S$  and  $W$

$\lambda$  – check if you can negotiate the arrival rate with the business. This will decrease  $W$

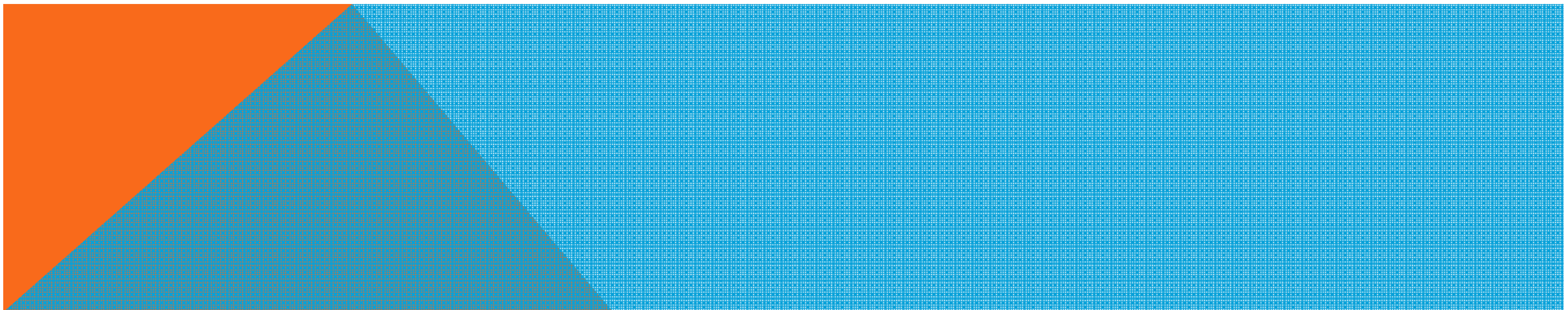
$\mu$  – you can increase the service rate by providing faster hardware. This will decrease both  $S$  and  $W$

$m$  – you can think of increasing the service channels (CPUs) . This will decrease  $W$



# TIPS ON USING MATHEMATICAL MODELS

- **Be careful with getting input data**
  - Getting too many samples may influence the real performance and make the calculations harder
  - Getting too few samples will produce bogus results. At least 30 samples are needed for the model to work
  - Be careful with the outliers
- **Always check the precision of the forecasting**
  - E.g. if you have 100 samples, make the model with 80 and check it with the rest
- **Avoid steps that may lead to lower precision**
  - Estimations, external influences, etc.



# DRAWBACKS OF USING MATHEMATICAL MODEL

Response time calculation is based on a model. So it's not real; it's an abstraction

There is no perfect mathematical model

Models are not reality. They are not perfect, cannot be expected to be perfect, and are not designed to be perfect

Service time is never perfectly horizontal, and queue time does not occur exactly as we plan in real systems

There are physical constraints: you can't install 6.38 CPUs. Neither you can install 1000

No system is perfectly scalable. 8 CPUs does not work 8 times more than 1 CPU



There is always some other workload

There will be changes in input parameters sooner than you think

# BENEFITS OF USING MATHEMATICAL MODEL

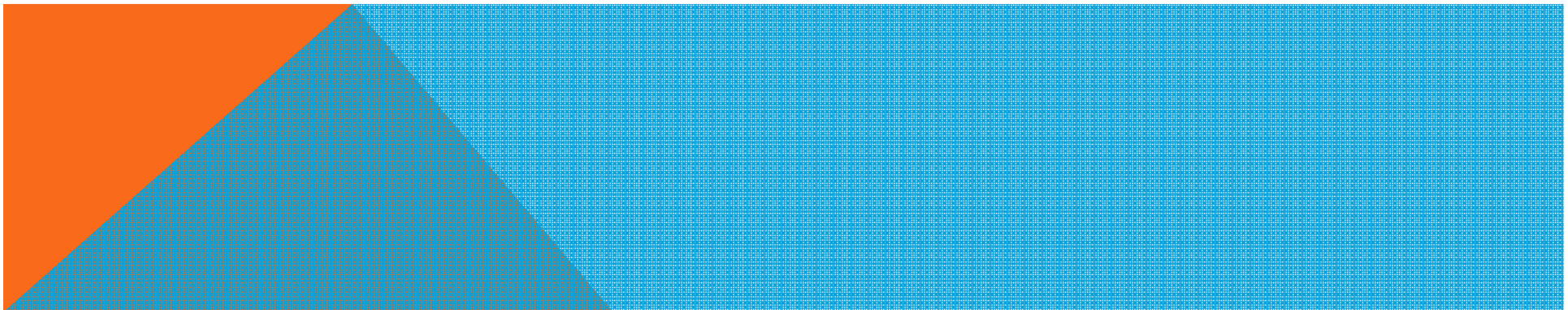
Calculation is cheaper than real world tests

Calculation is faster than real world tests

Changing parameters is easier in a spreadsheet than in real world

It makes you think about lots of aspects of your system

Most of all, it helps you really understand that the performance of a system is predictable. Forecasting performance is not black magic, it is science.



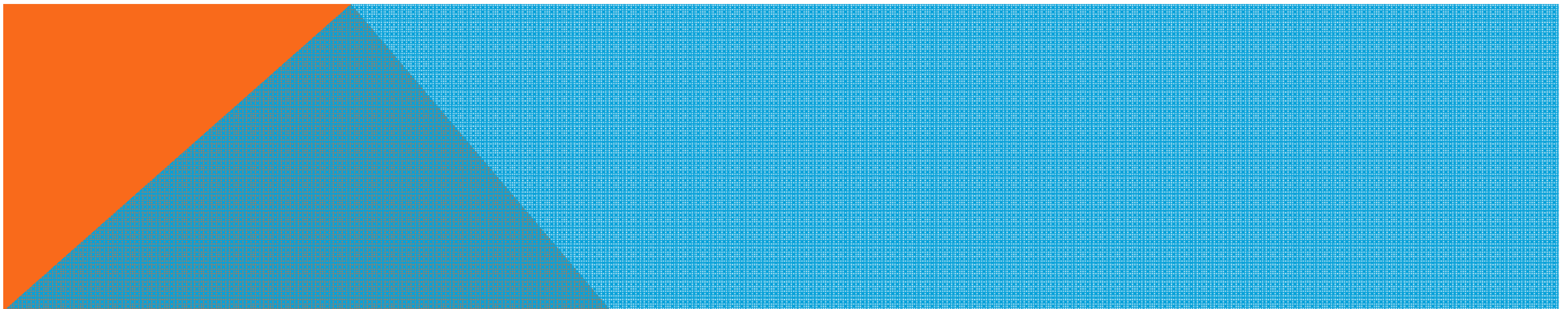
## REFERENCES

*“Optimizing Oracle Performance”*, Cary Millsap

*“Forecasting Oracle Performance”*, Craig Shallahamer

*“The CoE performance method”*, Roger Snowden, Center of Expertise, Oracle Corporation

[http://method-r.com/downloads/doc\\_details/44-thinking-clearly-about-performance](http://method-r.com/downloads/doc_details/44-thinking-clearly-about-performance)





## RESOURCES

On Orapub.com there is section called Tools. There you can find:

*“M/M/m Queuing Theory Analysis Spreadsheet”*  
workbook for Excel

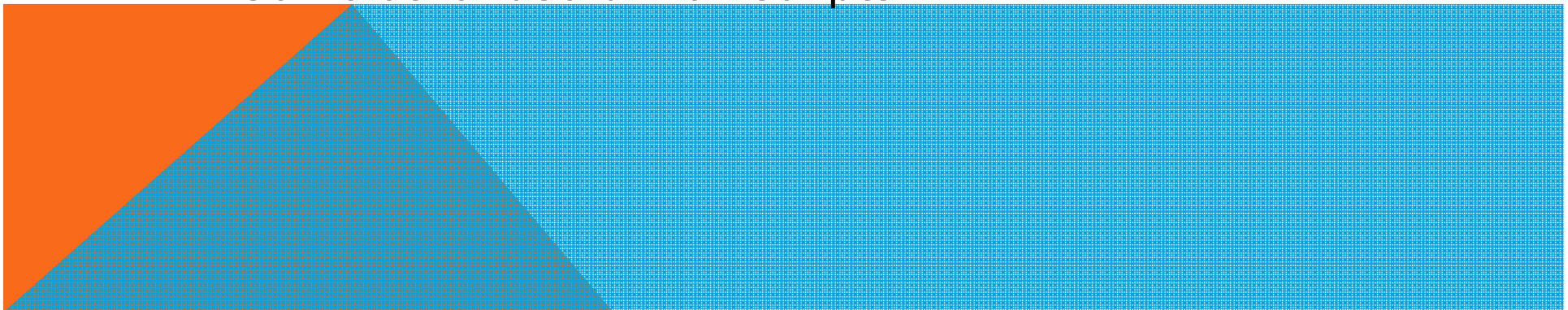
*Erlang-C Mathematics* in Perl


Excel Add-in: *M/M/m Queuing Theory Functions*

With the book “Optimizing Oracle Performance” you get

*“Queueing Theory Multiserver Model”* workbook for  
Excel

Some other useful Perl scripts





QUESTIONS?

Yavor Ivanov  
Oracle Certified Master  
<http://blog.yavor.info/>