QUEUEING THEORY

Brice Figureau - Asmodee Digital - 2017

DEFINITION

DEFINITION

Queueing theory is the mathematical study of waiting lines.

HISTORY

Agner Krarup Erlang designed models to describe the Copenhagen telephone exchange in 1909.

THERE ARE QUEUES EVERYWHERE !

IN LIFE

- toilets in the morning
- traffic jam
- banks, post office
- restaurant
- even at work! ...

for instance at work: work in progress is a queue that you want to minimize

OR IN COMPUTER SYSTEMS

- RAM / CPU
- routers
- kernels
- software ...

QUEUEING HAPPENS EVERYWHERE.

WAITING TIME IS LOST TIME.

WHY WOULD YOU WANT TO STUDY THAT ?

- systems get congested under load (traffic jams?)
- waiting time increases
- queueing theory allows understanding relationships between congestion and delay
- capacity planning
- performance analysis

MODELS

Queueing theory proposes models based on some assumptions, if your system verifies those assumptions you can predict the queues behaviors and now the system limits.

It allows to simulate behaviors by changing the queue model.

LITTLE'S THEOREM

$$L = \lambda W$$

- L is the average number of customers or events in the queuing system
- λ is the average customer or event arrival rate
- W is the average service time for the customer or event processing time

restaurant: double the number of customer arriving, you get double of customers in the restaurant same if you double the service time. Applies only to steady systems and to averages.

WHAT'S IN A QUEUE ?

QUEUEING DISCIPLINE

aka how do you serve the clients

- FIFO
- LIFO
- Priority
- Loop

we're focusing only on FIFO

ARRIVAL DISTRIBUTION

- random arrival over long periods
- customers arrival independent of other
- count the arrival per time slot
- distribution: histogram of arrival frequency
- this is the Probability Mass Function

ARRIVAL DISTRIBUTION (SUPERMARKET)

- count number of arrivals per 5 min slots
- count number of occurences we had for 0, 1, 2... customers
- relative frequency:

$$A_n/Total$$

count arrival of customers per 5 min slots count the number of times it happened for 0 cust, 1 cust, ... compute relative frequency

ARRIVAL DISTRIBUTION (SUPERMARKET)

Arrivals	number	relative freq
0	255	47%
1	190	35%
2	72	13%
3	19	4%
4	3	1%
total	539	100%

ARRIVAL DISTRIBUTION (SUPERMARKET)

$$\lambda = 0x47\% + 1x35\% + 2x13\% + 3x4\% + 4x1\% = 0$$

This forms a *Poisson distribution* (like most random phenomenon)

$$P(x) = \frac{\lambda_x e^{-\lambda}}{x!}, x = 0, 1, 2...$$

$$P(0) = \frac{1e^{-0.75}}{1} = e^{-0.75} = 0.47$$

This also is called a Markovian distribution

SERVICE TIME DISTRIBUTION

Same kind of analysis (except we measure the time spent to serve a client) as the Arrival distribution, if random an Exponential distribution applies.

KENDALL CLASSIFICATION

- suggested by David Georges Kendall in 1957
- simplest form: a/b/c

with

- a: probability distribution of arrival
- b: probability distribution of service time
- c: number of servers

KENDALL CLASSIFICATION

- M: Poisson distribution (arrival) or Exponential (service time)
- E: Erlang distribution
- D: Constant distribution
- G: General distribution (with known mean/variance)

KENDALL

- M/M/1 -> Poisson, Exponential 1 server
- M/M/2 -> 2 servers
- M/D/3 -> constant service time, 3 servers

complex form adds 2 parameters total capacity and total population assumed infinite in simplest form

INFINITE QUEUES ?

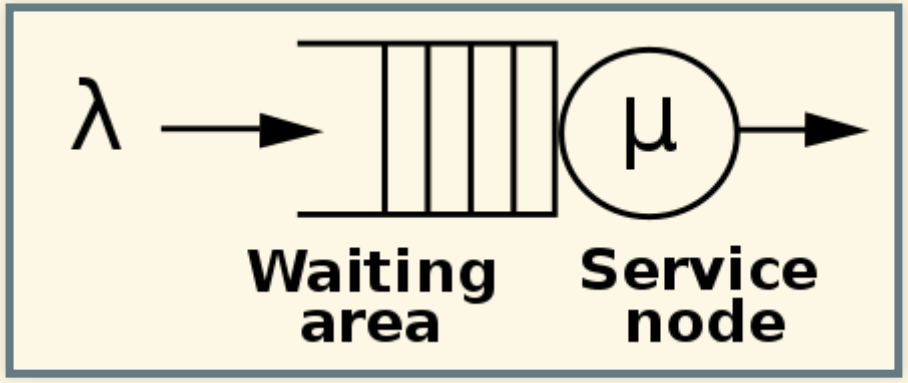
- λ: arrival rate (client/unit of time)
- μ: processing rate (client/unit of time)
- λ / μ < 1 -> finite queue
- λ / μ > 1 -> need an inifinite: bad!

M/M/1

- most used/simple model
- Poisson:
  - assumes large number of clients
  - impact of one client small on the system
  - indepence of clients

M/M/1 THE GORY DETAILS

given λ and μ (=1/S)



Utilization (% of use)

$$\rho = \frac{\lambda}{\mu}$$

I'm won't do any demonstration but all those formulas can be demonstrated with probabilities (see Erlang 1917 paper)

M/M/1 THE GORY DETAILS

Average time in waiting line:

$$W = \frac{\rho S}{1 - \rho}$$

Average residence time:

$$R = W + S = \frac{S}{1 - \rho} = \frac{1}{\mu - \lambda}$$

M/M/1 THE GORY DETAILS

Average number of customers in waiting line:

$$L = \lambda W = \frac{\rho^2}{1 - \rho}$$

Average number of customers in system:

$$Q = \lambda R = \frac{\rho}{1 - \rho}$$

M/M/1 THE GORY DETAILS

Probability of no customers:

$$P_0 = 1 - \frac{\lambda}{\mu}$$

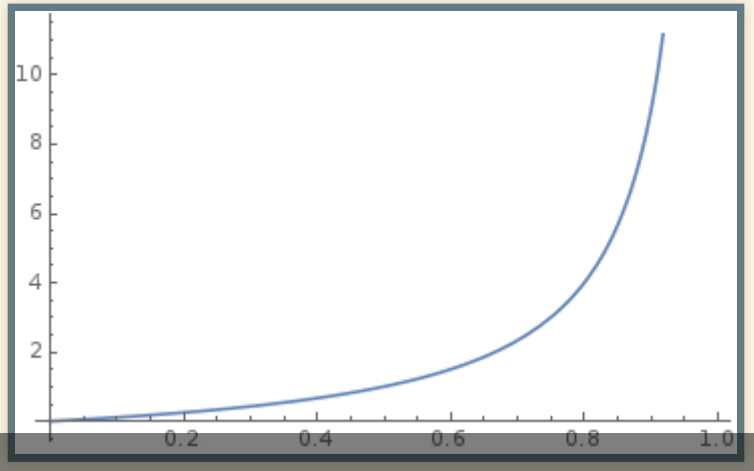
Probability of a # *n* of customers

$$P_n = \rho^n P_0$$

M/M/1 CONCLUSION

(normalized) residence time proportional to

$$\frac{1}{1-\rho}$$



at high utilization, waiting times become asymptotical ! Clearly we need to keep the system under optimal utilization.

M/M/C

Much more complex, based on Erlanc C formula

Probability that a customer has to enter a queue

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

$c\rho$ = traffic intensity mesured in Erlangs

Average number of customers in system:

$$L = \frac{\rho}{1-\rho} C(c,\rho) + c\rho$$

C equation requires iterative computations - can't be computed in Excel it's also not intuitive (can't derive any information if c increases/decreases) Erlang is a measure of load (number of call active at a moment in a telephone system)

M/M/C

We need approximations !

Hirotaka Sakasegawa 1977 approximation

$$L_q \approx \frac{\rho\sqrt{2(c+1)}}{1-\rho}$$

increase rho -> increase in q time increase c -> decrease of q time

M/M/C

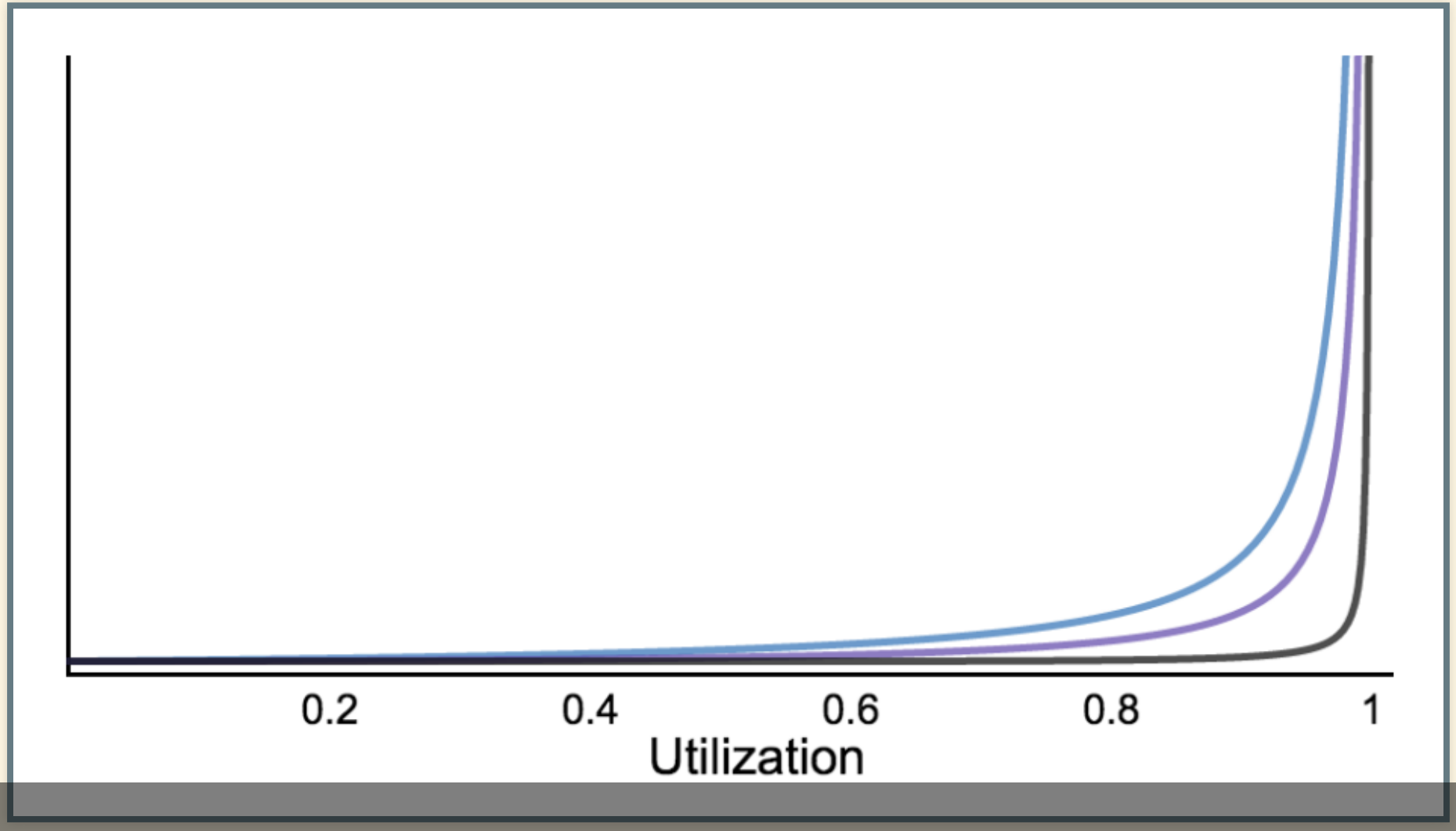
Gunther's approximation:

$$L_q \approx \rho c(\frac{1}{1-\rho^c} - 1)$$

M/M/C

servers	utilization	Erlang C	Sakasegawa	Gu
1	0.500	0.500000	0.500000	0.5
2	0.500	0.333333	0.333333	0.3
2	0.999	997.50175	997.50175	99
8	0.500	0.059044	0.015686	0.1
8	0.999	995.760755	994.509747	99
64	0.999	989.334082	966.873556	98

UTILIZATION PER # OF SERVERS



different server count=1, 2, 16 this has an impact -> more concurrency can use system at higher utilization

COMBINED OR SEPARATE ?

Airport checking with 2xM/M/1 or an M/M/2 with 240 passengers/hour with a service time of 15s per passenger:

Metric	Combined	Separate
Utilization	50%	50%
Avg Q Length	0.33	0.5
Avg W	20s	30s

COMBINED OR SEPARATE ?

4 times more passengers, adding 4 servers

Metric	Combined	Separate
Utilization	50%	50%
Avg Q Length	0.06	0.5
Avg W	15.24s	30s

CONCLUSION?

- There are queues everywhere
- we add queues to improve availability, utilisation and throughput but it costs latency/time/money
- Helps build better intuition about effecting variables
- Understand variables relationship (Little's linear Law, non-linear variables)

BUT

BUT

- those models don't apply to all systems (not all processes are Markovian)
- complex for networks of queues (ie computers, distributed systems etc)
- closed queues are a different world (machine repairman model)

for instance M/M/c can't be used to predict end-to-end traffic as adding travel time makes it non markovian.

... AND NOBODY EXPECTS THE

UNIVERSAL SCALABILITY LAW

DEFINITION

Andre B. Bondi:

> *the capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged in order to accommodate that growth*

quite informal

DEFINITION

Neil Ghunther:

> *Scalability can be defined as a mathematical function, a relationship between independent and dependent variables (input and output).*

a function whose inputs is size/load and output is throughput

SOME ELEMENTS OF THIS FUNCTION

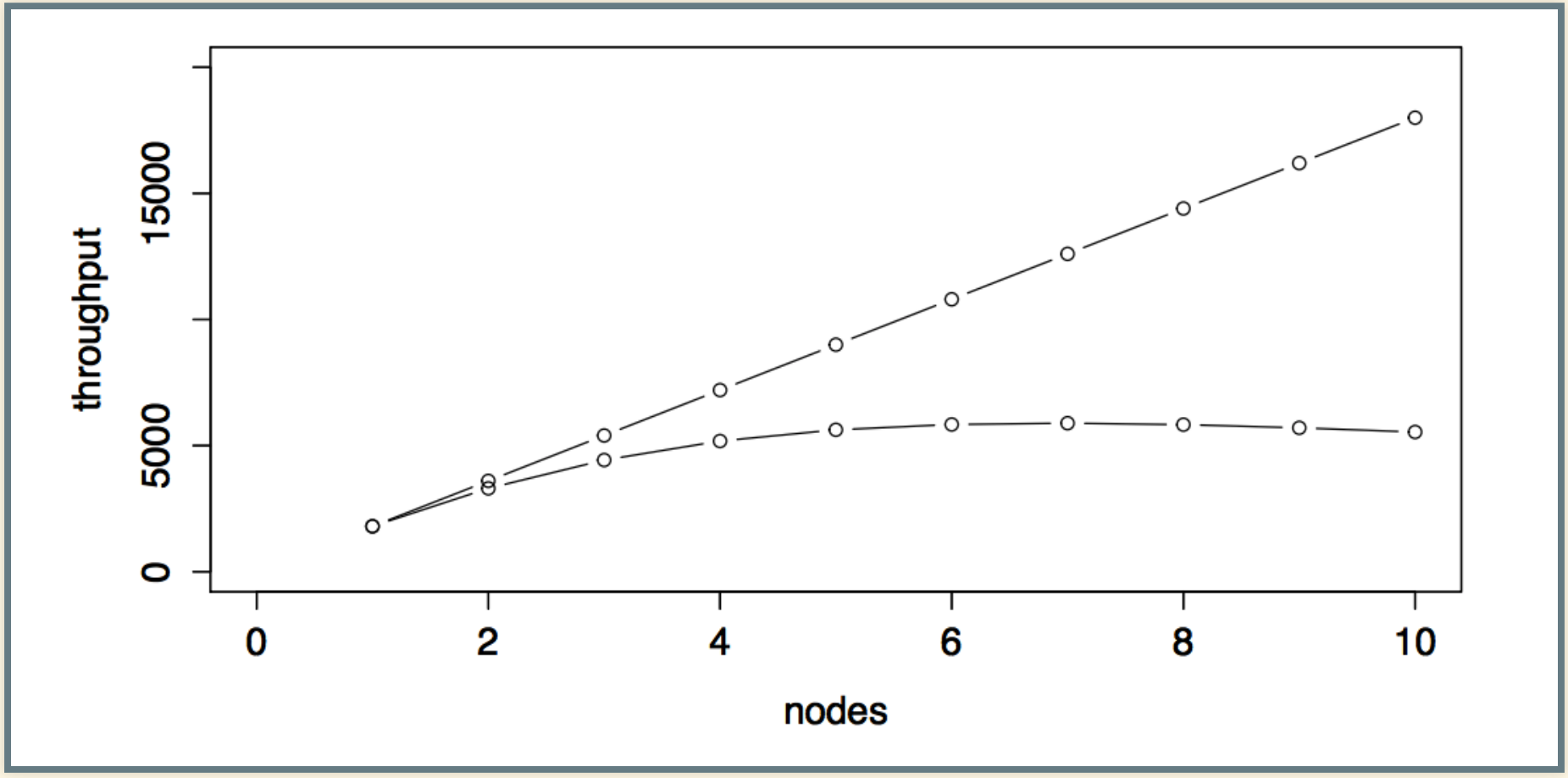
- requests
- arrival rate
- concurrency
- number of clients

some of those elements are dependent, some might be independent of other elements

THERE IS NO SUCH THING AS LINEAR SCALABILITY

you can add more resource, it will give a boost, but nothing tells the system scale linearly

THERE IS NO SUCH THING AS LINEAR SCALABILITY



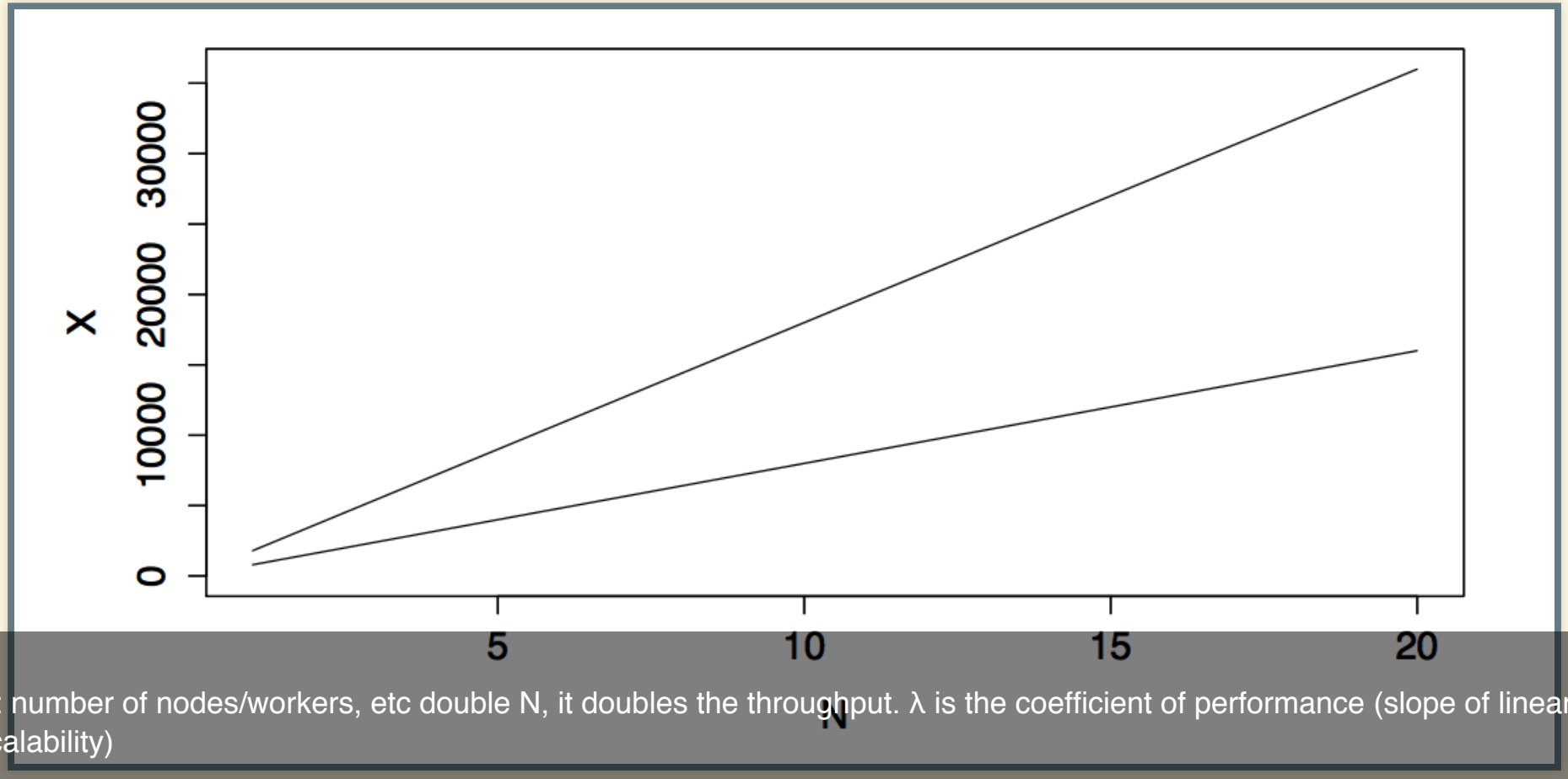
WHY ?

- system loose efficiency when growing, can even exhibit retrograde scalability
- contention: queues form because not everything can happen in parallel
- crosstalk: coherency penalty lost when parts has to synchronize or exchange information

contention: think RAM access in multi-cpu, or specialized workers in team crosstalk: think threads, cpu, host in a distributed system happens also to people in teams

USL: THE IDEAL CASE

$$X(N) = \frac{\lambda N}{1}$$

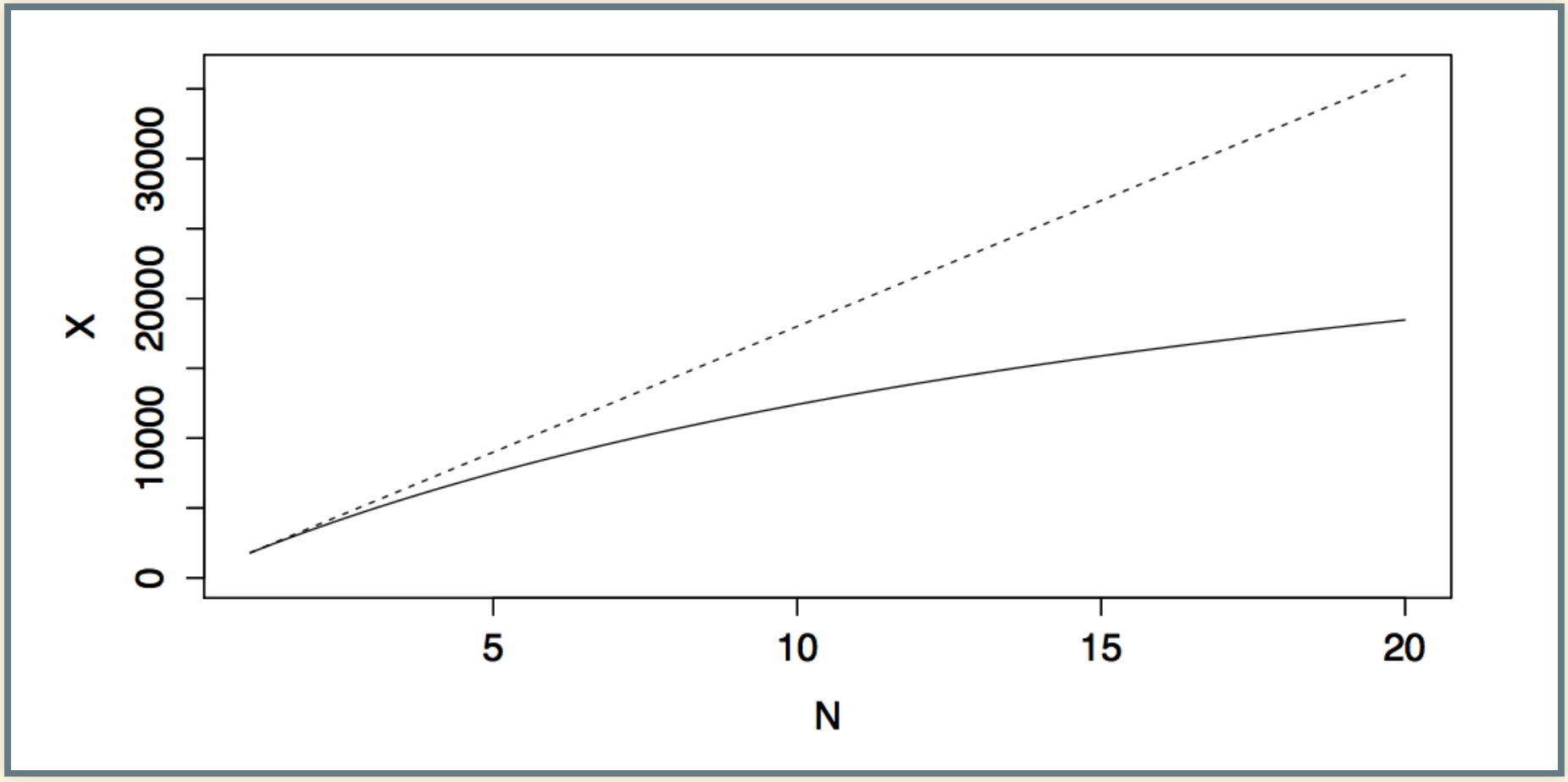


N: number of nodes/workers, etc double N, it doubles the throughput. λ is the coefficient of performance (slope of linear scalability)

USL: ADDING CONTENTION PENALTY

Amdahl's Law:

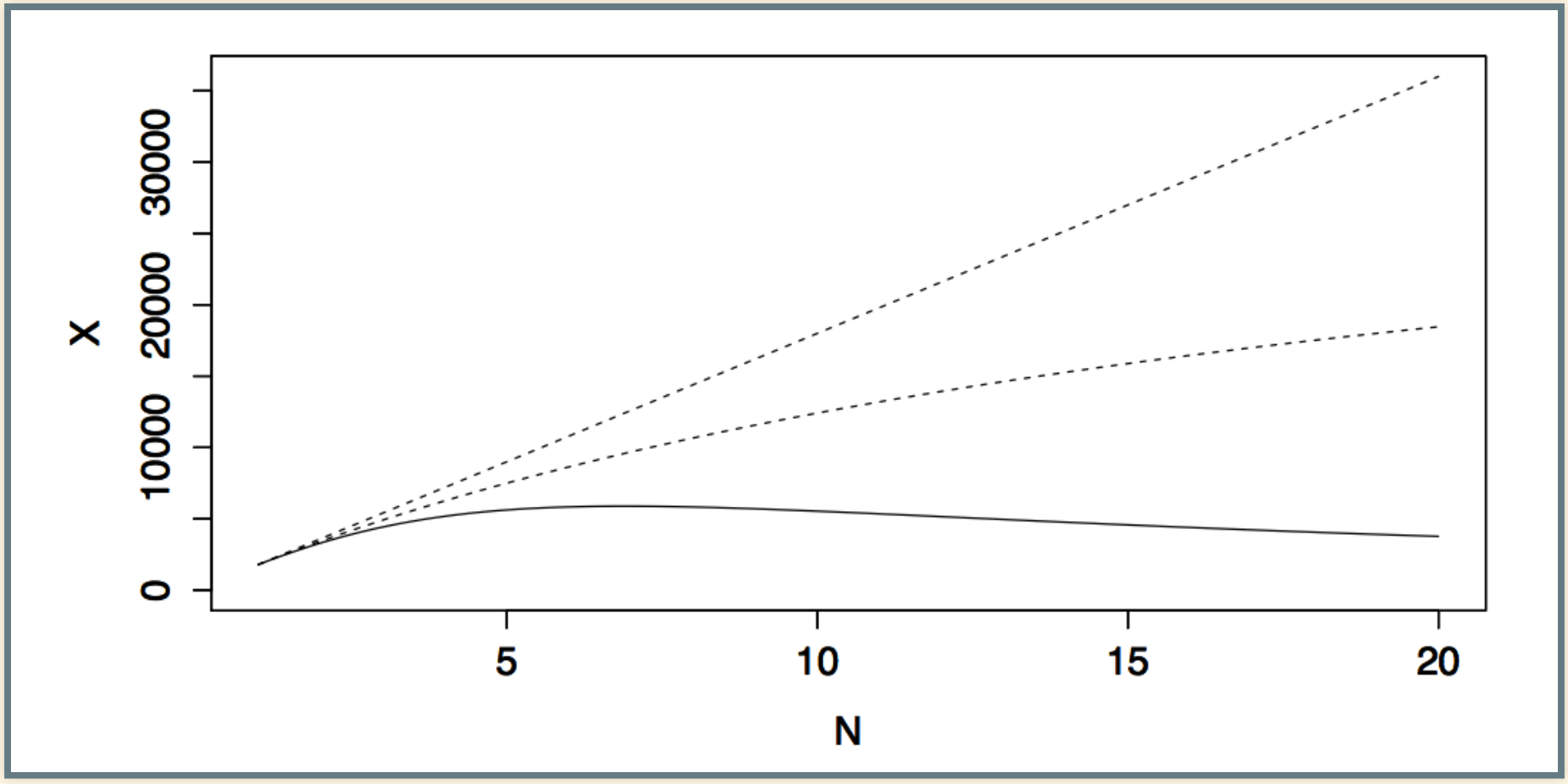
$$X(N) = \frac{\lambda N}{1 + \sigma(N - 1)}$$



Contention increase with parallelism Amdahl's Law: the maximum speedup possible is the reciprocal of the non-parallelizable portion of the work sigma: coefficient of contention

USL: ADDING CROSSTALK PENALTY

$$X(N) = \frac{\lambda N}{1 + \sigma(N - 1) + \kappa N(N - 1)}$$



crosstalk happens between 2 components of a system, there are n(n-1) relationships possible in a system. The crosstalk penalty grows fast: quadratic! even with small kappa, it will outgrow the linear part

in graphs 1 node system produces: 1800 TPS (lambda)

5% contention 2% crosstalk

at 4 nodes -> 72% of ideal throughput

USL: TIPPING POINT

$$N_{max} = \lfloor \sqrt{\frac{1-\sigma}{\kappa}} \rfloor$$

max concurrency exhibiting the max throughput

RELATIONSHIPS TO QUEUEING THEORY ?

- USL demonstrated to map to machine repairman model (closed queues)
- contention adds queueing delay on access to shared resources
- crosstalk increases service time

- remember queueing delay (and length, Little Law) is non linear function uf utilisation
- crosstalk: service time increases but not due to queueing delay
- QT assumes service time not dependent of concurrency or load
- QT can explain retrograde scalability

HOW TO USE THE USL ?

- capture system throughput at various load
- retrofit to the USL, get lambda, sigma, kappa
- then get the max throughput the system can have

WHAT ABOUT LATENCY ?

- system performance = throughput (what we care about)
- request performance = latency (what client cares about)
- Little's law allows to solve for latency

$$R(N) = \frac{1 + \sigma(N-1) + \kappa N(N-1)}{\lambda}$$

relationship between throughput and latency response time follows the square of concurrency!

USE

- predict system maximum throughput for a given latency
- capacity planning (USL best case scenario)
- improve system scalability (why is this system degrading much more than USL ?)

USL is just a macro model, might not work for everything (ie some systems are superlinear) USL can't see everything (ie network capacity)

SUMMARY

- Better know Lille's Law
- QT is hard and not intuitive
- Residence time/queue length asymptotical (hockey stick)
- USL math is intuitive -> capacity planning
- USL black-box analysis friendly
- USL can explain system sub-performance (too much queueing, too much crosstalk)

References:

http://people.revoledu.com/kardi/tutorial/Queuing/Que
What-Is.html
http://irh.inf.unideb.hu/~jsztrik/education/16/SOR_Mair
https://github.com/VividCortex/approx-queueing-
theory https://speakerdeck.com/drqz/m-a-new-view-
of-an-old-queue
https://www.slideshare.net/vividcortex/quantifying-
scalability-with-the-usl
https://github.com/VividCortex/approx-queueing-
theory/blob/master/sakasegawa-1977.pdf
http://www.perfdynamics.com/Manifesto/USLscalability