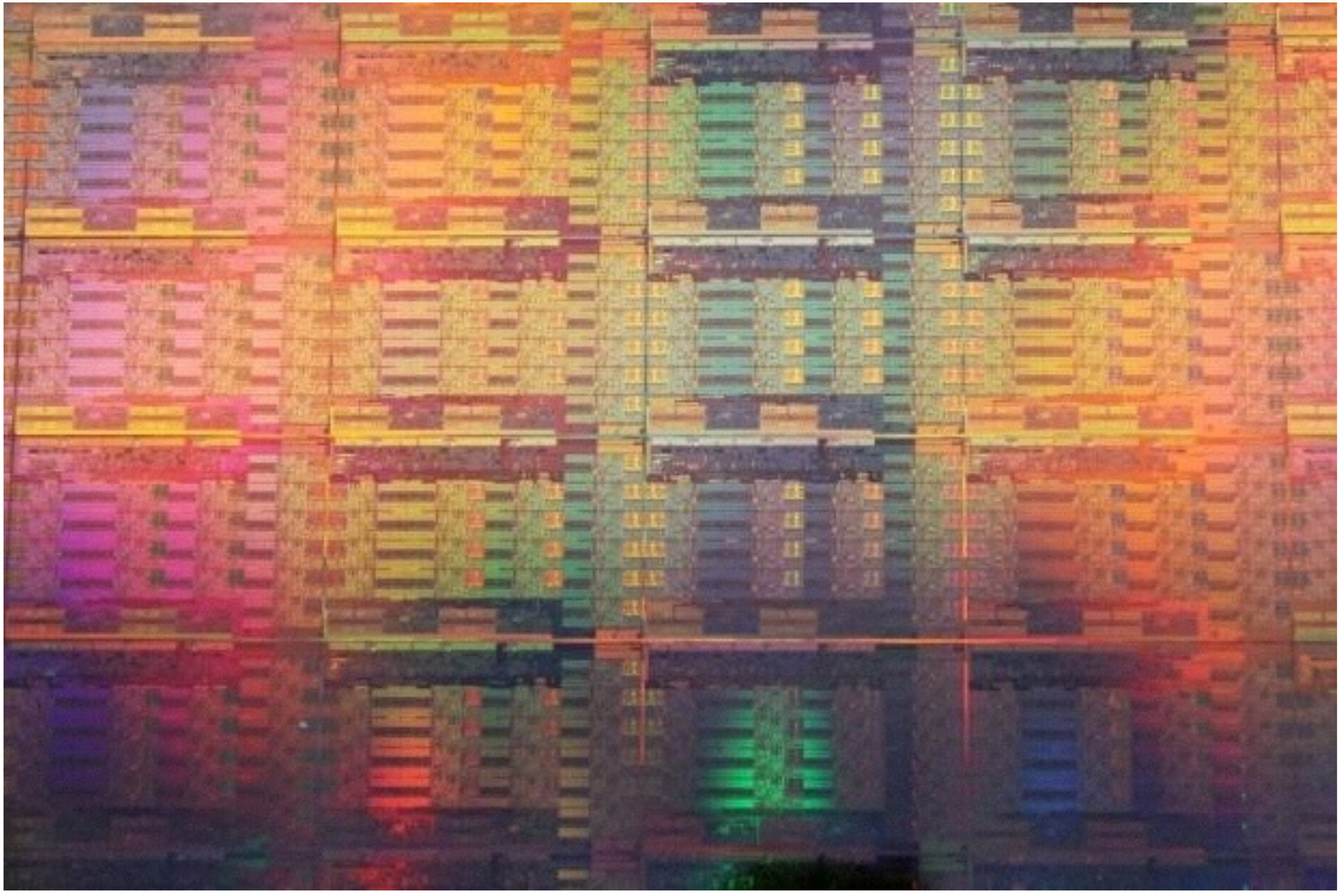
# Reactive Systems

# *Why now?*

# Electronic Commerce Era

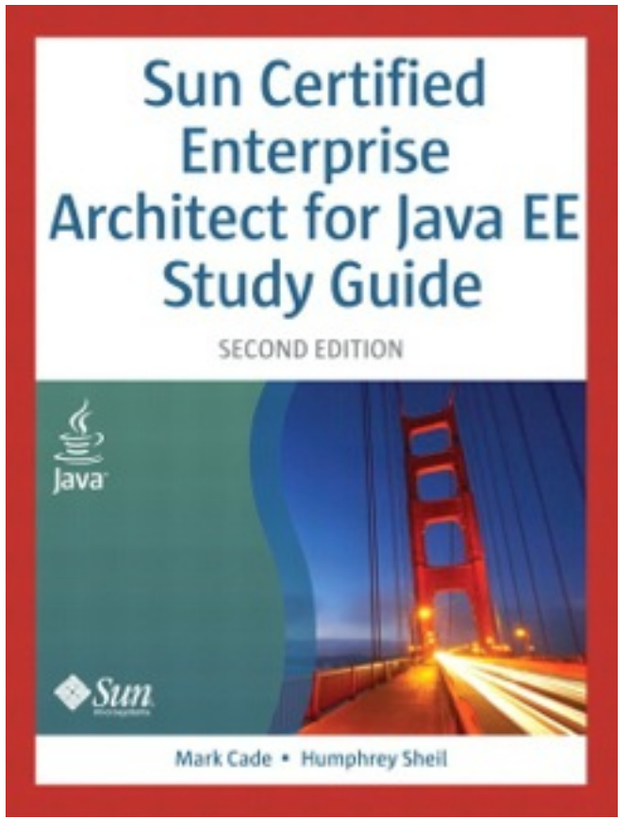
# Multicore Era



# Cloud Era

# Backlash to the BOFH Era

# Rails, JEE, or X are just not good enough!





# Rails, JEE, or X are just not good enough!



## Simultaneous Invention/Evolution

***Successful systems patterns are being "discovered"***

# Desirable System Properties

***Responsive***
***Resilient***
***Elastic***
***-***
***Message-Driven***

# *What's in a name?*

# re·act·ive

adjective \rē-'ak-tiv\

: **done in response to a problem or situation**

: **reacting to problems when they occur instead of doing something to prevent them**

Source: http://www.merriam-webster.com/

*http://www.reactivemanifesto.org/*

# Responsive

# re·spons·ive

adjective \ri-'spän(t)-siv\

: **reacting in a <u>desired</u> or positive way**

: **<u>quick</u> to react or respond**

Source: http://www.merriam-webster.com/


KINGFISHER
ROYAL CHALLENGE

## How to be Responsive?

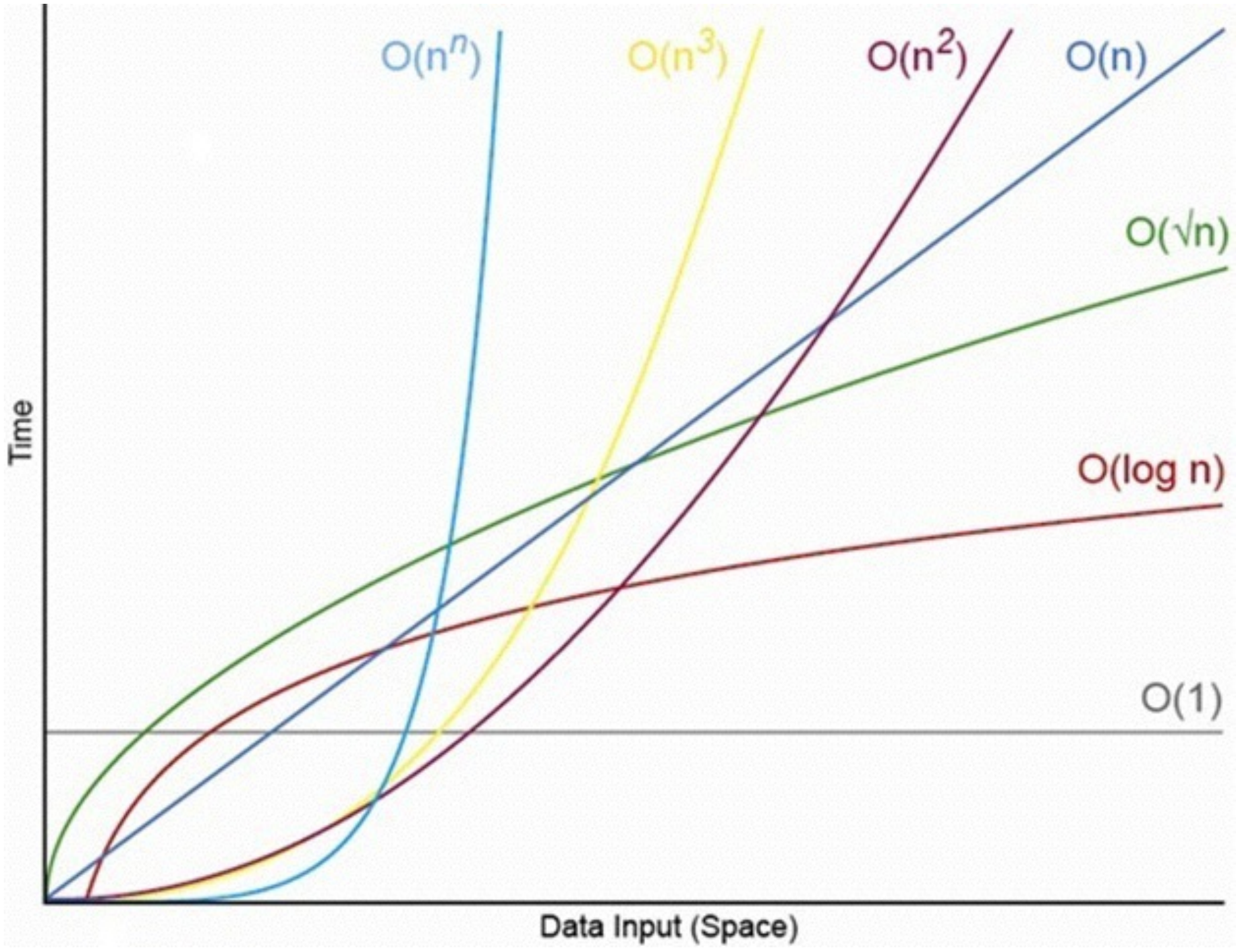
1. Be *Deterministic*
2. Offer good *Service Times*
3. Go *Parallel* to divide work

# *1. Deterministic*

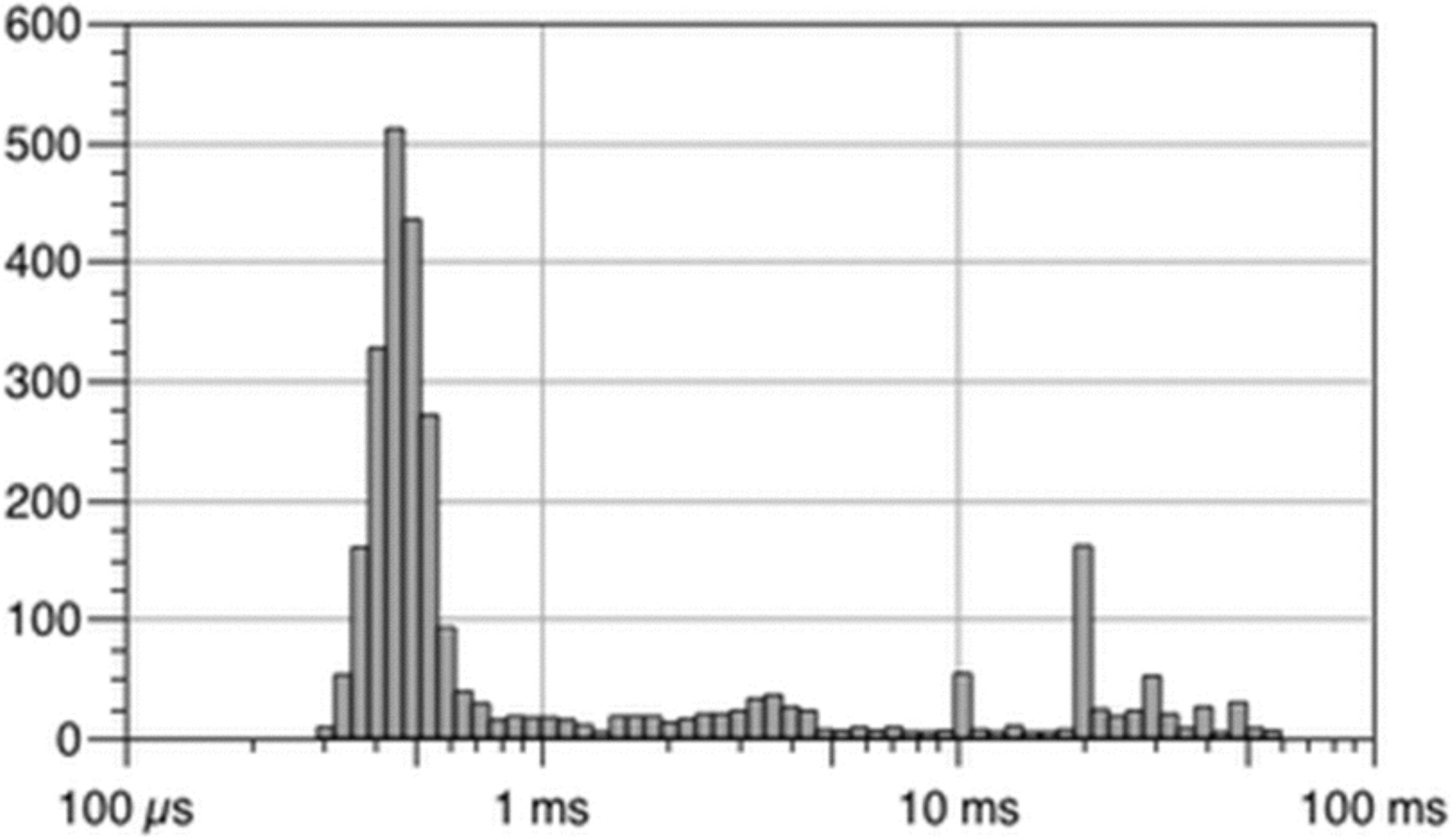
# *=>*

# *Order of Algorithms*

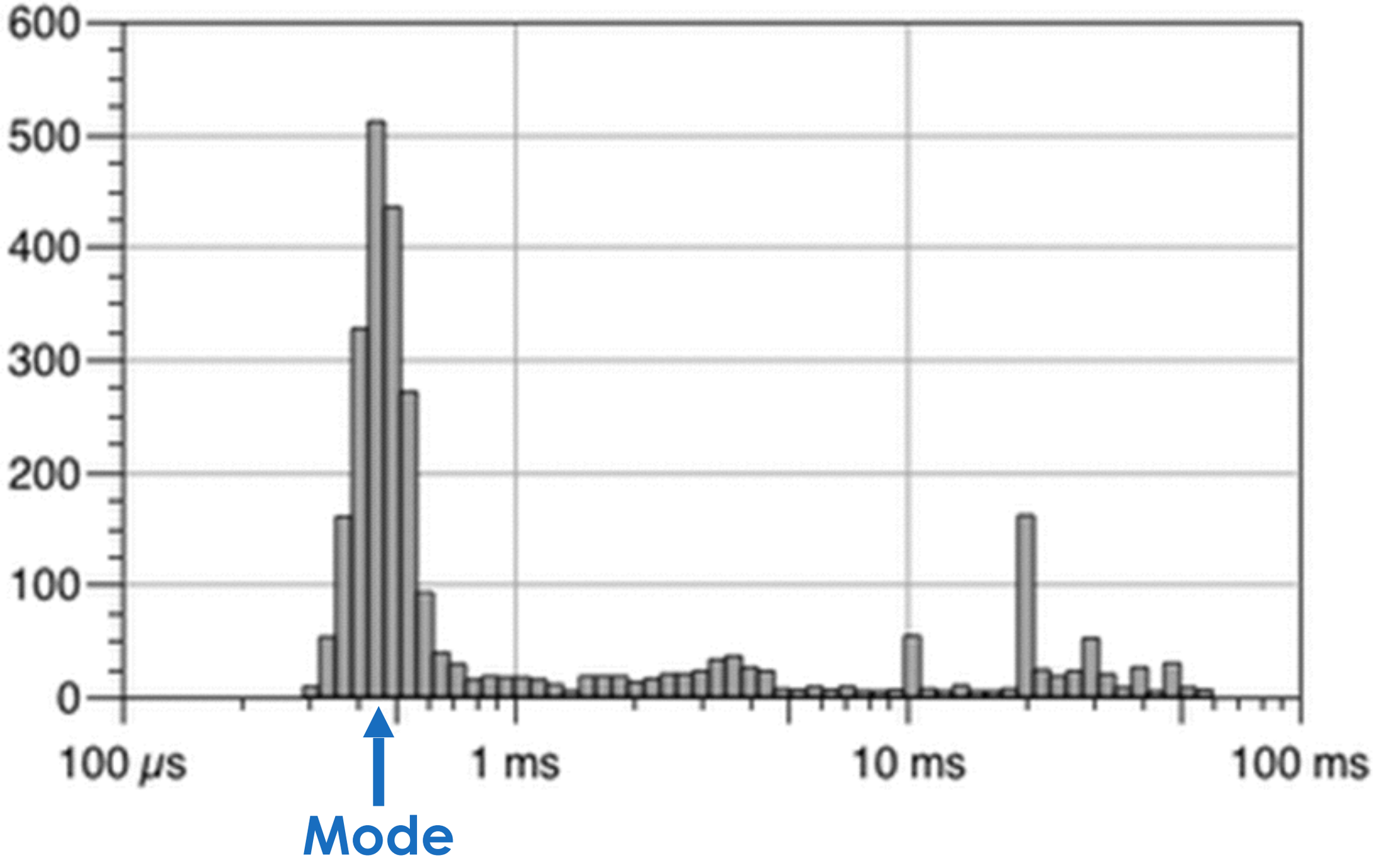
# Order of Algorithms



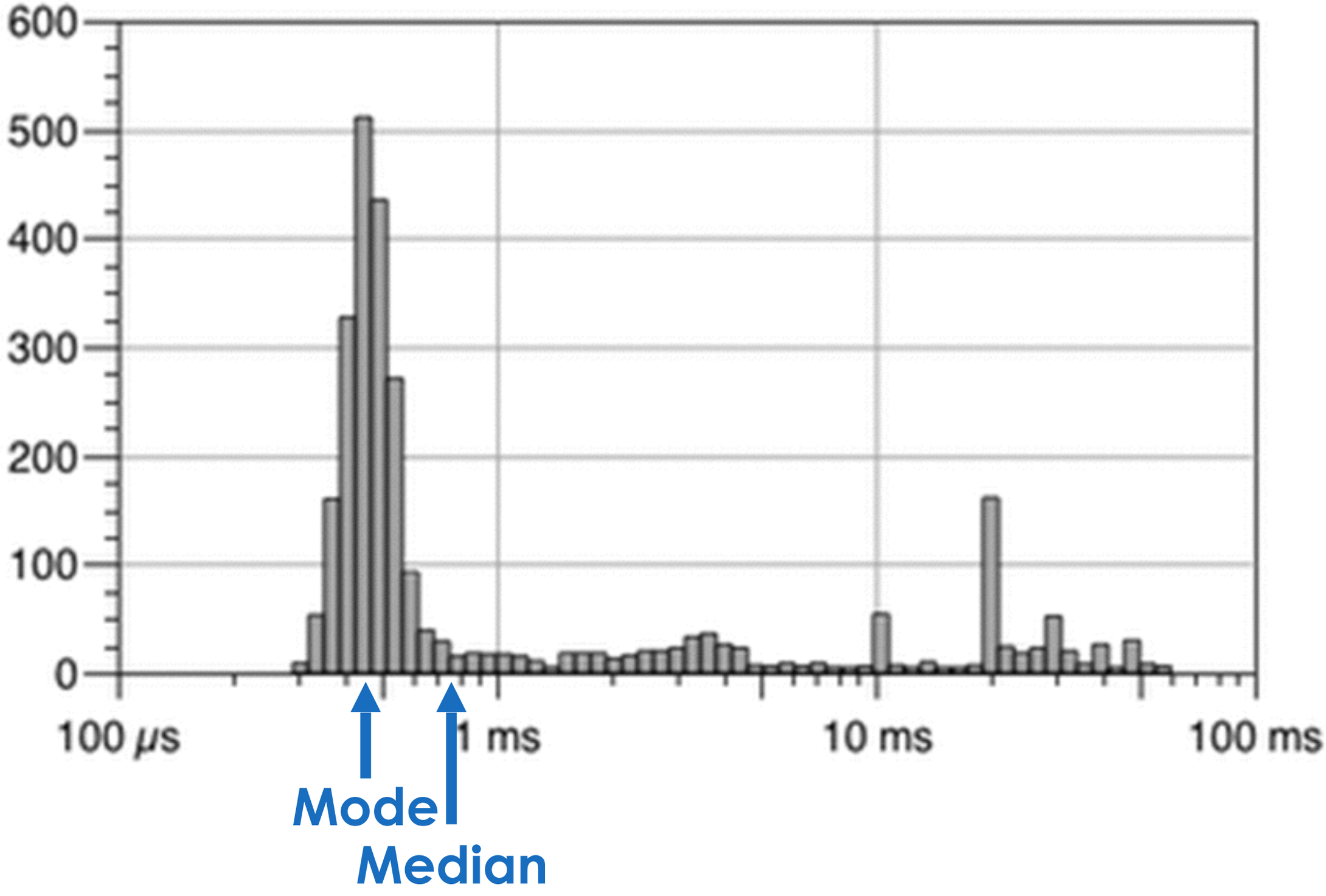
# Latency Histograms



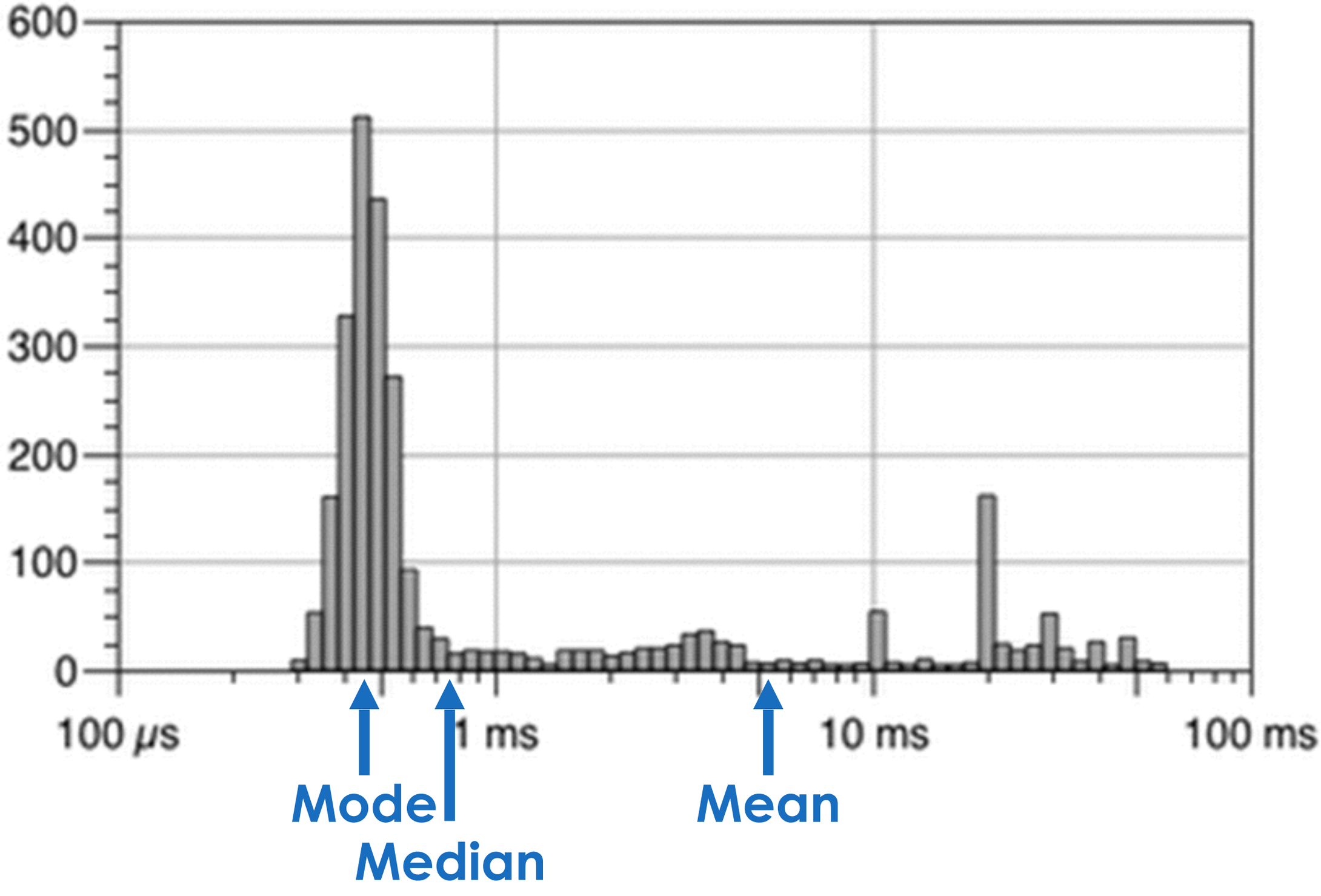
# Latency Histograms



# Latency Histograms



# Latency Histograms



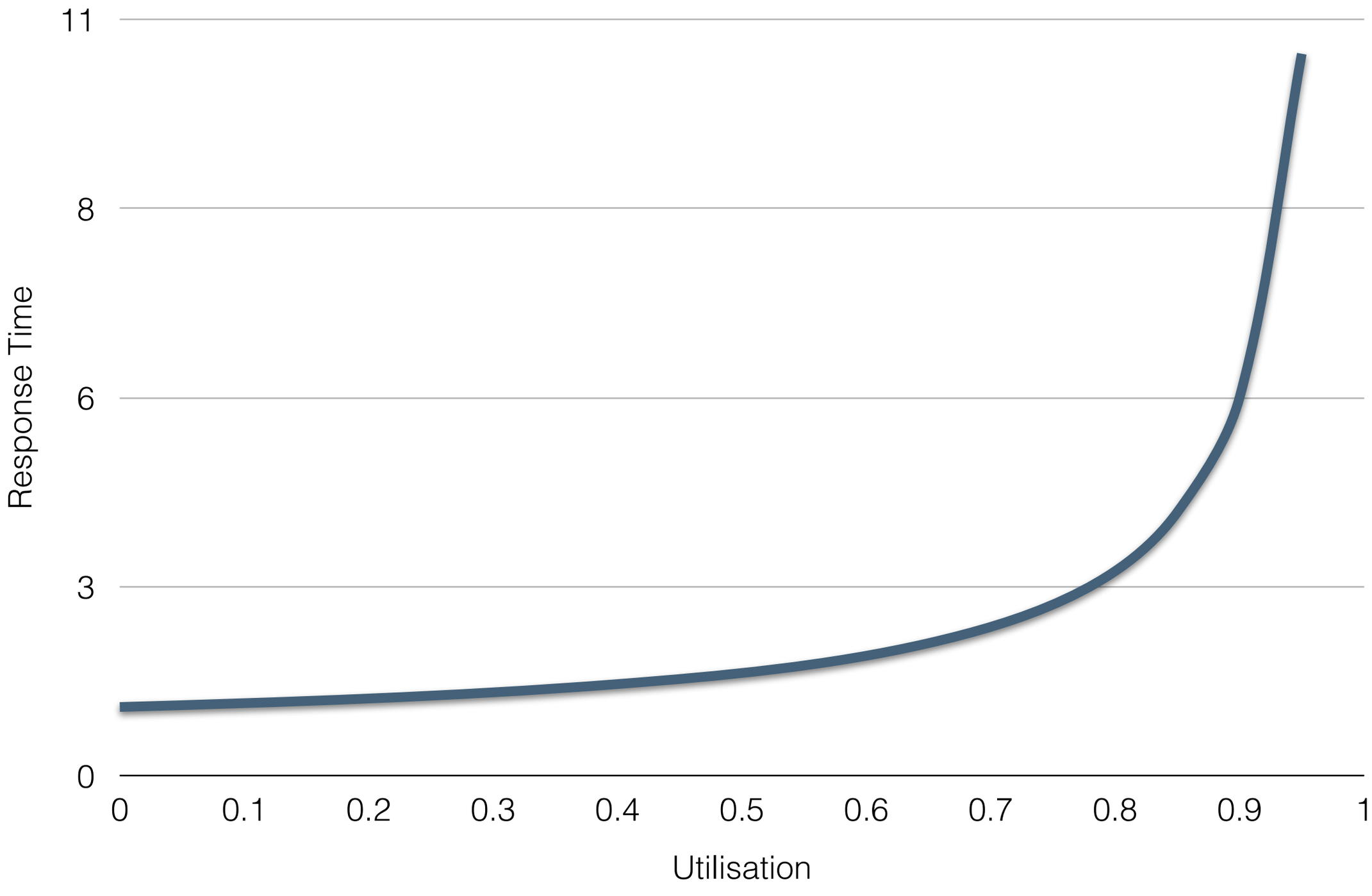
# Don’t be a Resource Hog

# *2. Service Time*

# *=>*

# *Utilisation*

# Queuing Theory



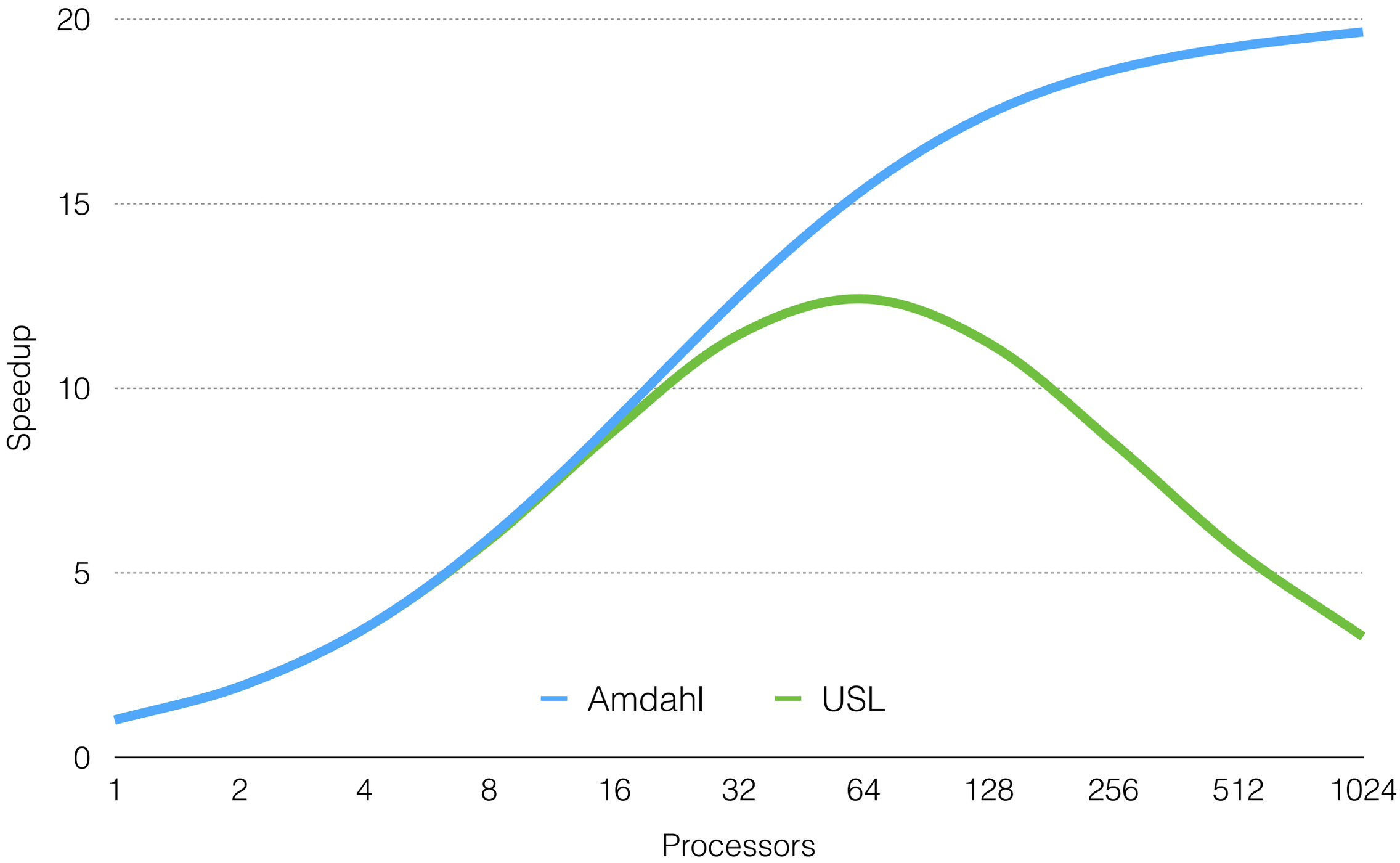
# *Limit queue lengths to control response times*

# *3. Parallel*

# =>

# *Contention & Coherence*

# Universal Scalability Law



***Shared <u>mutable</u> state is the crystal meth of concurrent systems***

***Break work into batches and pipelines with no contention***

# *Are you a Hipster or a Geek?*

# *Learn to Measure*

# *&*

# *Apply Science*

*"Synchronous RPC is the crack cocaine of distributed programming"*

**- @mjpt777**

# Resilient

re·sil·ient *adjective* \ri-ˈzil-yənt\

: able to become strong, healthy, or successful again after something bad happens

: able to return to an original shape after being pulled, stretched, pressed, bent, etc.

[Merriam-Webster online Dictionary]

# Bad things that happen

**“Broken”**

- Computers
  - Memory
  - Disks
- Networks
  - Routers
  - Cables

**“Stretched”**

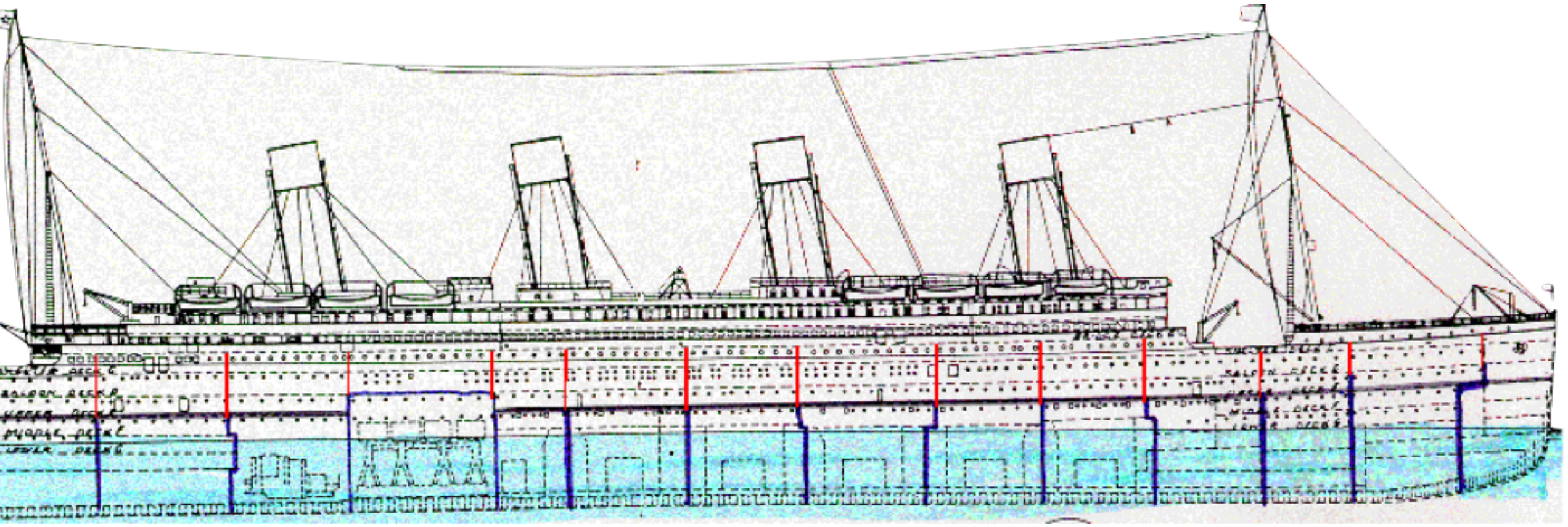
- Memory
- Compute
- I/O Load
- Storage capacity
- Congestion

**“Unforeseen”**

- Input Validation
- Configuration
- Inconsistency
- Hackers
- Just plain BUGS

*“Anything that can possibly go wrong, eventually does.”*

# Bulkheads

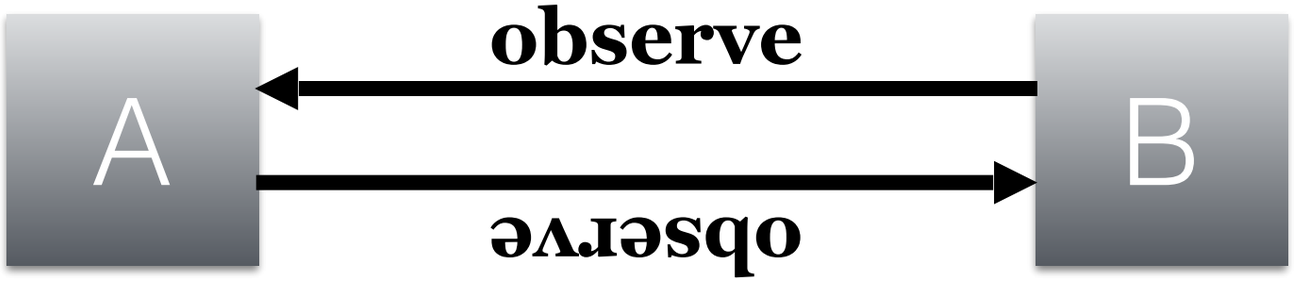


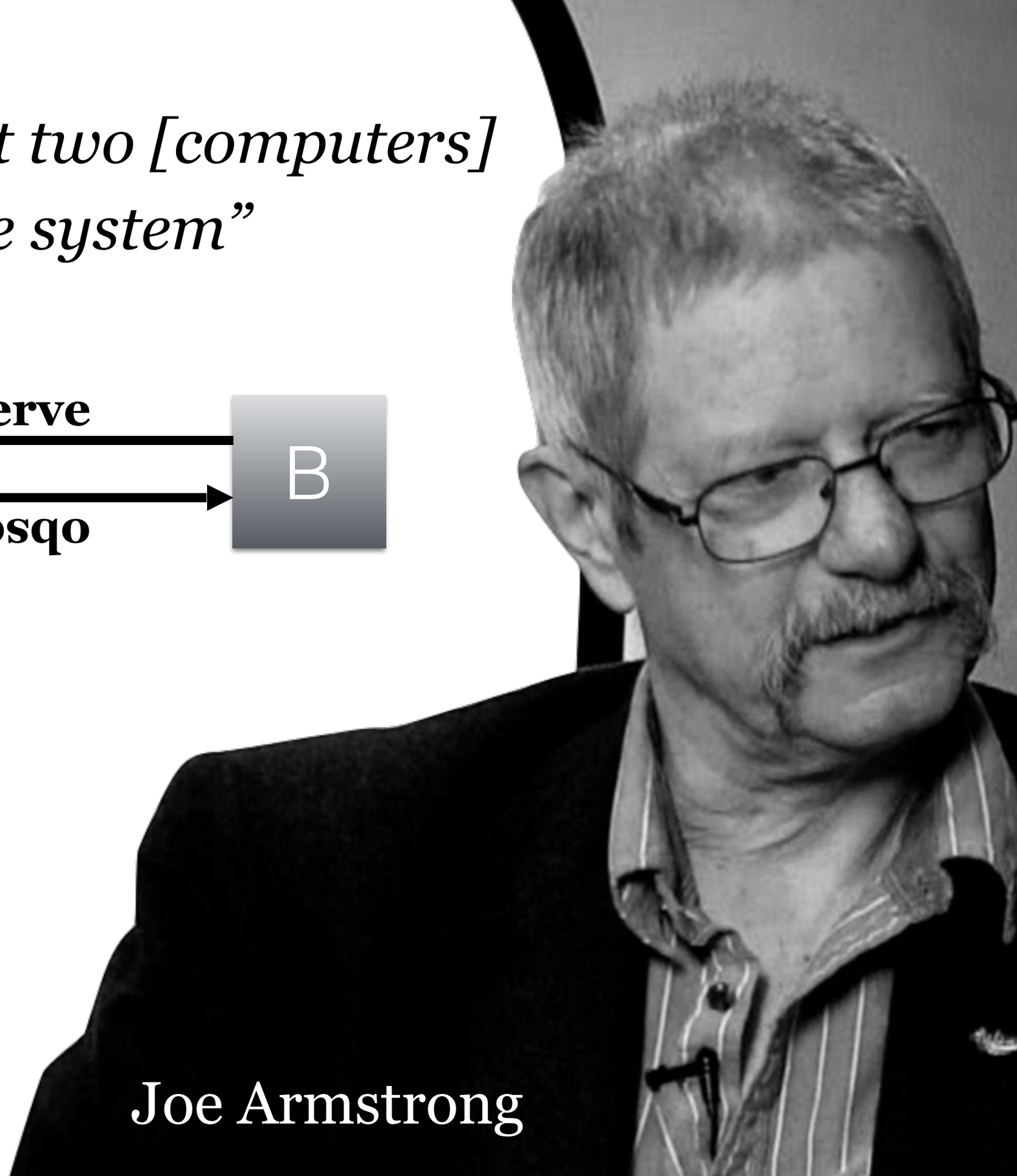
*In essence: contain faults*

# The 3 rules of resilience

1. *Isolate*,
2. *Isolate*, and
3. *Isolate*.

*"You need at least two [computers] to make a reliable system"*



Joe Armstrong

# Joe’s version of Titanic

# The 3 rules of resilience

1. *Isolate*,
2. *Isolate*, and
3. *Make faults observable.*

# Units of Isolation

1. Data Center
2. Rack / Cluster
3. Machine

---

4. Operating System Process
5. Software Component

# Most software faults are transient

When *Microsoft Word* hangs, you restart it, and move on with life.

*… this is where the Titanic analogy ends.*

# The 3 rules of resilience

1. *Isolate*,
2. *Make faults observable*,
3. *Restart*

# Units of Isolation

1. Data Center
2. Rack / Cluster
3. Machine
4. Operating System Process

---

5. Software Component

# Actors: Isolated Components

1. Encapsulated
2. Faults are handled *outside* – by another actor
3. Patterns for fault handlers are called Supervisors

# Toolbox

- Heartbeats / alive monitors
- Transactions
- Append-only file formats
- Actors / Micro-processes
- Component-local resources
- Supervisors (Erlang, Akka)
- Circuit Breaker Patterns
- *and many more …*

# KEEP CALM AND LET IT CRASH

… because you know it's ISOLATED

@drkrab

Elastic

# elas·tic

adjective /i-'las-tik/

: **capable of ready change or easy expansion or contraction**

: **able to be changed**

Source: http://www.merriam-webster.com/

# Outline

1. Scaling
2. Elastic
3. Profit!

# Why do we need to be Elastic?

The rules of the game

**have changed**

| Apps in the 60s-90s were written for | Apps today are written for |
| --- | --- |
| Single machines | Clusters of machines |
| Single core processors | Multicore processors |
| Expensive RAM | Cheap RAM |
| Expensive disk | Cheap disk |
| Slow networks | Fast networks |
| Few concurrent users | Lots of concurrent users |
| Small data sets | Large data sets |
| Latency in seconds | Latency in milliseconds |

# Outline

1. Scaling
2. Elastic
3. Profit!

# WAIT! What is **Scalability**?

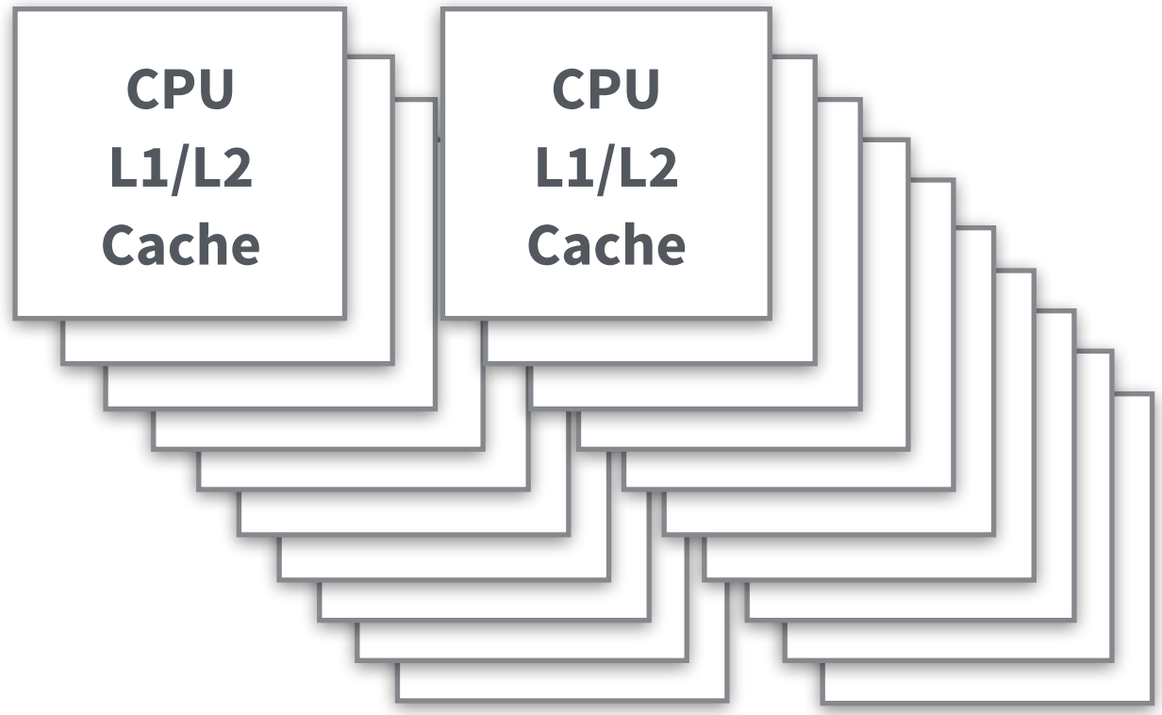
# Scalability vs Performance

# Scale

# UP

Scale

**OUT**


CPU
L1/L2
Cache
CPU
L1/L2
Cache

Thus Scaling **Up & Out** is practically

# the same thing

$cale
**DOWN**
$cale
**IN**

# be

# ASYNC

# don’t

# BLOCK

# divide

# CONQUER

pipe

LINE

# share

# NOTHING

# location TRANSPARENCY

Typesafe

# obtain

# METRICS

Typesafe

# Outline

1. Scaling
2. Elastic
3. Profit!

# reactive ELASTICITY

Typesafe

# predictive ELASTICITY

become

ELASTIC

# Outline

1. Scaling
2. Elastic
3. Profit!

# Message Driven

It’s not what Message Passing provides.

It’s what it makes *harder* or even *impossible*.

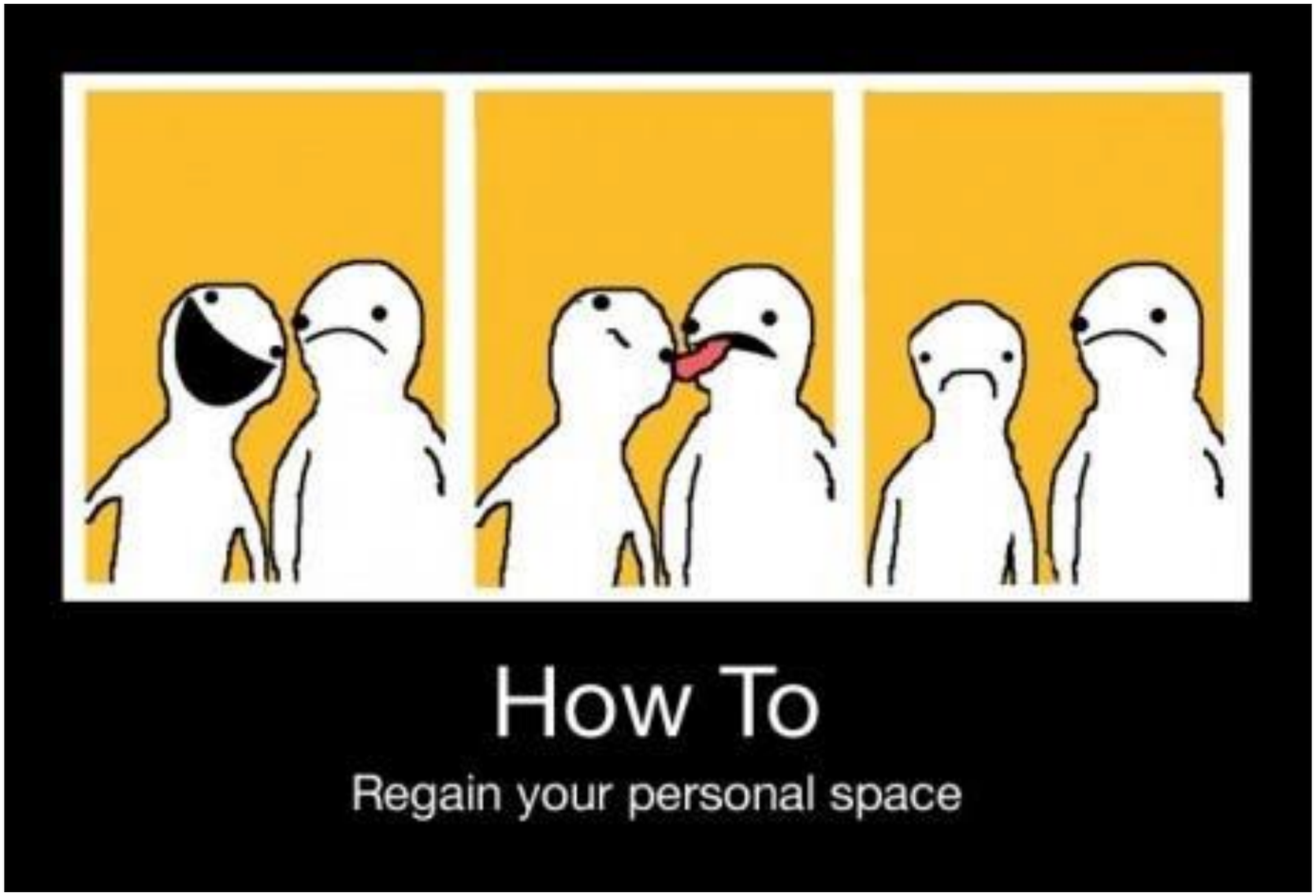
pro·to·col ***noun*** \'prō-tə-ˌkȯl, -ˌkōl, -ˌkäl, -kəl\

. . .

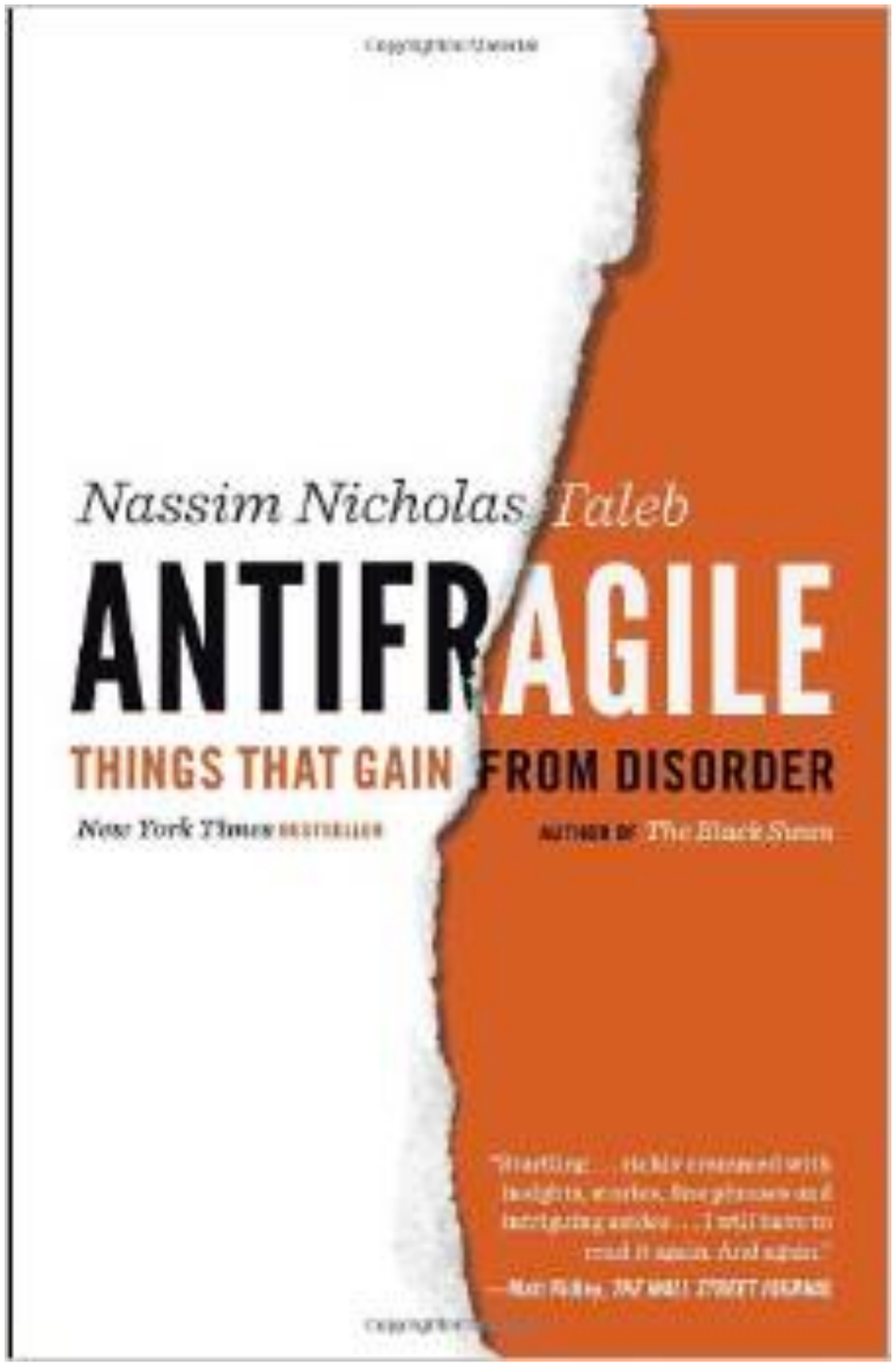
**:** a set of conventions governing the *treatment* and especially the *formatting* of data in an electronic communications system <network *protocols*>

. . .

**:** a code prescribing strict adherence to correct etiquette and precedence (as in diplomatic exchange and in the military services) <a breach of *protocol*>




# Boundaries are Good!


Nassim Nicholas Taleb
ANTIFRAGILE
THINGS THAT GAIN FROM DISORDER
New York Times
AUTHOR OF The Black Swan

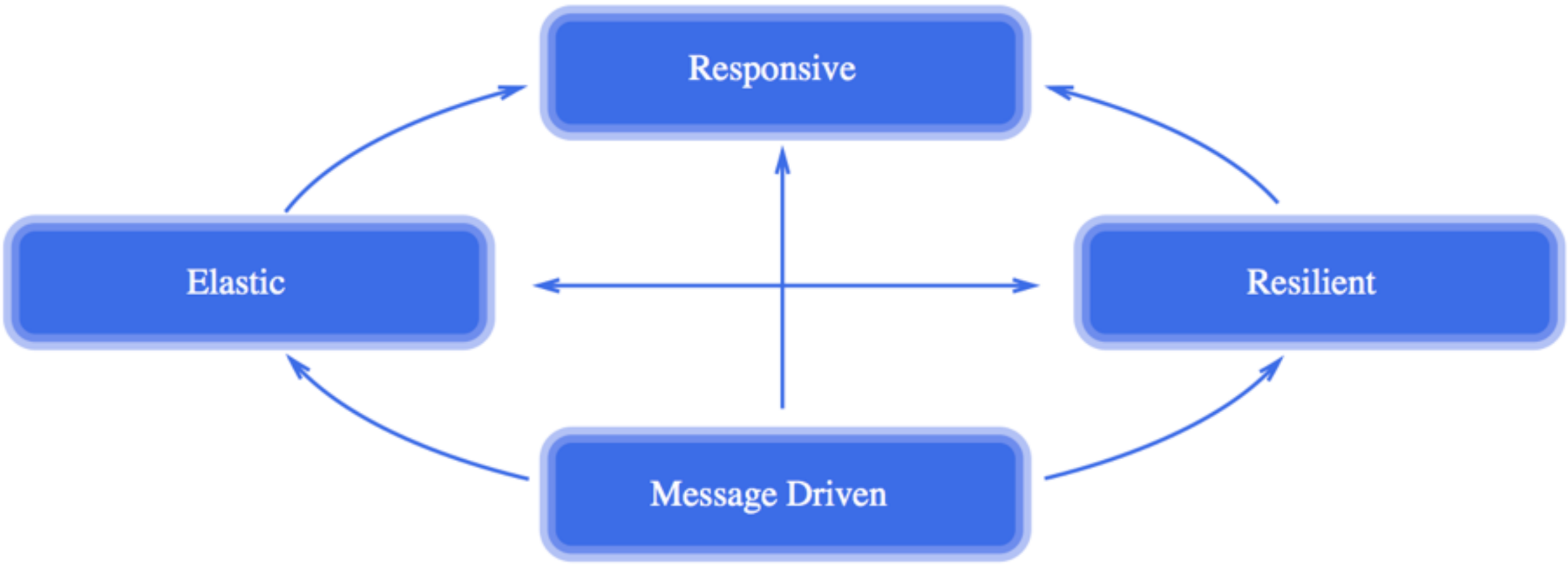
Binary Boundary

Asynchronous Boundary

# *Forced* Decoupling & Separation of Concerns

Event Ordering Implied Correlation

Errors are Messages




# Message Driven facilitates other traits

Asynchrony
Built-in

Contention
is Evil

# Responsive

Amdahl's Law
& USL

Decoupling Forces
Responsive Design

Supervisors &
Hierarchies

Boundaries
enforce bulkheads

# Resilient

Localized Errors

Live Piecemeal
Upgrade

Decoupling key to Scale

Contention is Evil

# Elastic

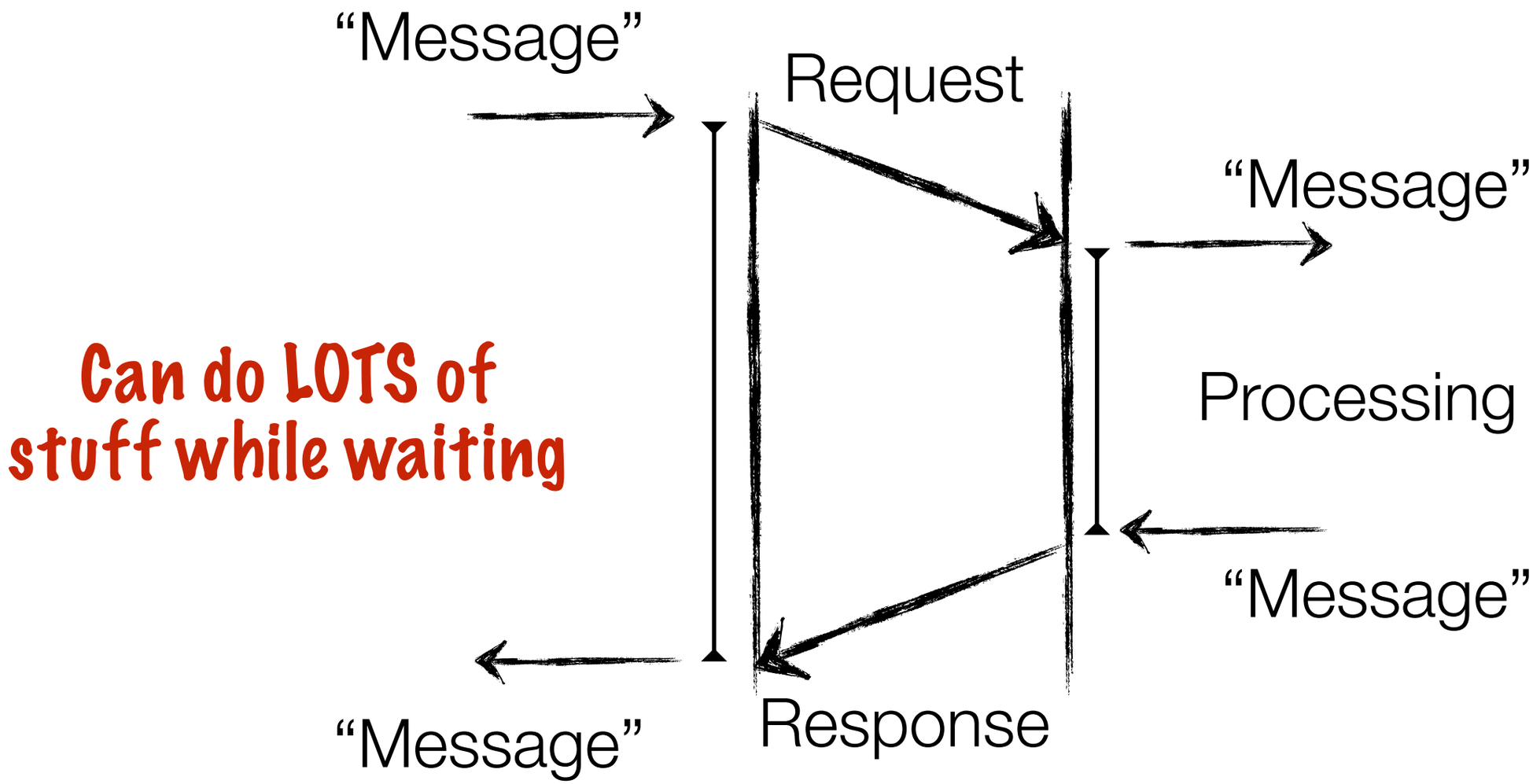
Amdahl's Law & USL

Spin Up, Down, In, & Out

Even traditional blocking operations can be decoupled

# REST *is* Reactive!

# HTTP



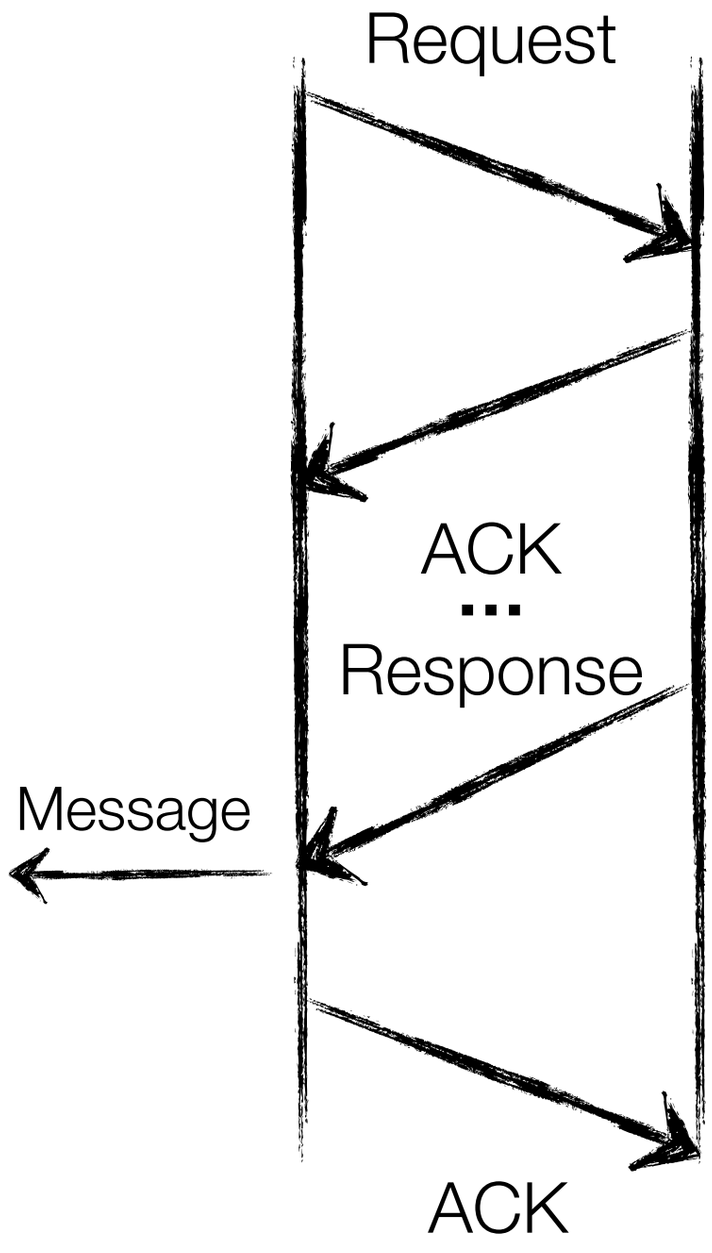

Can do LOTS of stuff while waiting

But what about correlating responses with requests?!
Don't I need to wait?

# Web Services

http://en.wikipedia.org/wiki/List_of_web_service_specifications

No, *seriously*, lots of these!!




Sync Request

Sync Response

But... Async Request/Response... kinda


# Errors are Messages

Got an error, so let's send a new error message back...

# Mistakes & BCP

TCP RST behavior...

# Reactive Streams