Responding in a timely manner

Martin Thompson - @mjpt777


KINGFISHER
ROYAL

Hard Real-time

Soft Real-time


0.18
33.41
3.1
2.33%
+0.19
0.22
95.67
6.4
1.22%
+2.5
0.19
18.94
1.1
2.33%
+0.45
0.28
19.44
0.5
-5.22%
-1.56
0.35
19.43
0.6
1.46%
+0.73
0.46
26.17
0.2
0.44%
+0.43
0.23
16.11
1.7
4.12%
+0.14
0.77
38.19
4.1
-1.78%
-1.22
0.32
56.12
1.1
2.36%
0.67
21.96
0.9
1.98%
0.97
17.55
7.1
2.36%
0.17
49.19
3.9
-3.97%

Squidgy Real-time

The Unaware

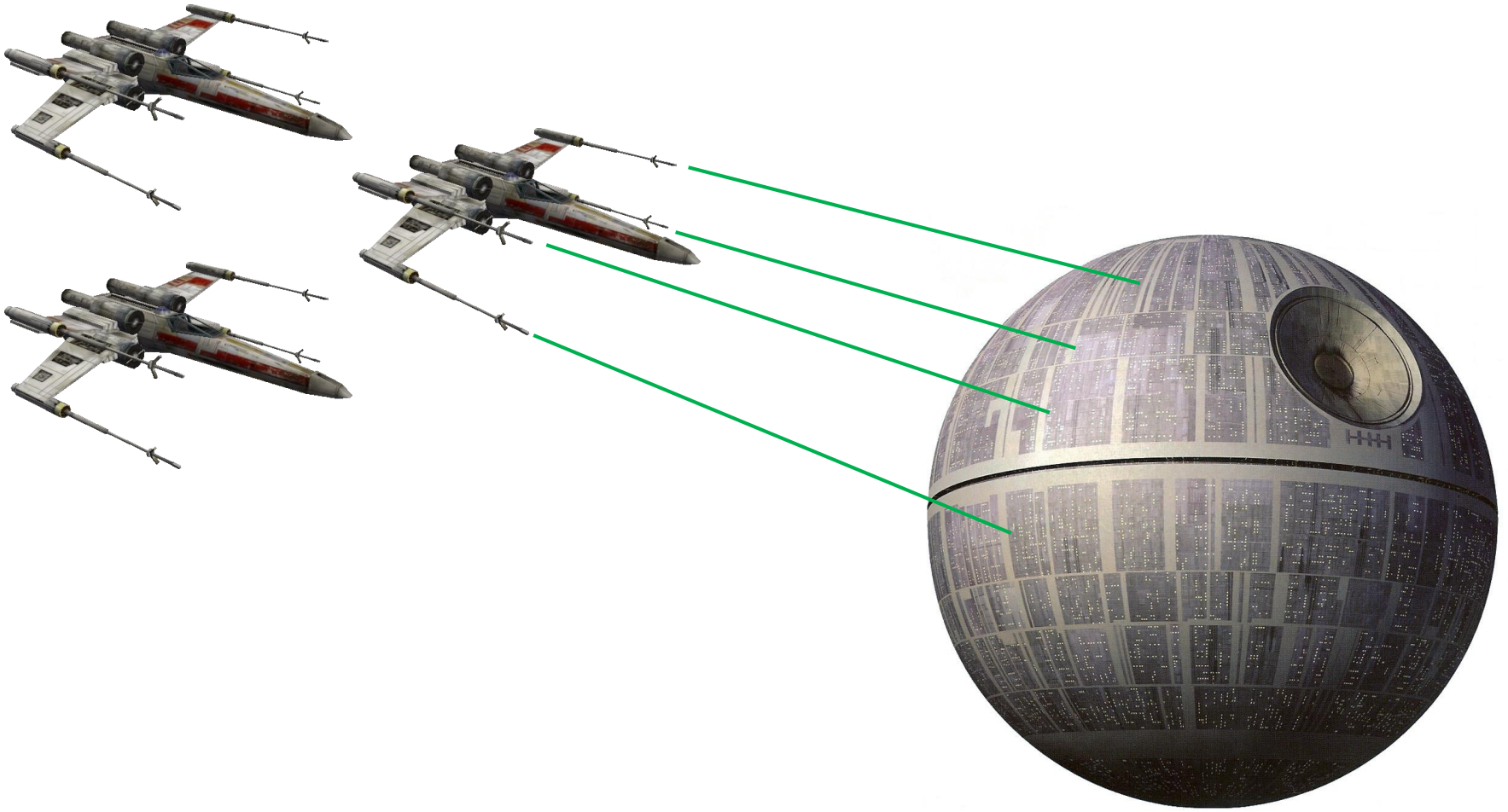

FAIL

1. How to Test and Measure
2. A little bit of Theory
3. A little bit of Practice
4. Common Pitfalls
5. Useful Algorithms and Techniques

Test & Measure

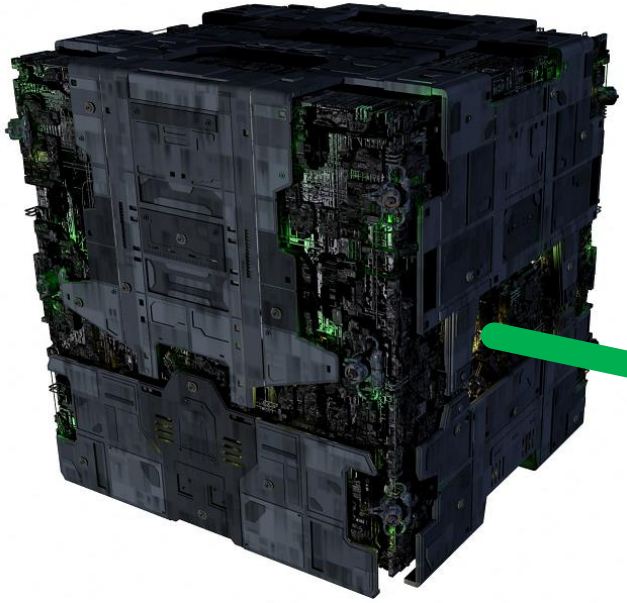
System Under Test

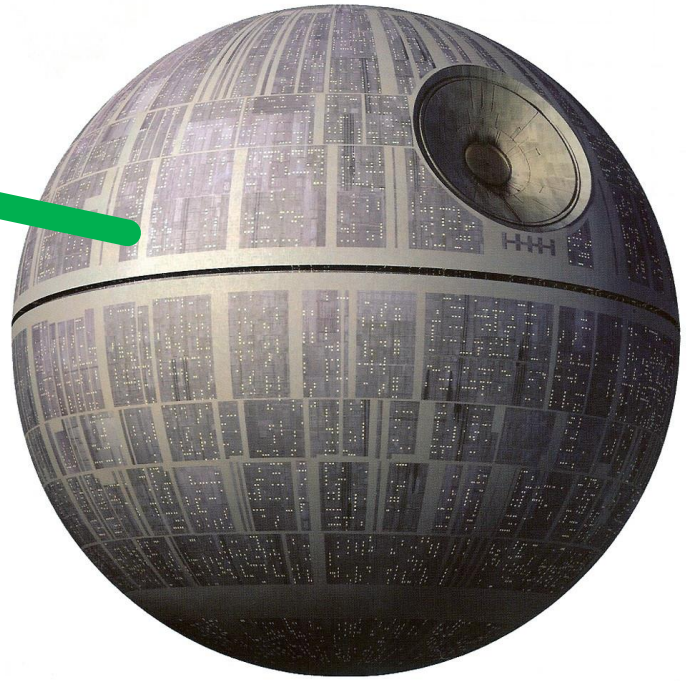
Distributed Load Generation Agents



System Under Test

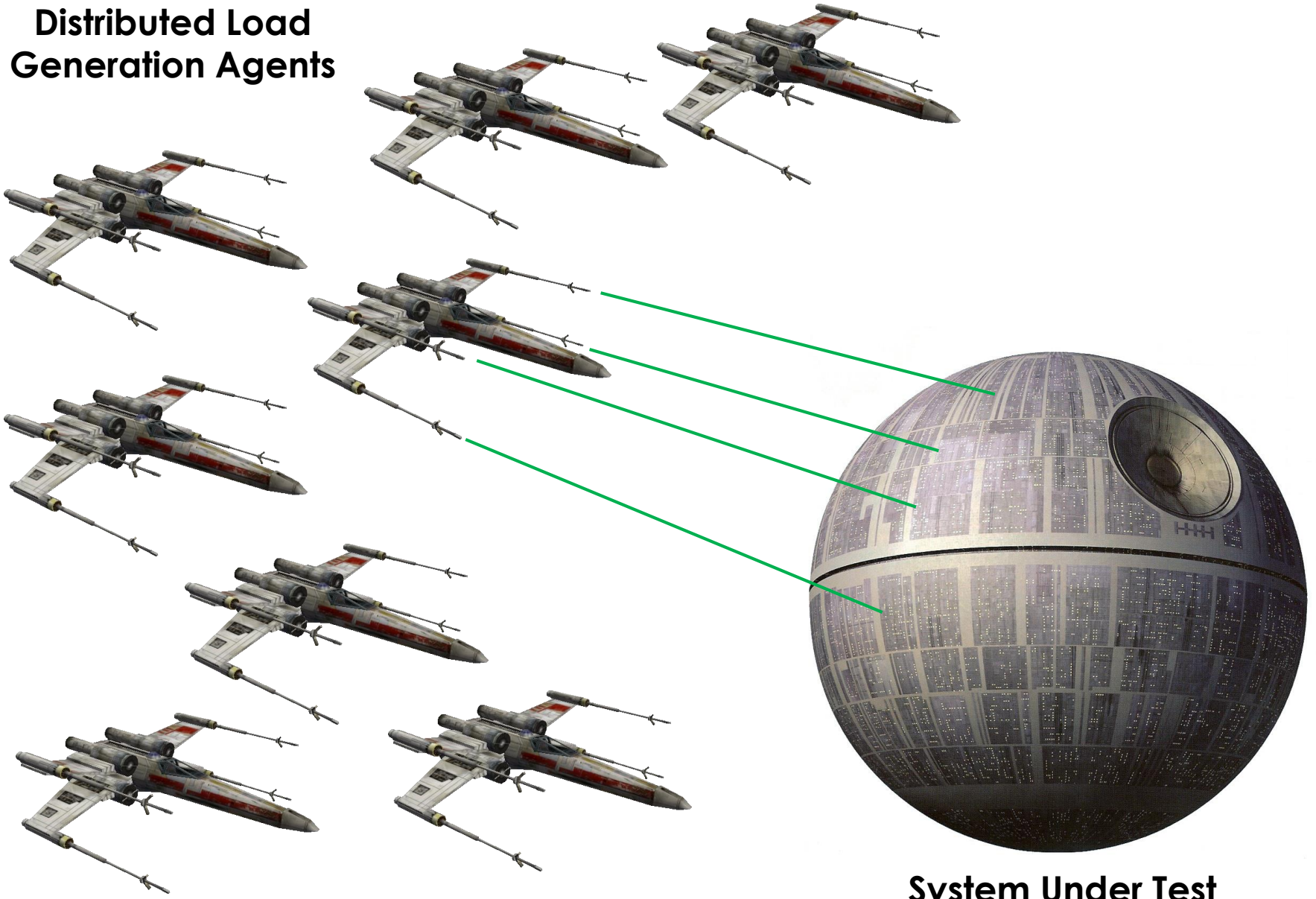
Distributed Load Generation Agents



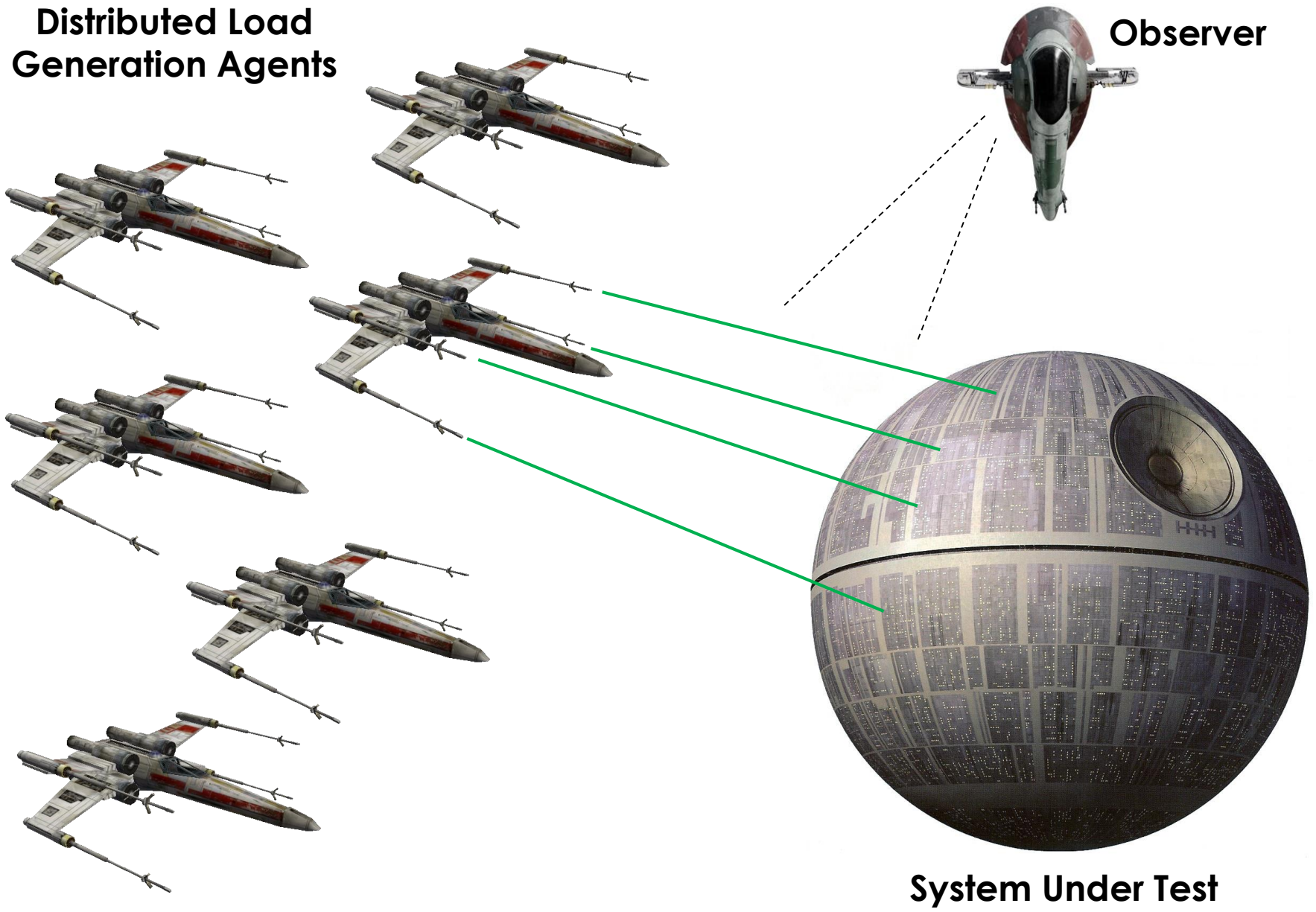


System Under Test

Distributed Load Generation Agents



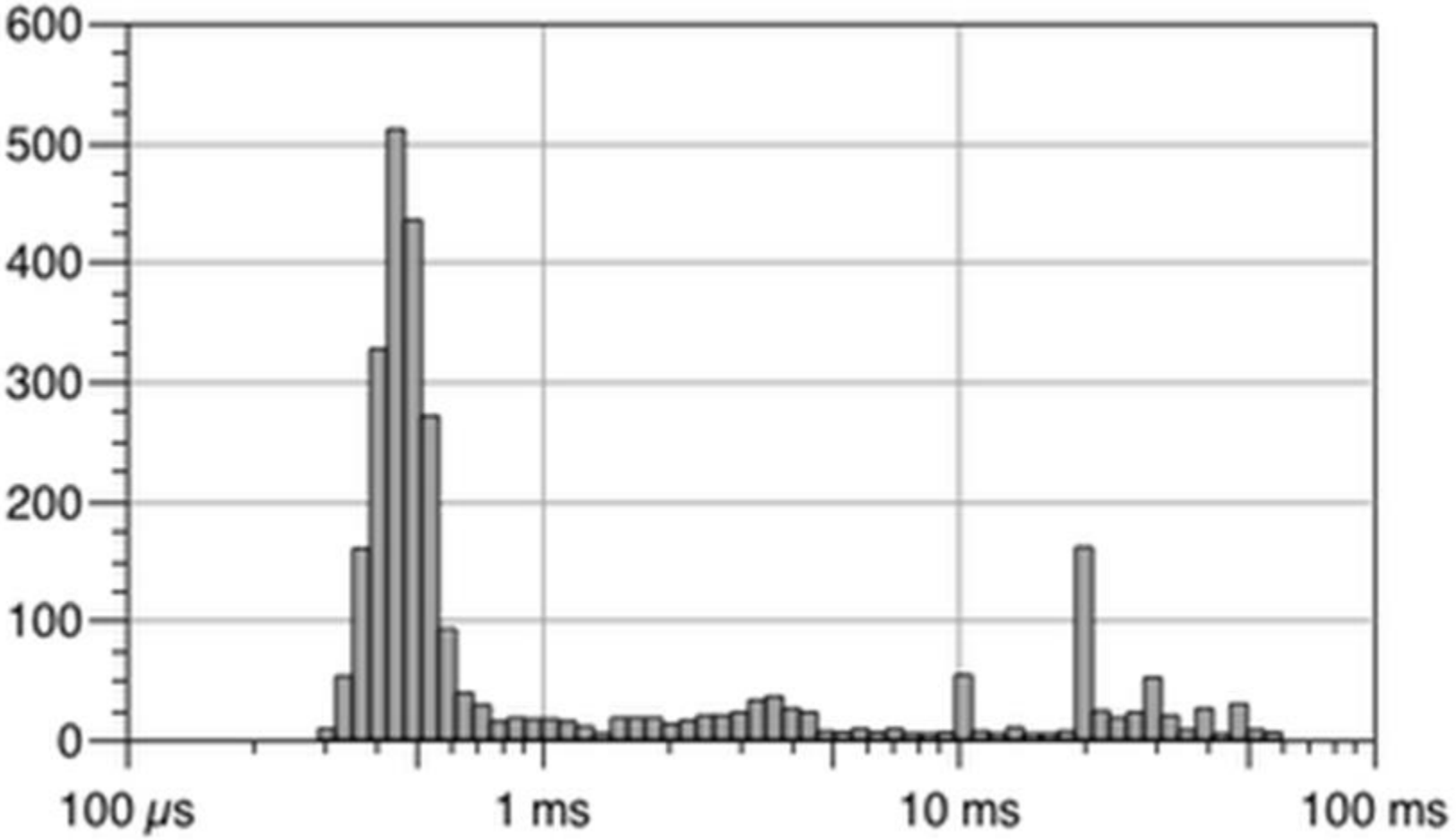
System Under Test


Distributed Load Generation Agents
Observer
System Under Test

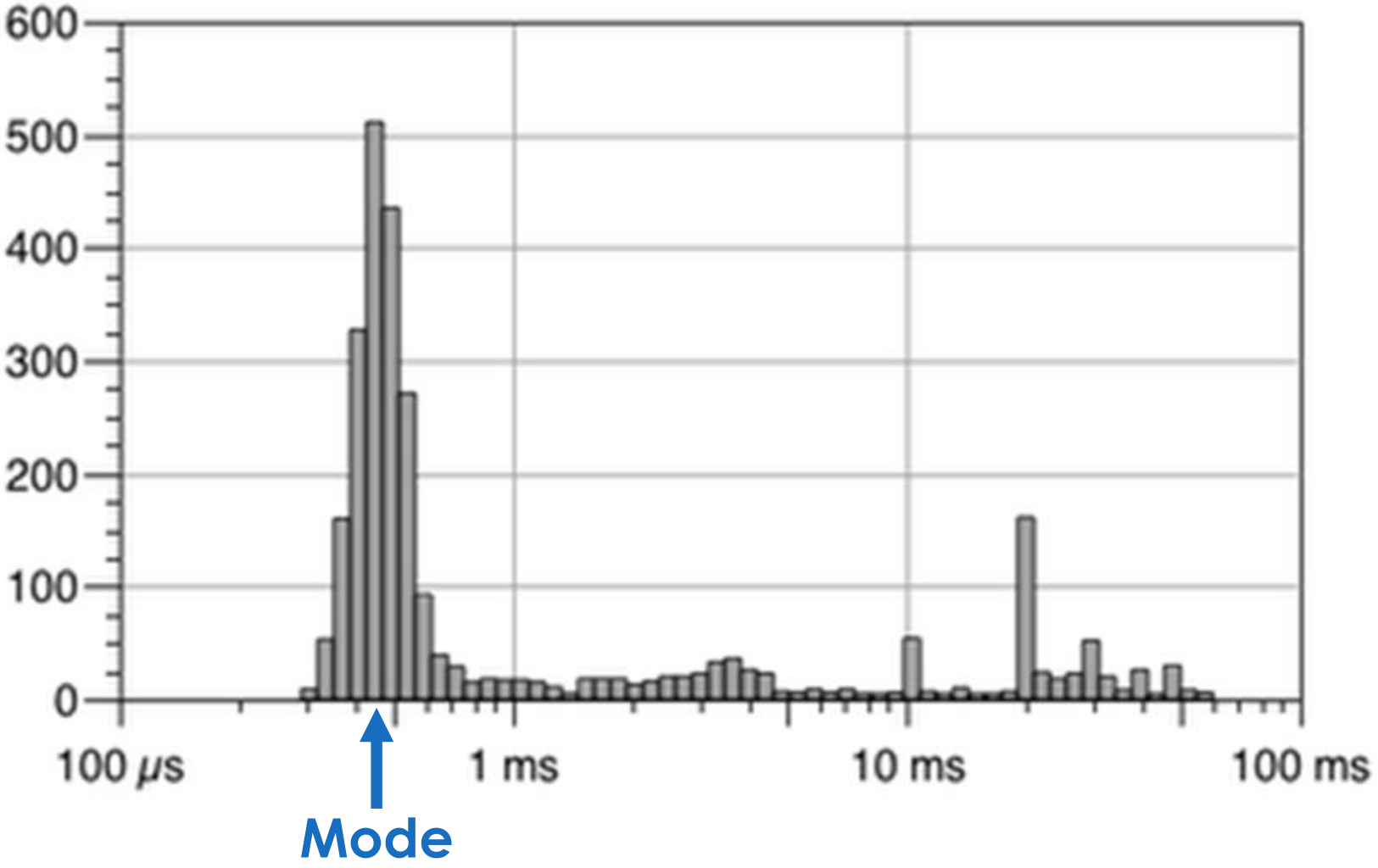
Pro Tip: Setup a continuous performance testing environment

Pro Tip: Record Everything

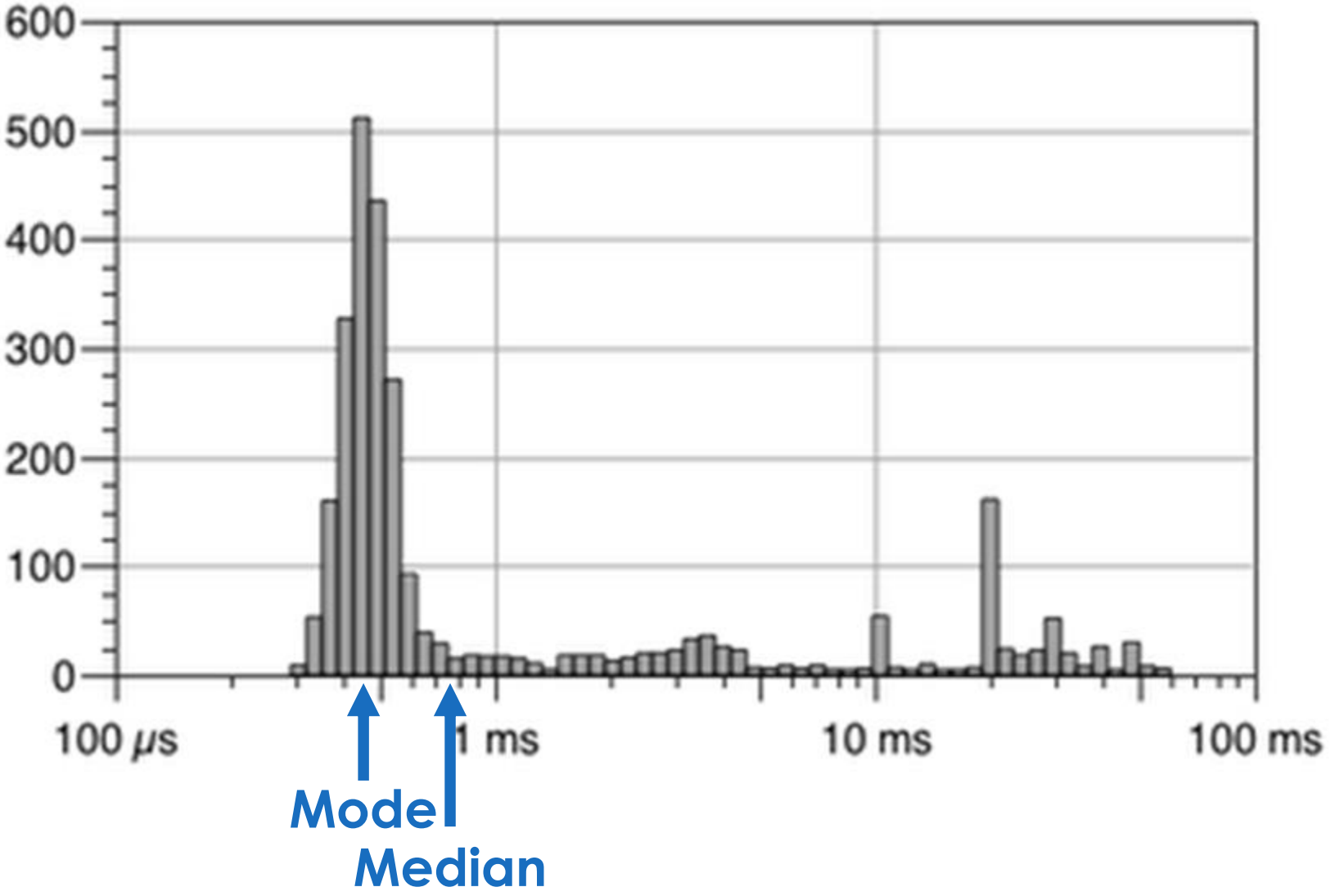
Latency Histograms



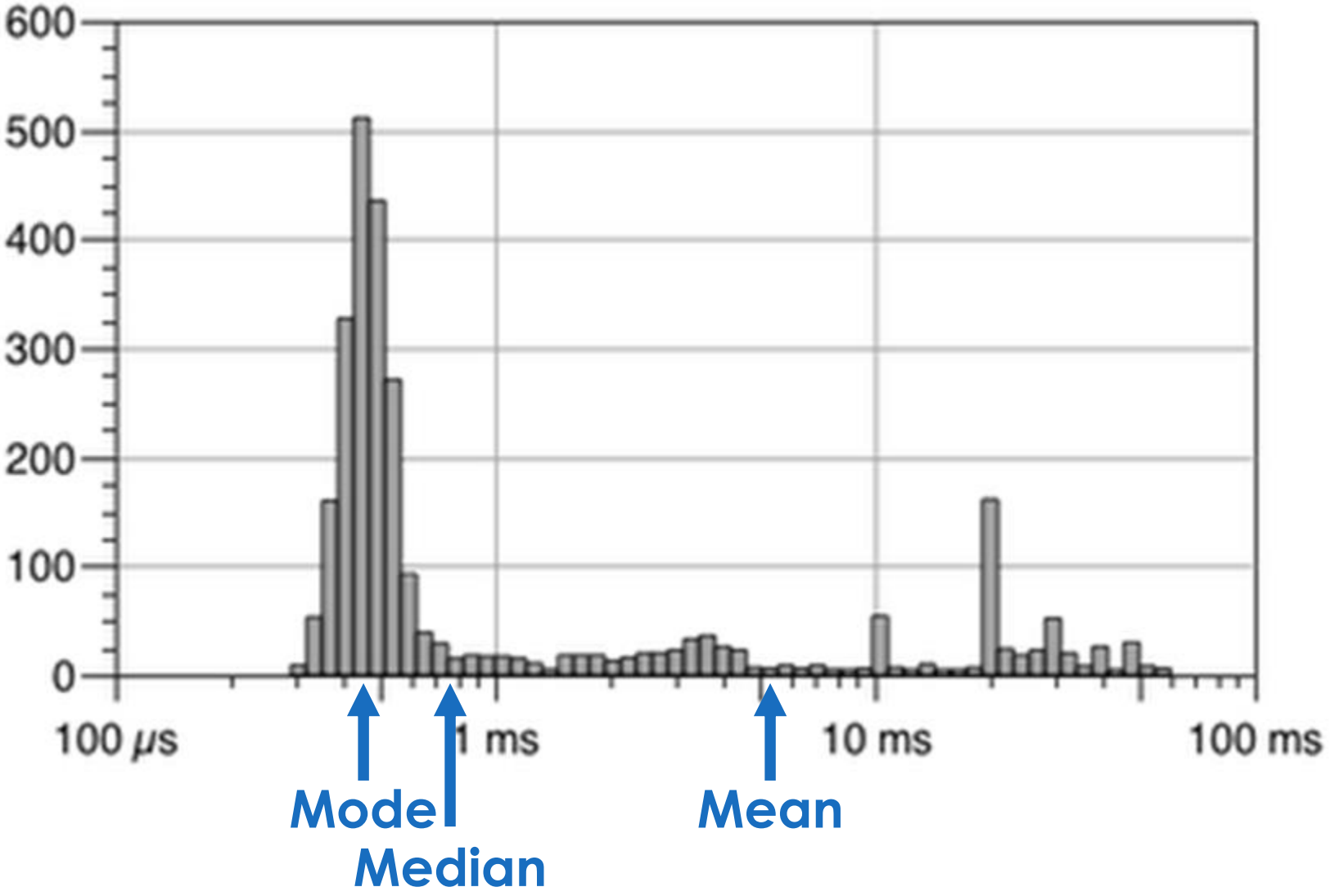
Latency Histograms



Latency Histograms



Latency Histograms



System: 1000 TPS, mean RT 50μs

System: 1000 TPS, mean RT 50µs

What is the mean if you add in a 25ms GC pause per second?

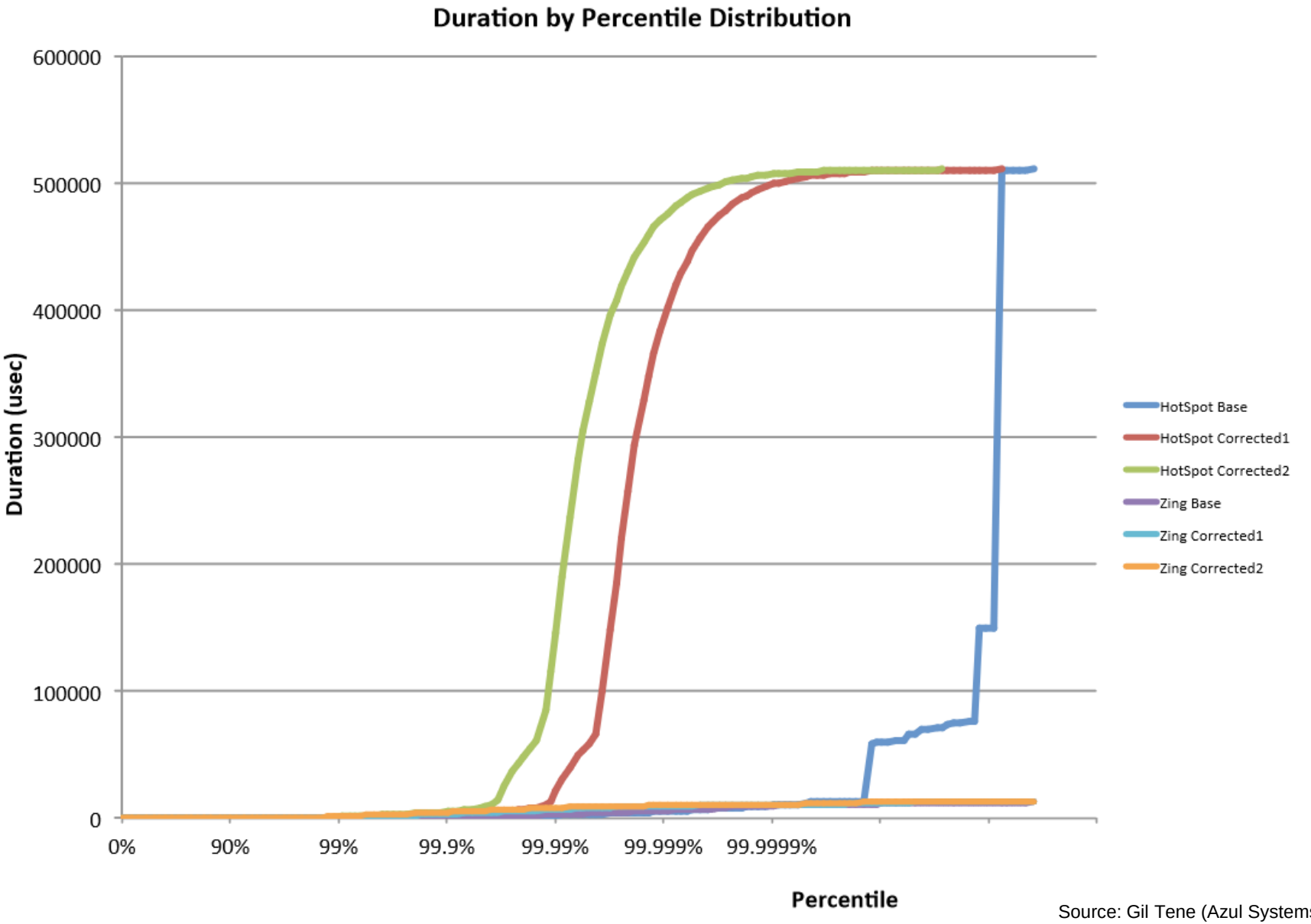
System: 1000 TPS, mean RT 50µs

What is the mean if you add in a 25ms GC pause per second?

~300µs

Forget averages,
it's all about percentiles

Coordinated Omission

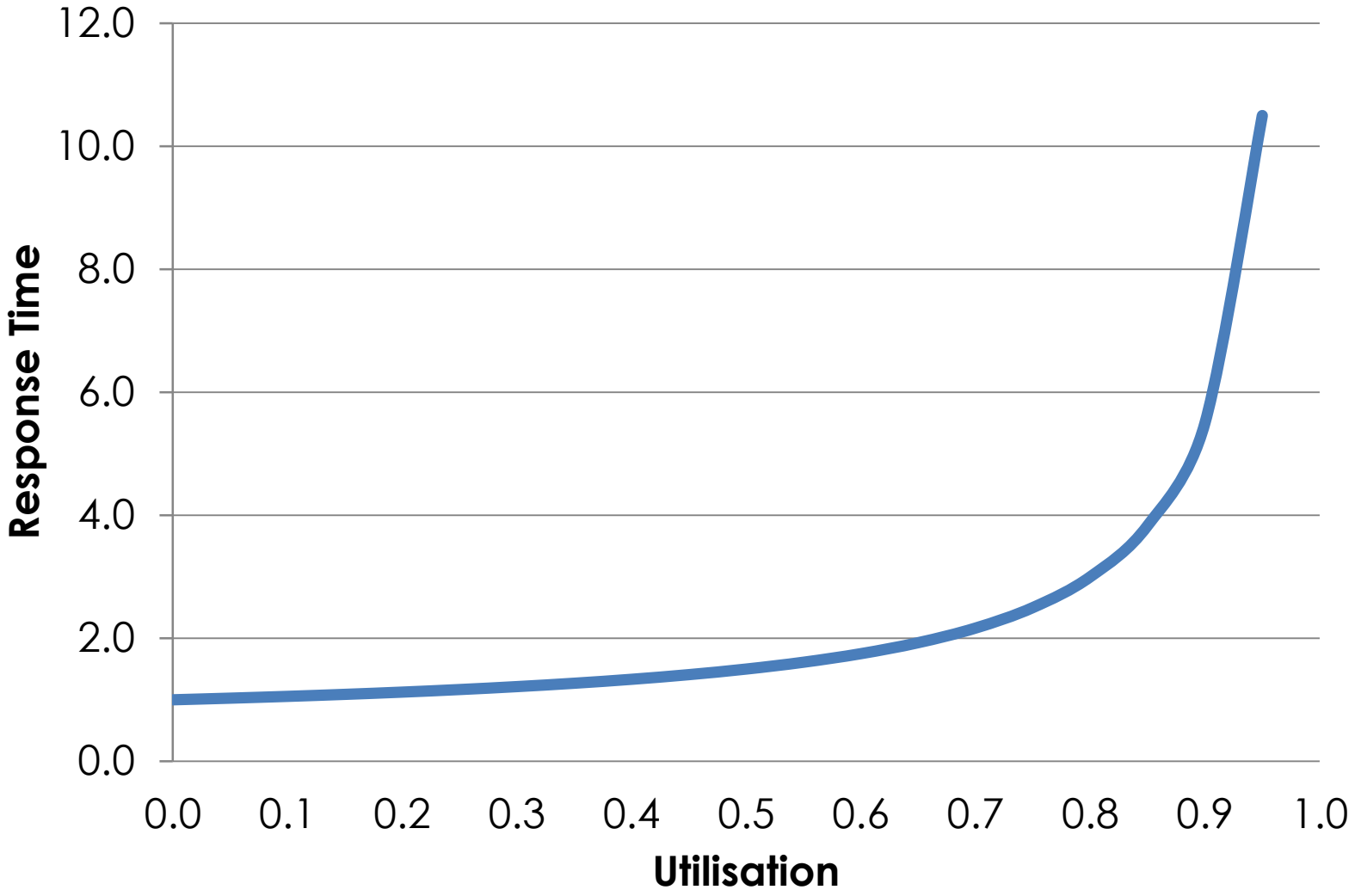


Source: Gil Tene (Azul Systems)

Pro Tip: Don’t deceive yourself

Theory

Queuing Theory



Kendall Notation

M/D/1

Queuing Theory

$$r = s(2 - \rho) / 2(1 - \rho)$$

r = mean response time

s = service time

ρ = utilisation

Queuing Theory

$$r = s(2 - \rho) / 2(1 - \rho)$$

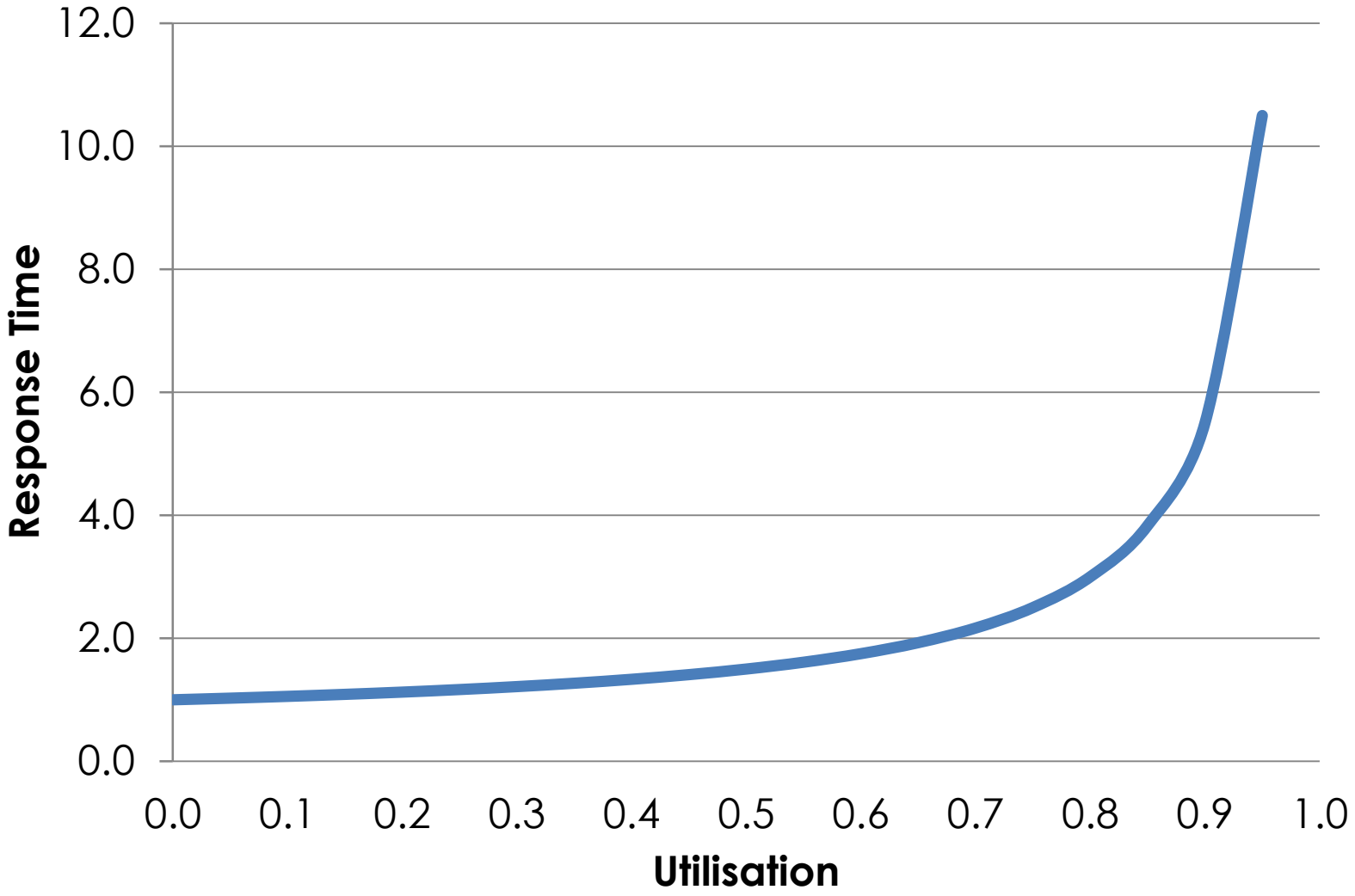
r = mean response time

s = service time

ρ = utilisation

Note: $\rho = \lambda * (1 / s)$

Queuing Theory



Pro Tip: Ensure that you have sufficient capacity

Queuing Theory

Little's Law: $L = \lambda * W$

L = mean queue length

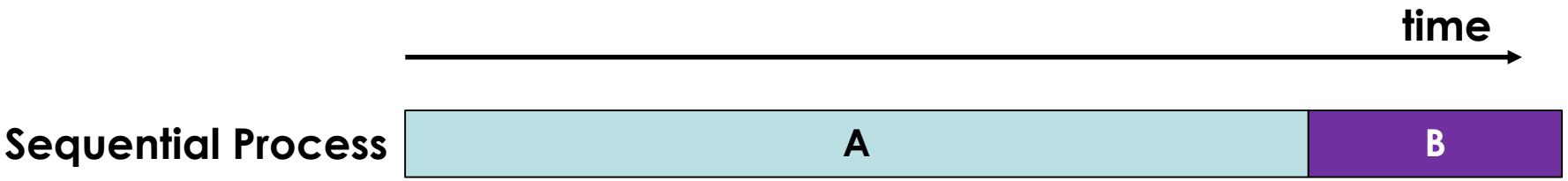
λ = mean arrival rate

W = mean time in system

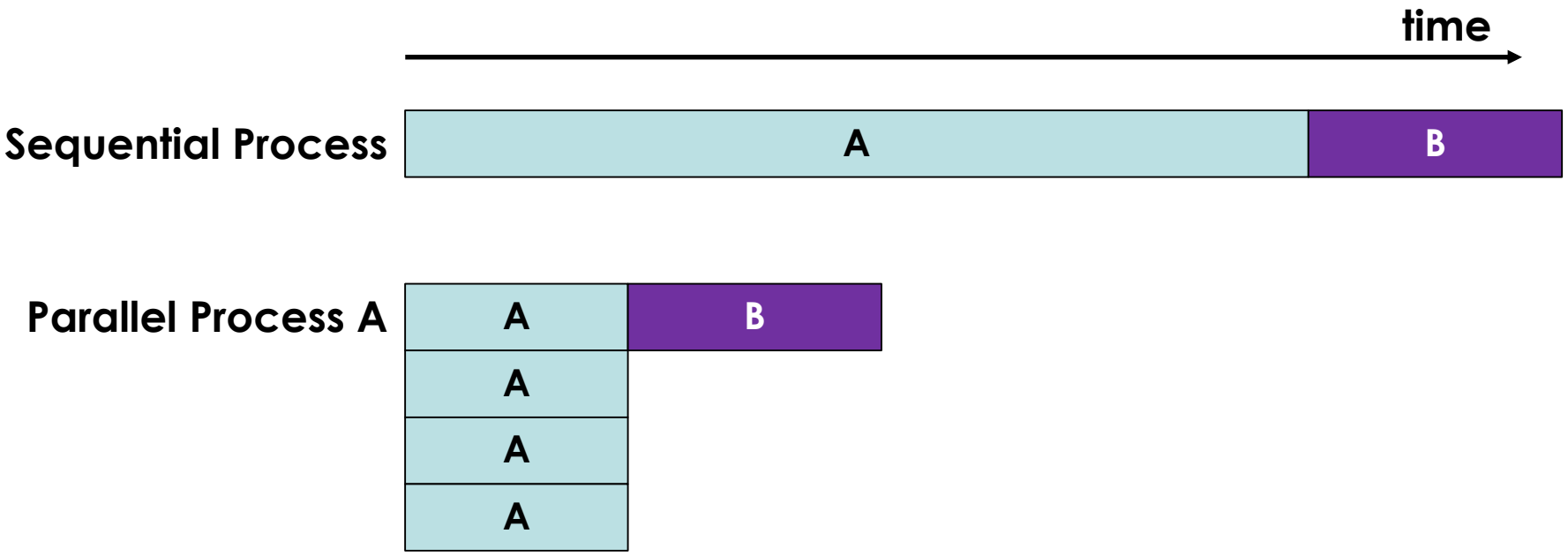
Pro Tip: Bound queues to meet response time SLAs

Can we go parallel to speedup?

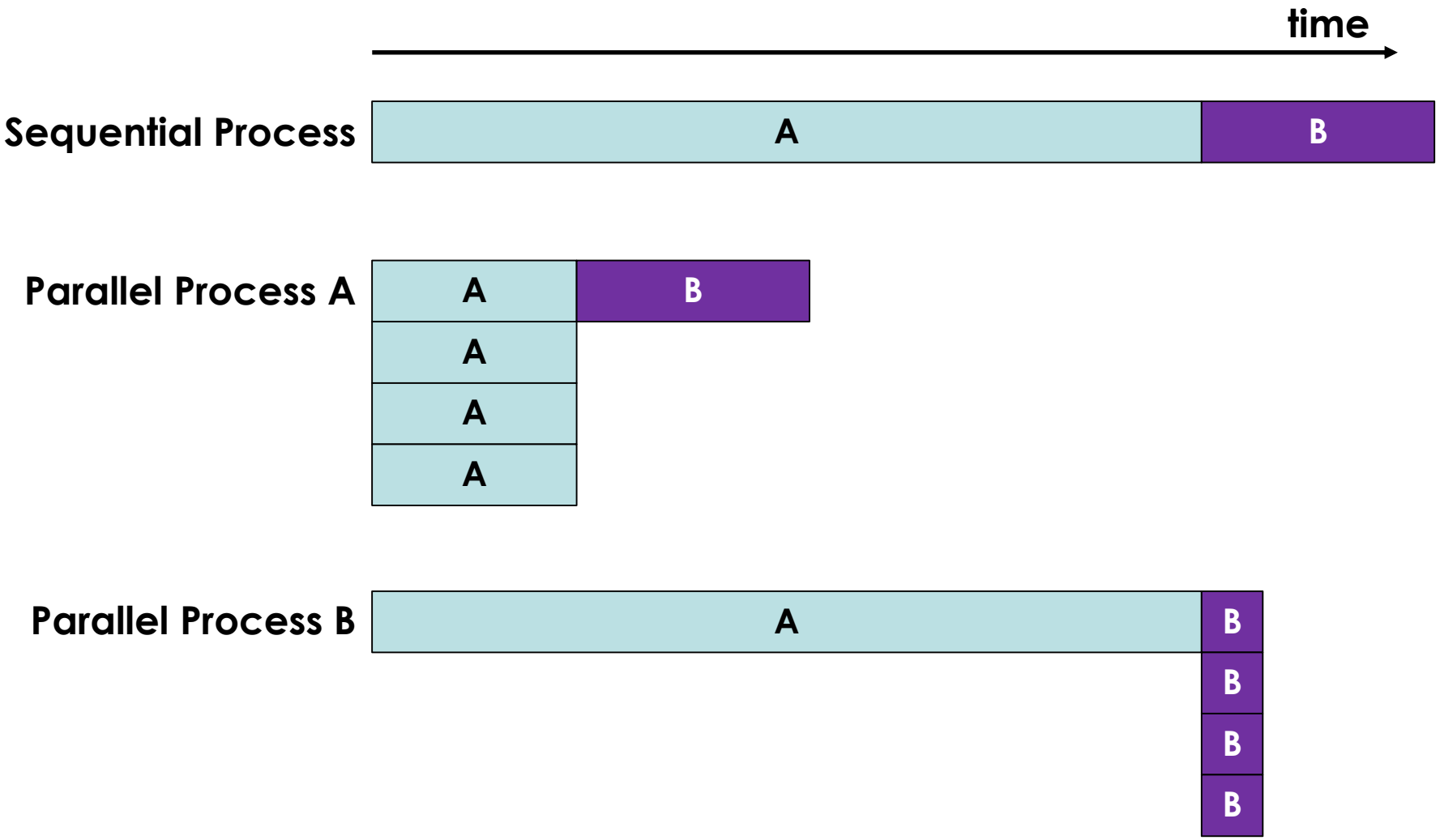
Amdahl's Law



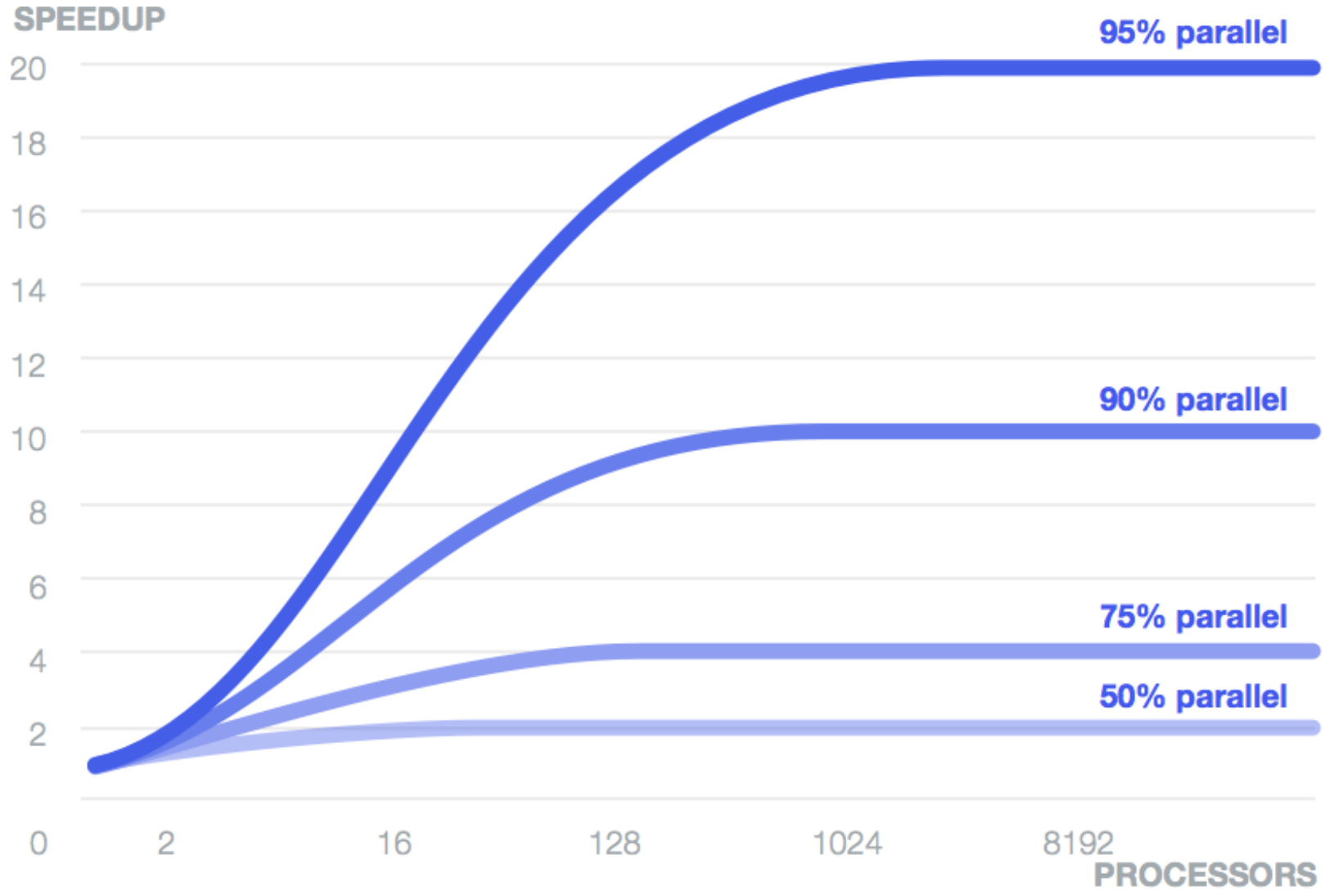
Amdahl's Law



Amdahl's Law


time
Sequential Process
A
B
Parallel Process A
A
B
A
A
A
Parallel Process B
A
B
B
B
B

Amdahl's Law


SPEEDUP
20
18
16
14
12
10
8
6
4
2
0
2
16
128
1024
8192
PROCESSORS
95% parallel
90% parallel
75% parallel
50% parallel

Universal Scalability Law

$$C(N) = N / (1 + \alpha(N - 1) + ((\beta * N) * (N - 1)))$$

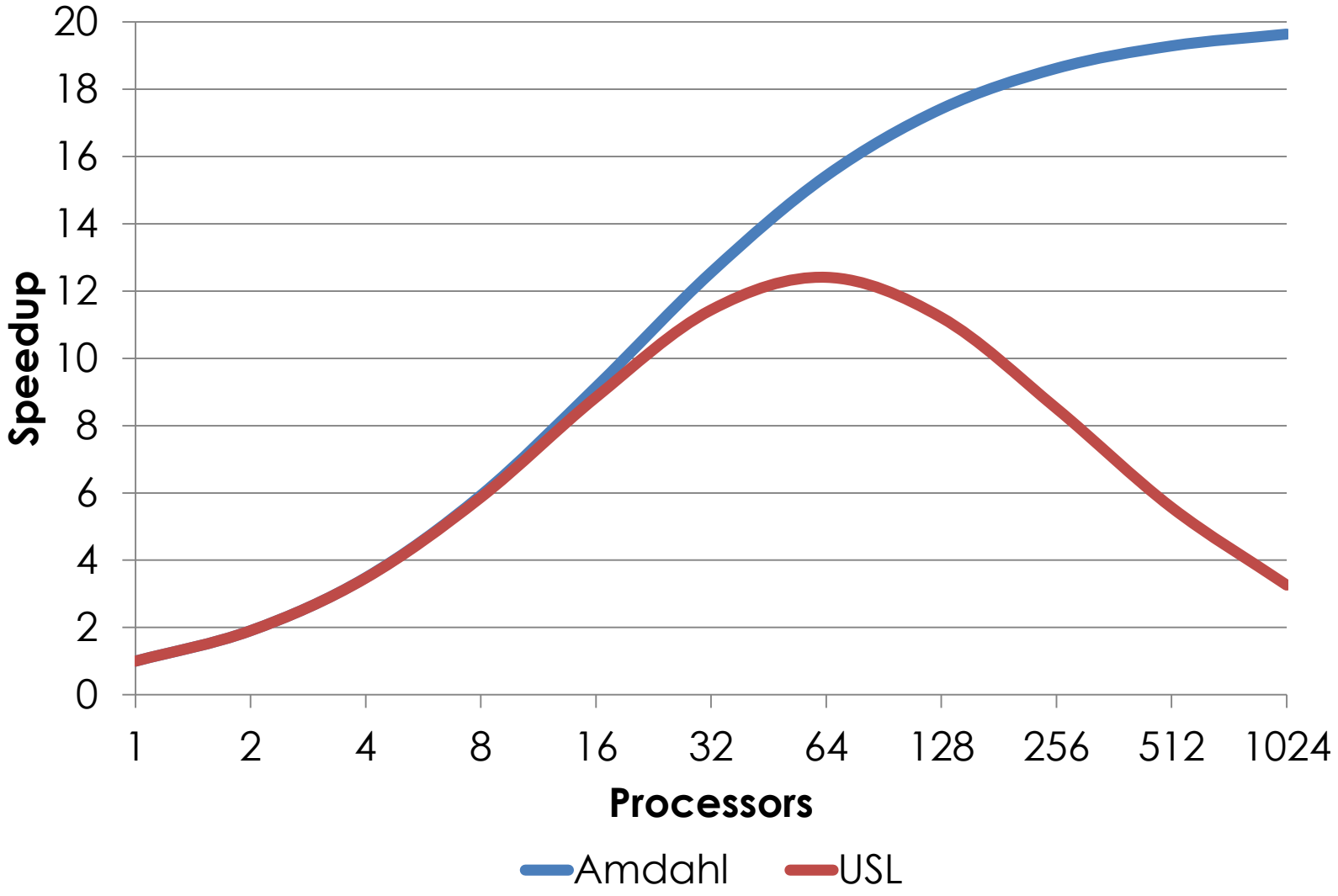
C = capacity or throughput

N = number of processors

α = ***contention*** penalty

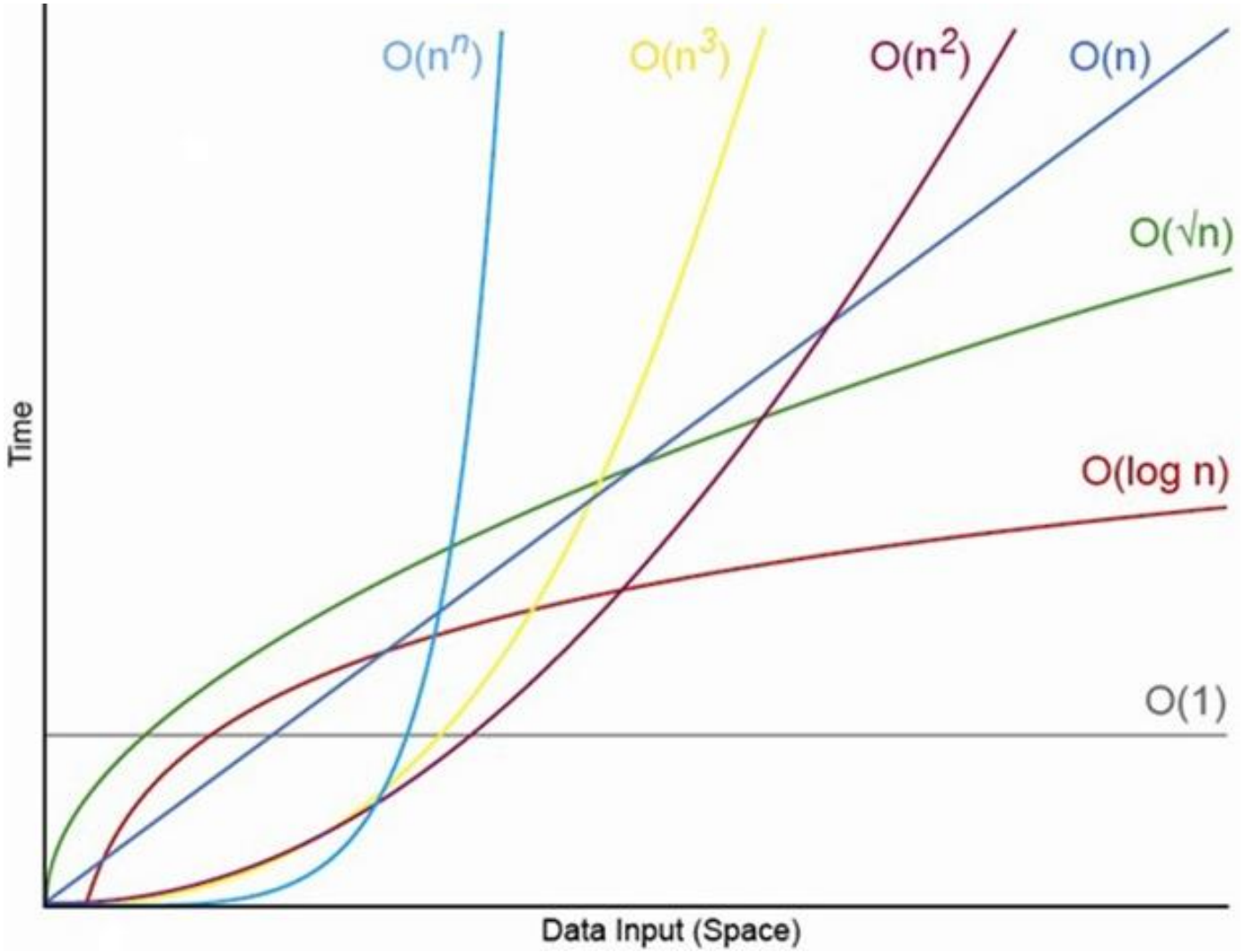
β = ***coherence*** penalty

Universal Scalability Law

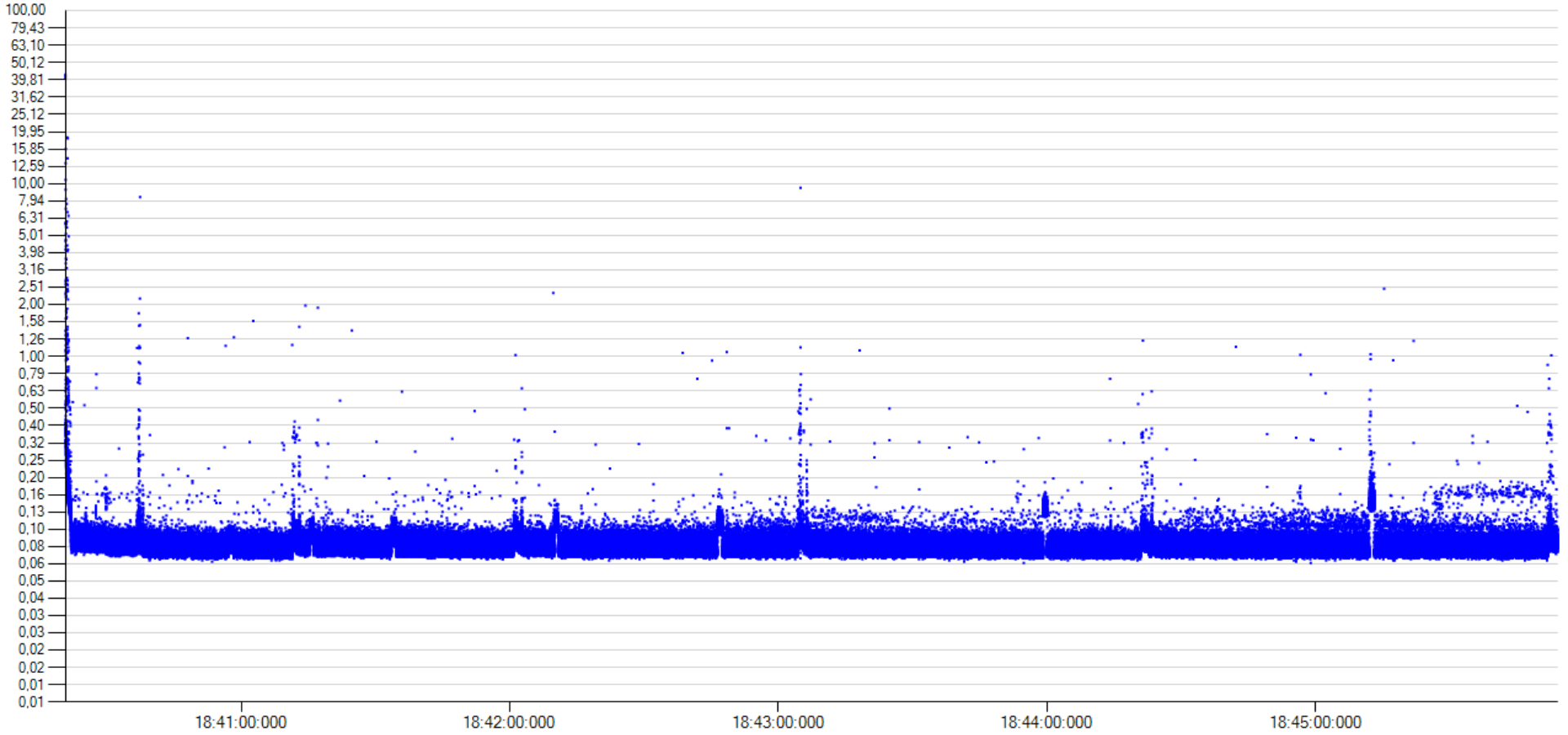


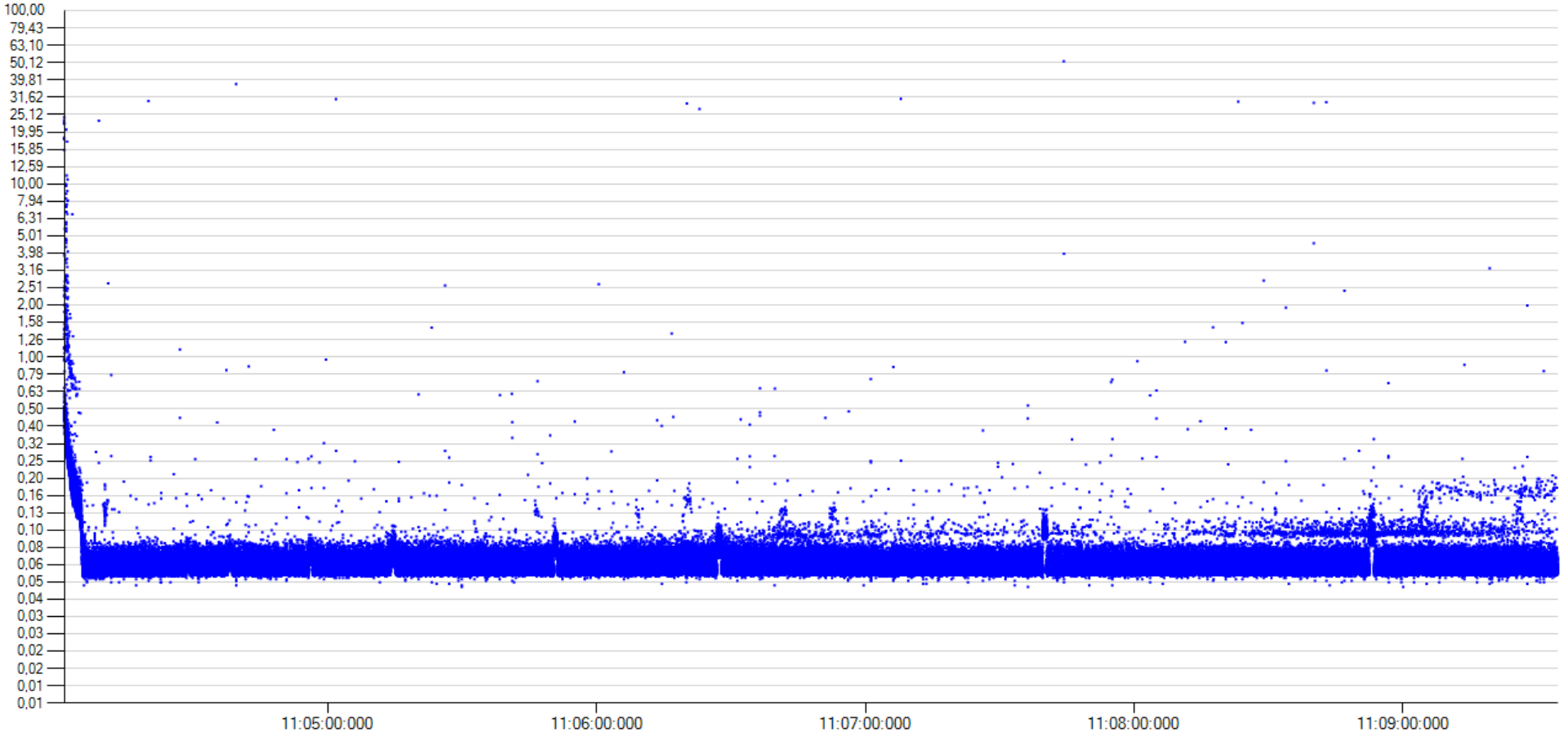
What about the service time?

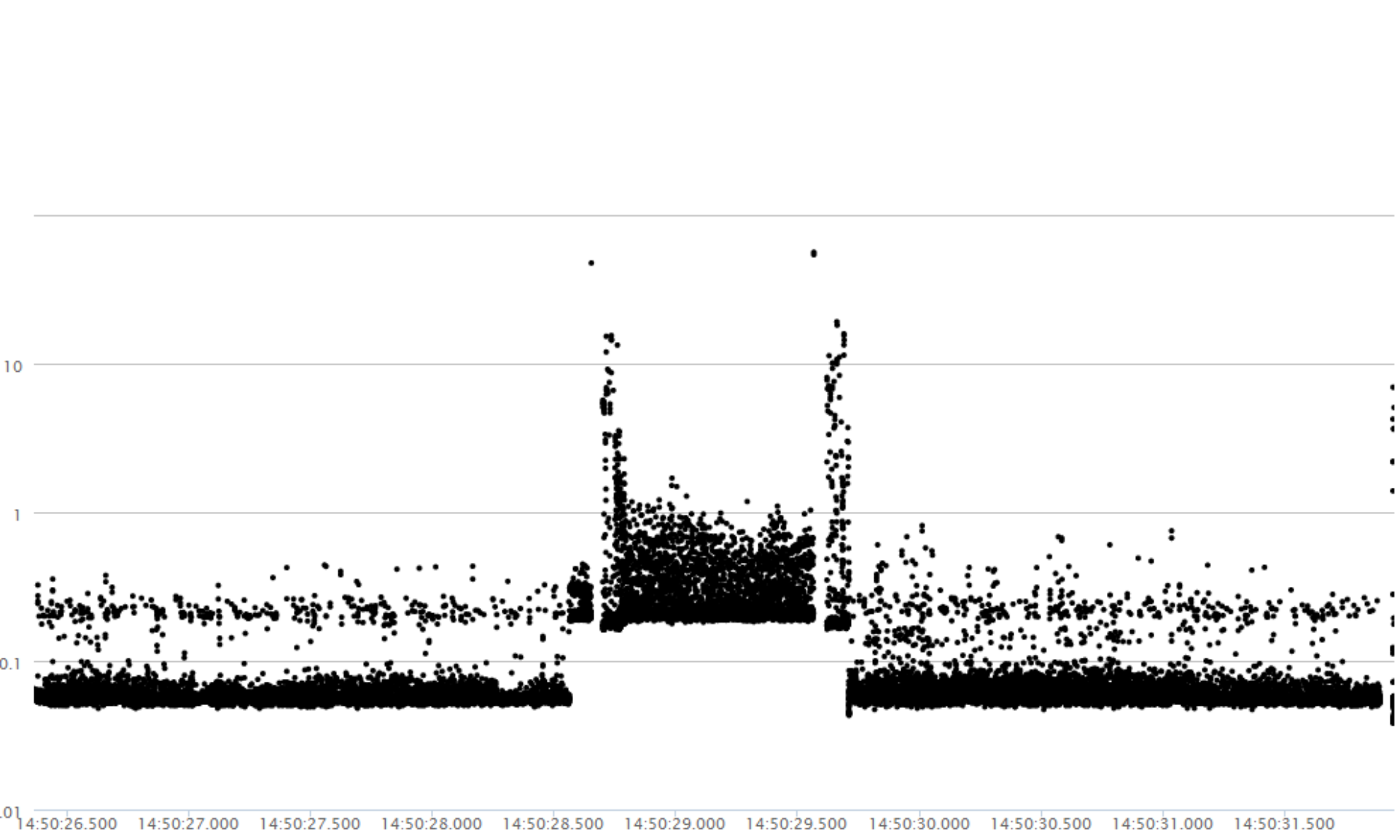
Order of Algorithms

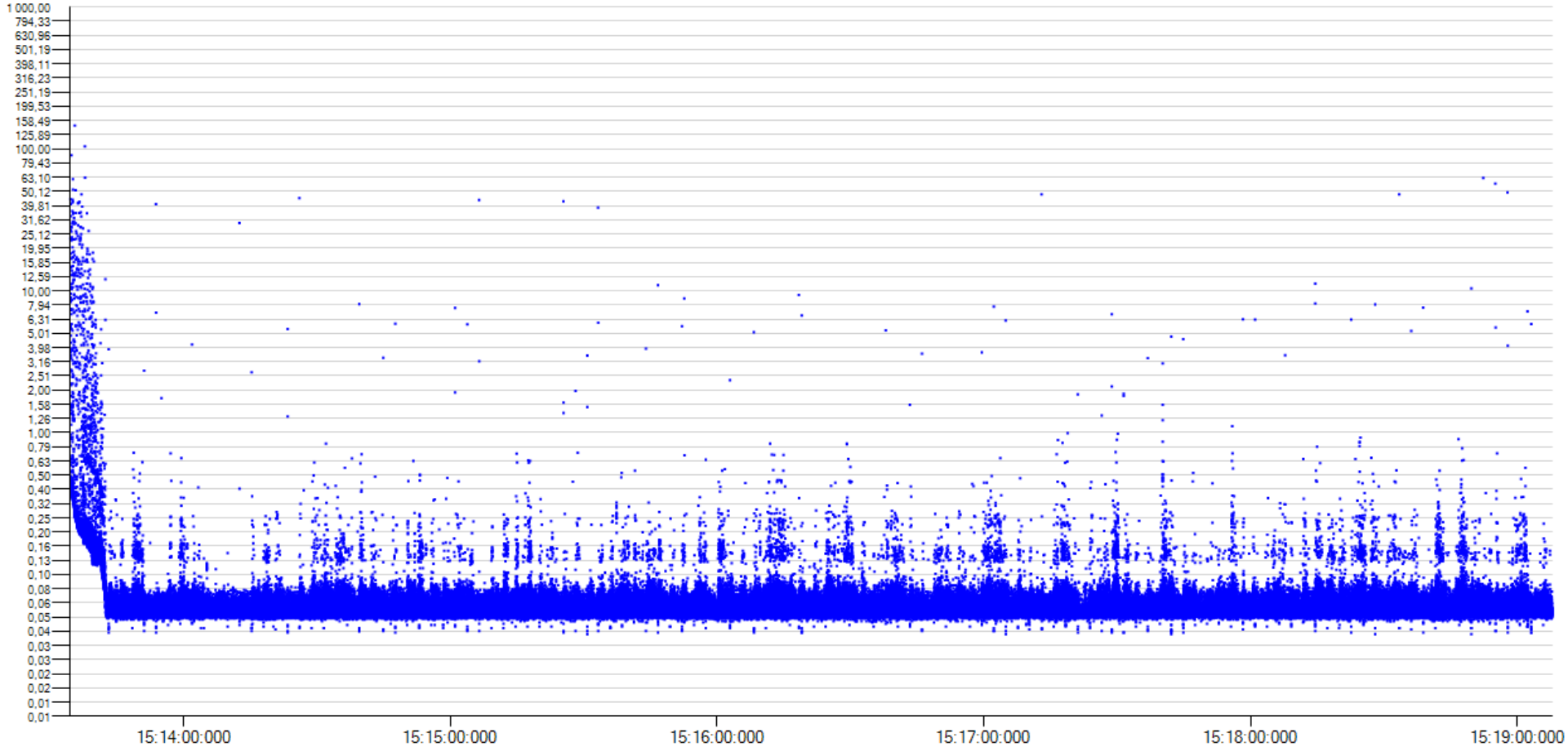


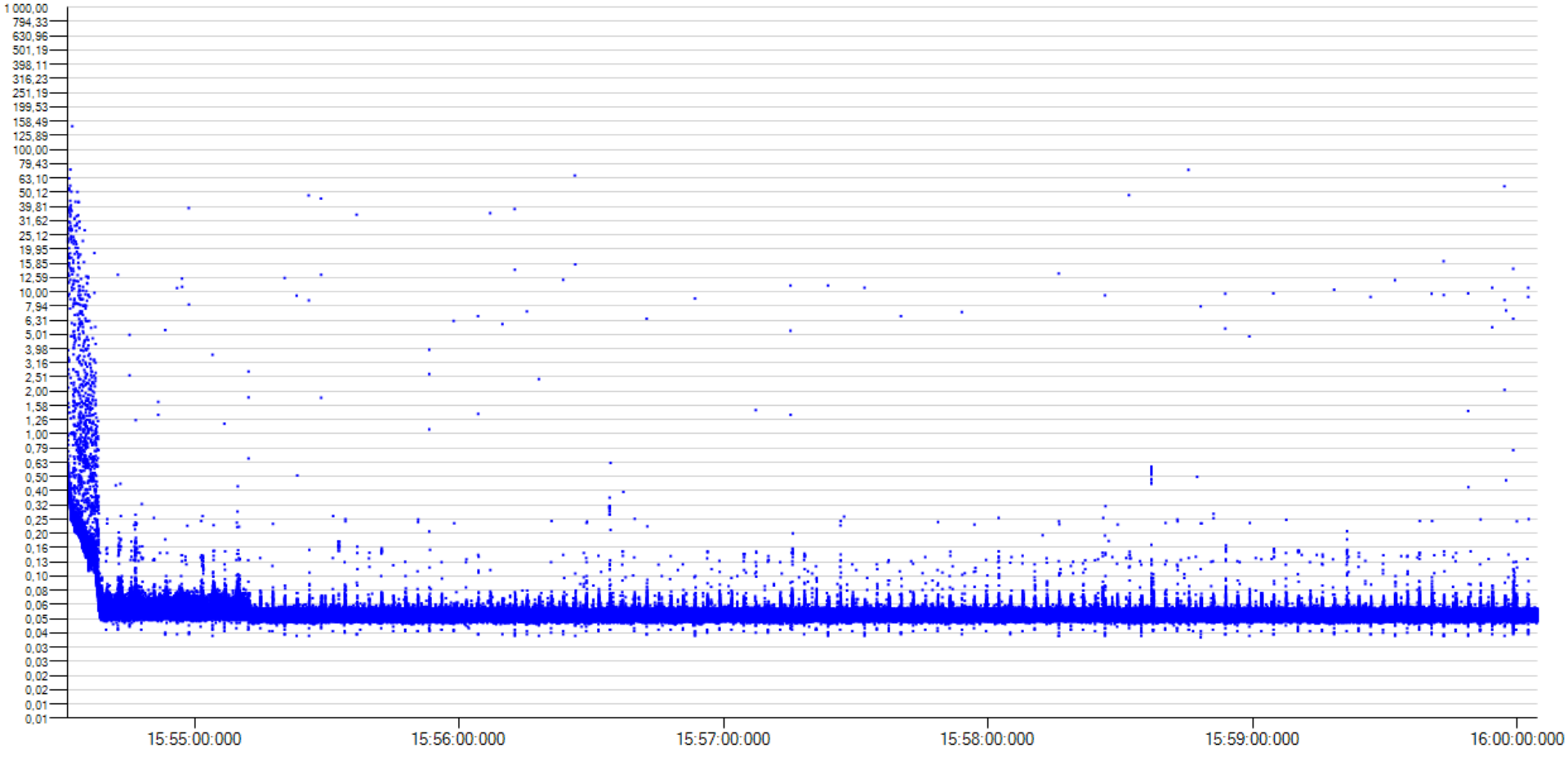
Practice


100,00
79,43
63,10
50,12
39,81
31,62
25,12
19,95
15,85
12,59
10,00
7,94
6,31
5,01
3,98
3,16
2,51
2,00
1,58
1,26
1,00
0,79
0,63
0,50
0,40
0,32
0,25
0,20
0,16
0,13
0,10
0,08
0,06
0,05
0,04
0,03
0,03
0,02
0,02
0,01
0,01
18:41:00:000
18:42:00:000
18:43:00:000
18:44:00:000
18:45:00:000


100,00
79,43
63,10
50,12
39,81
31,62
25,12
19,95
15,85
12,59
10,00
7,94
6,31
5,01
3,98
3,16
2,51
2,00
1,58
1,26
1,00
0,79
0,63
0,50
0,40
0,32
0,25
0,20
0,16
0,13
0,10
0,08
0,06
0,05
0,04
0,03
0,03
0,02
0,02
0,01
0,01
11:05:00:000
11:06:00:000
11:07:00:000
11:08:00:000
11:09:00:000


10
1
0.1
14:50:26.500
14:50:27.000
14:50:27.500
14:50:28.000
14:50:28.500
14:50:29.000
14:50:29.500
14:50:30.000
14:50:30.500
14:50:31.000
14:50:31.500


1 000,00
794,33
630,96
501,19
398,11
316,23
251,19
199,53
158,49
125,89
100,00
79,43
63,10
50,12
39,81
31,62
25,12
19,95
15,85
12,59
10,00
7,94
6,31
5,01
3,98
3,16
2,51
2,00
1,58
1,26
1,00
0,79
0,63
0,50
0,40
0,32
0,25
0,20
0,16
0,13
0,10
0,08
0,06
0,05
0,04
0,03
0,03
0,02
0,02
0,01
0,01
15:14:00:000
15:15:00:000
15:16:00:000
15:17:00:000
15:18:00:000
15:19:00:000


1 000,00
794,33
630,96
501,19
398,11
316,23
251,19
199,53
158,49
125,89
100,00
79,43
63,10
50,12
39,81
31,62
25,12
19,95
15,85
12,59
10,00
7,94
6,31
5,01
3,98
3,16
2,51
2,00
1,58
1,26
1,00
0,79
0,63
0,50
0,40
0,32
0,25
0,20
0,16
0,13
0,10
0,08
0,06
0,05
0,04
0,03
0,03
0,02
0,02
0,01
0,01
15:55:00:000
15:56:00:000
15:57:00:000
15:58:00:000
15:59:00:000
16:00:00:000

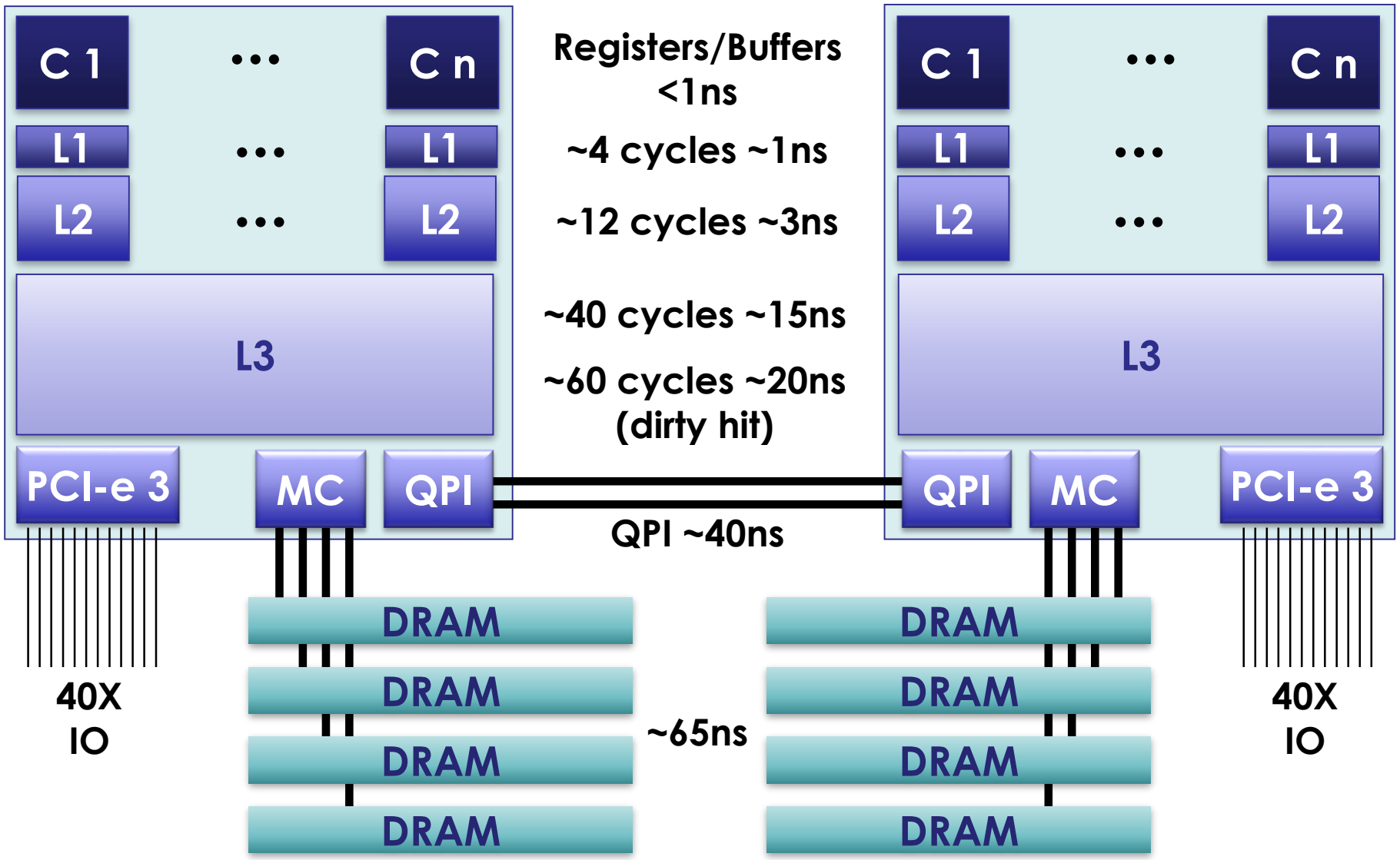
Pitfalls

Modern Processors

SMIs?

Hyperthreading?

Non-Uniform Memory Architecture (NUMA)

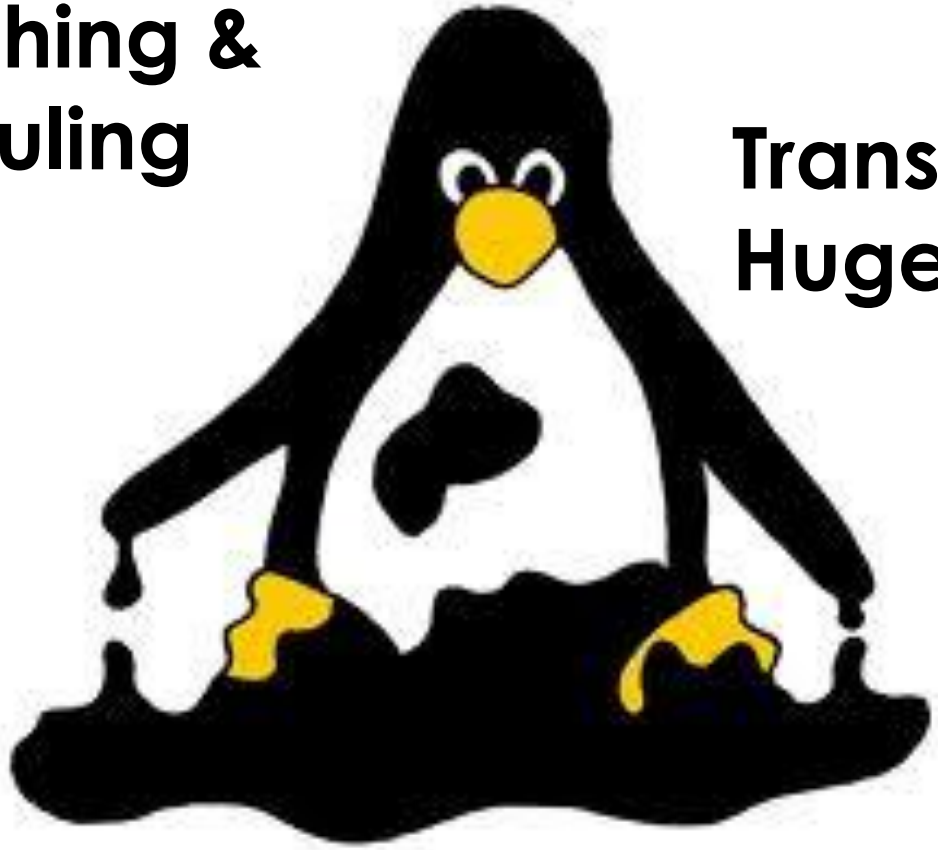


* Assumption: 3GHz Processor

Virtual Memory Management

Page Flushing &
IO Scheduling

Transparent
Huge Pages



Swap???

vm.min_free_kbytes

Safepoints in the JVM

Garbage Collection, De-optimisation, Biased Locking, Stack traces, etc.

Virtualization

System Calls

Notification

```
public class SomethingUseful
{
    // Lots of useful stuff

    public void handOffSomeWork()
    {
        // prepare for handoff

        synchronized (this)
        {
            someObject.notify();
        }
    }
}
```

Notification

```
public class SomethingUseful
{
    // Lots of useful stuff

    public void handOffSomeWork()
    {
        // prepare for handoff

        synchronized (thi
        {
            someObject.
        }
    }
}
```

Law of Leaky Abstractions

"All non-trivial abstractions, to some extent, are leaky."

- Joel Spolsky

Law of Leaky Abstractions

"The detail of underlying complexity cannot be ignored."

Mechanical Sympathy

Responding in the presence of failure

Algorithms & Techniques

Clean Room Experiments

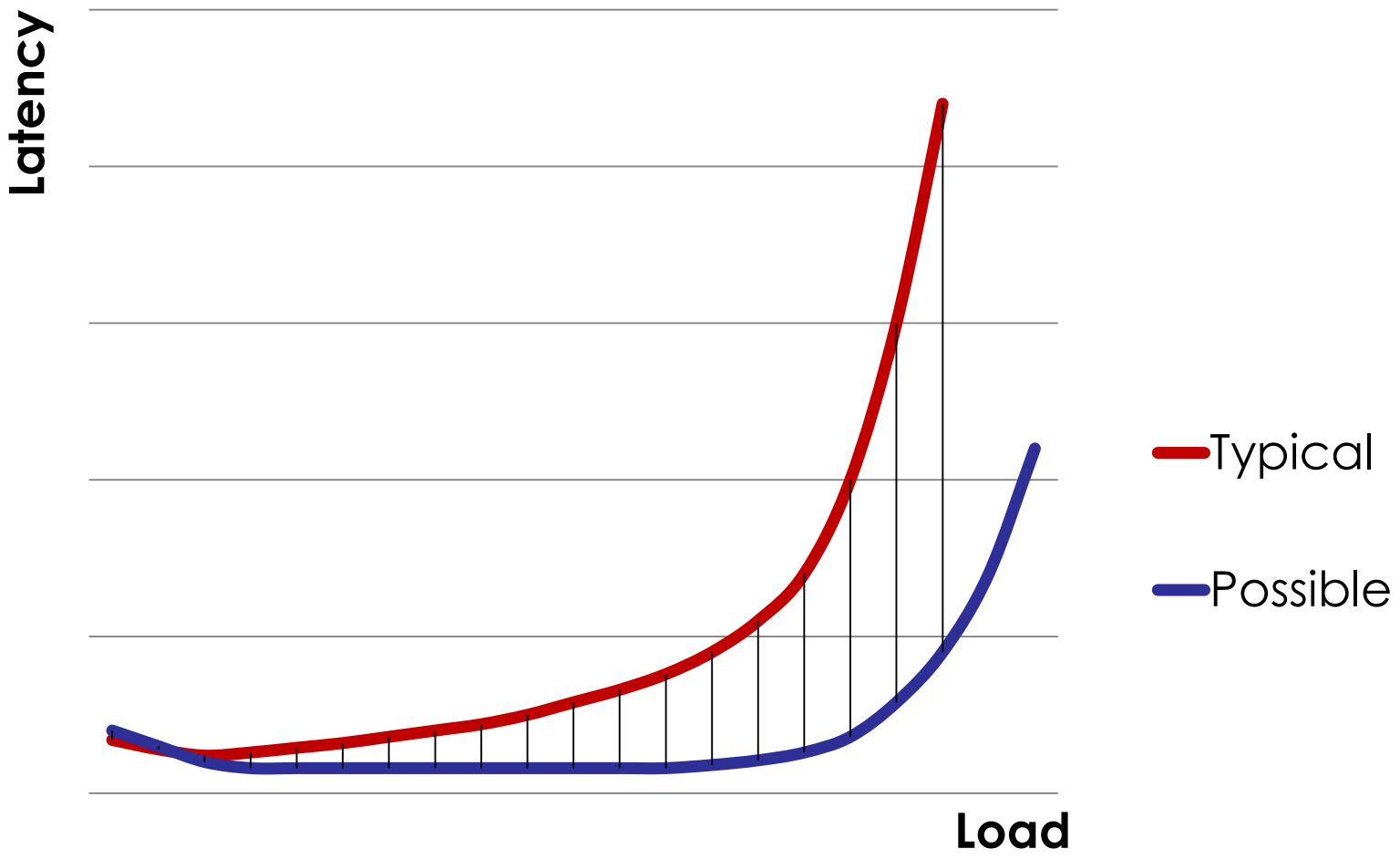
- sufficient CPUs
- intel_idle.max_cstate=0
- cpufreq
- isocpus
- numctl, cgroups, affinity
- “Washed” SSDs
- network buffer sizing
- jHiccup

- tune your stack!
- Mechanical Sympathy

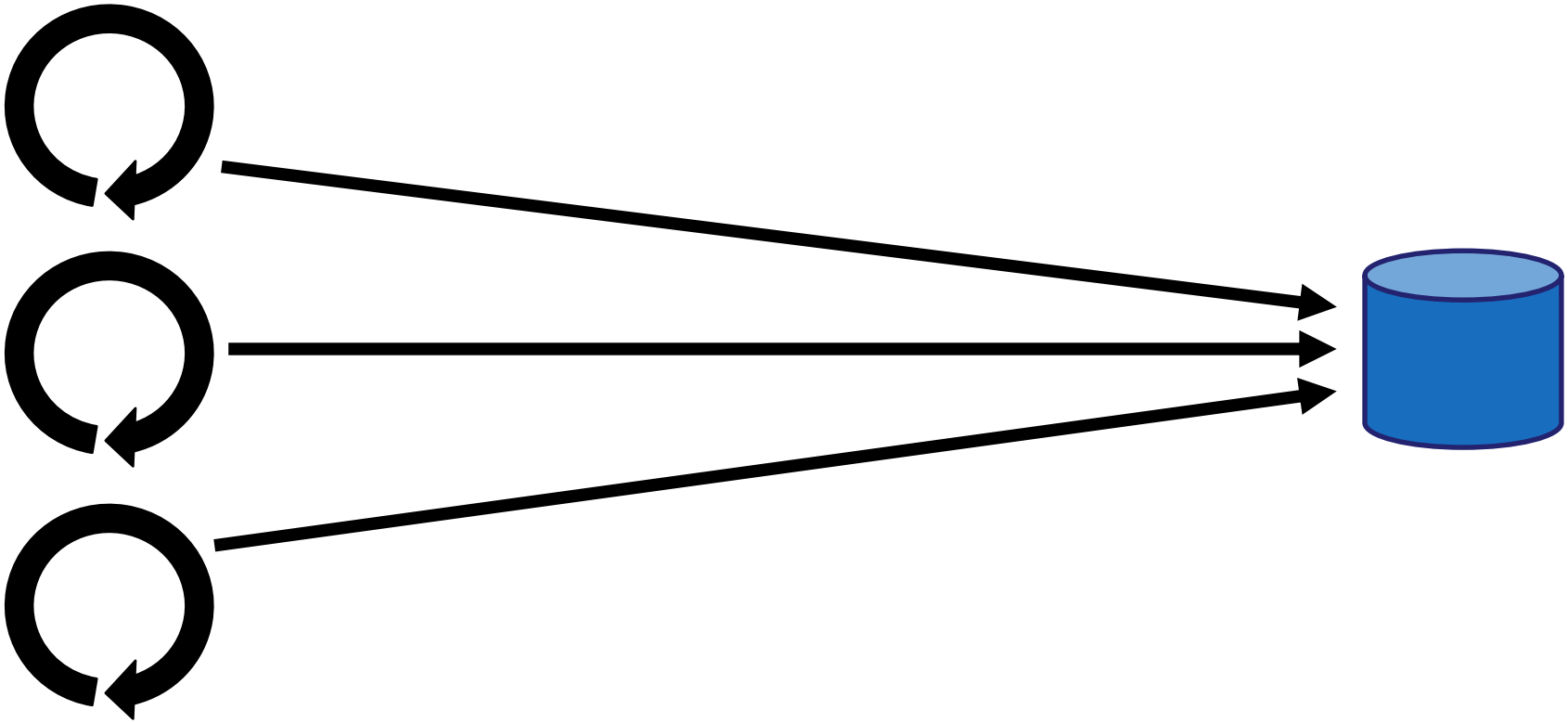
Profiling

Pro Tip: Incorporate telemetry and histograms

Smart Batching

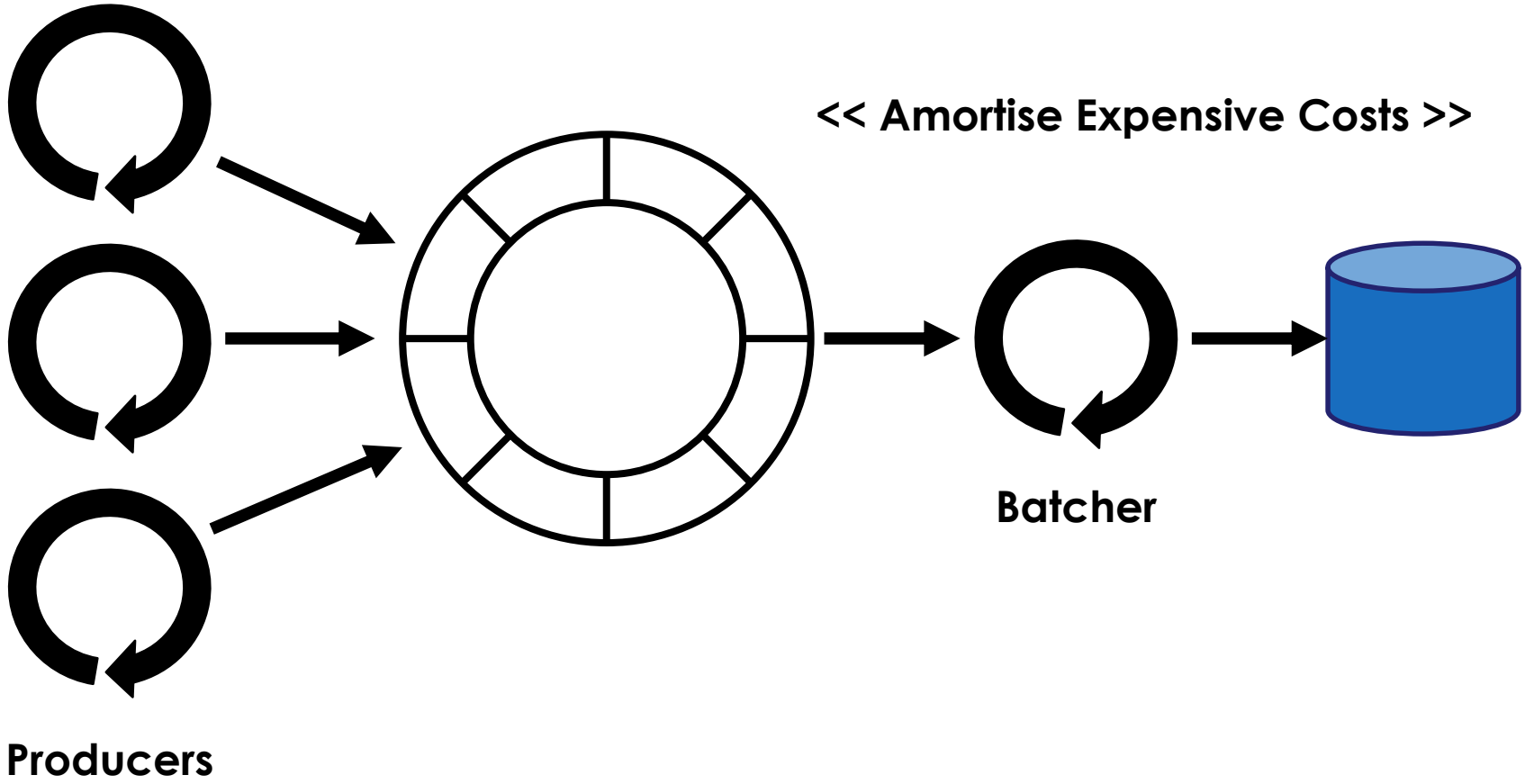


Smart Batching



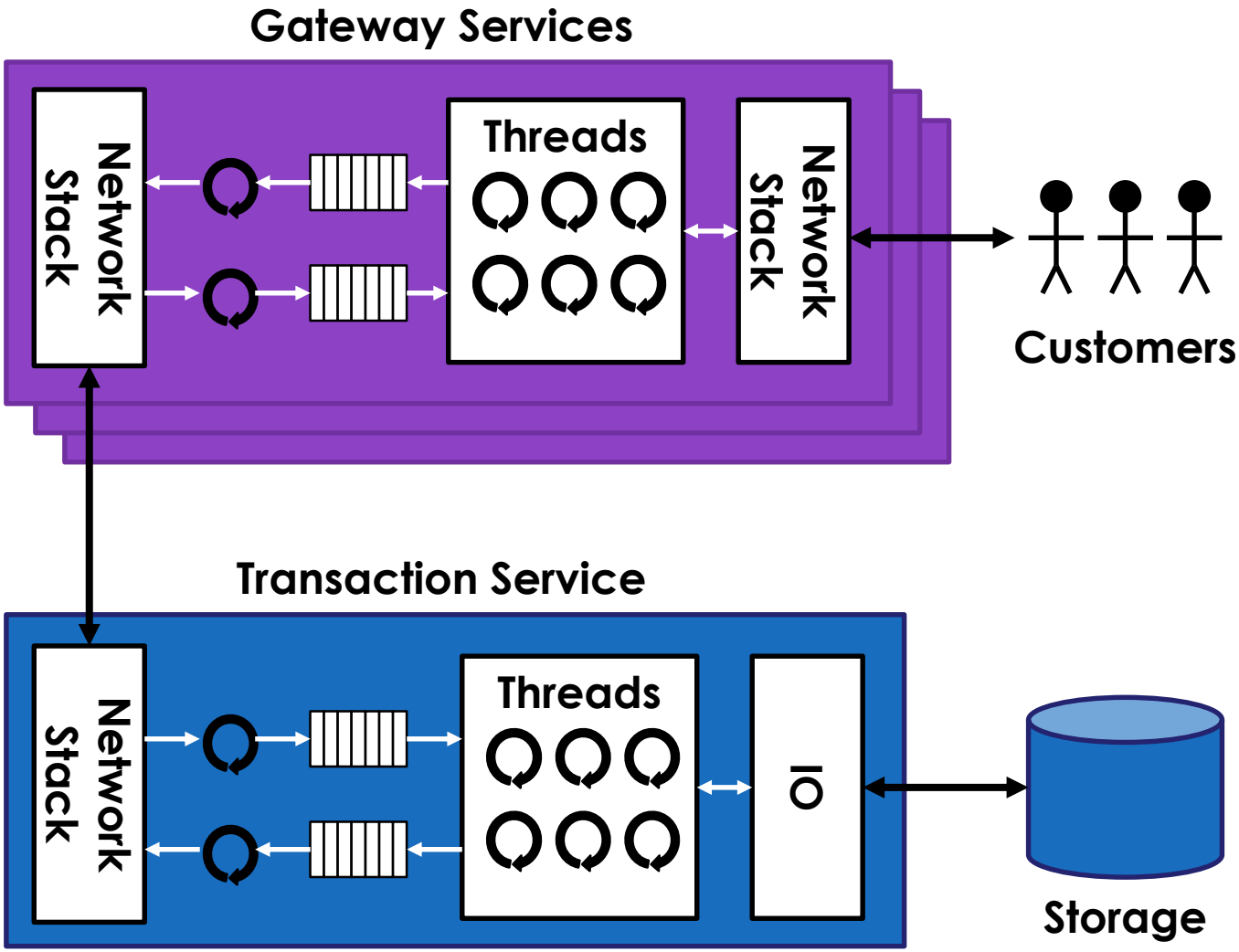
Producers

Smart Batching



Pro Tip: Amortise the Expensive Costs

Applying Backpressure



Non-Blocking Design

"Get out of your own way!"

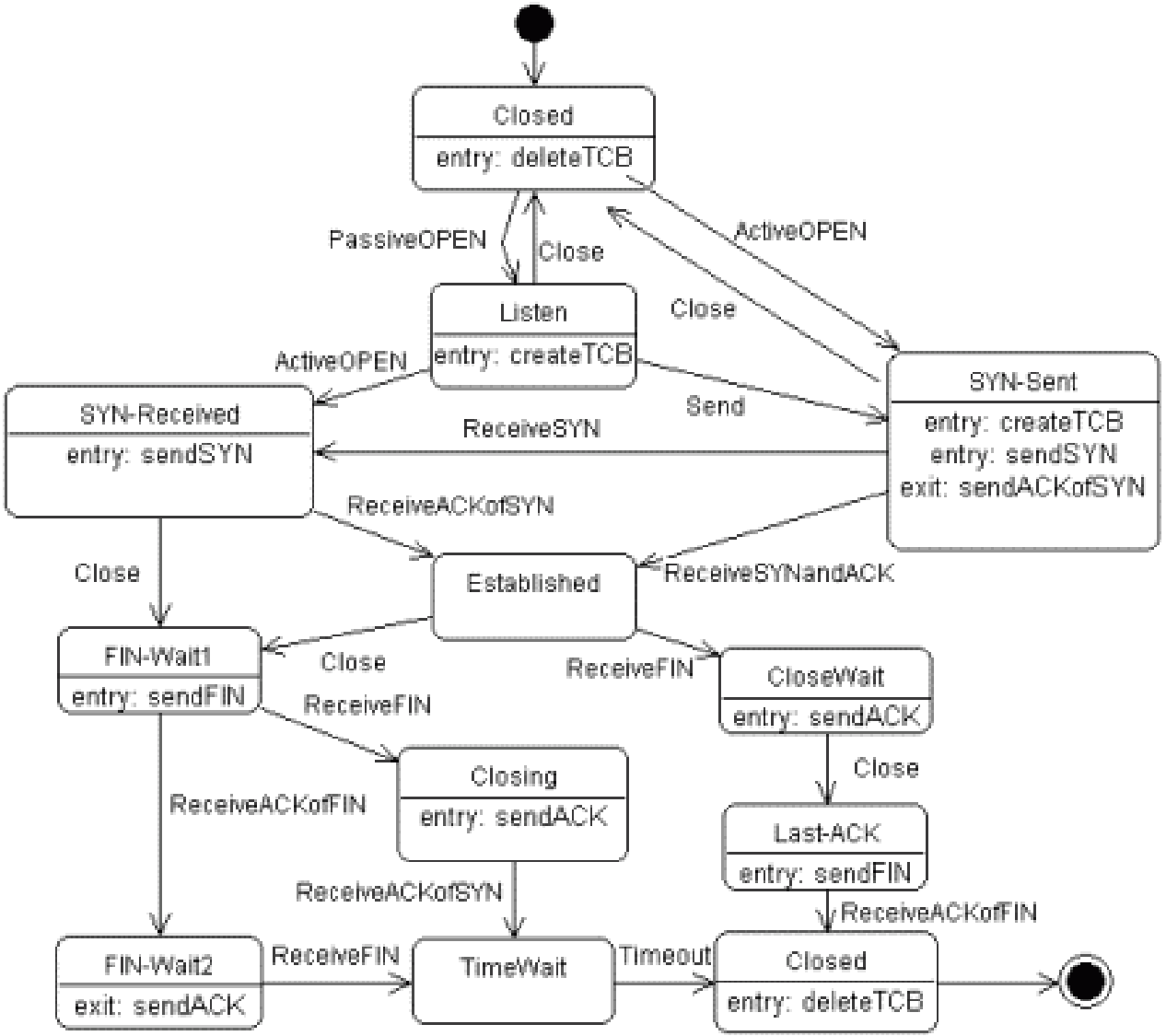
- Don't hog any resource
- Always try to make progress
- Enables Smart Batching

Pro Tip: Beware of hogging resources in synchronous designs

Lock-Free Concurrent Algorithms

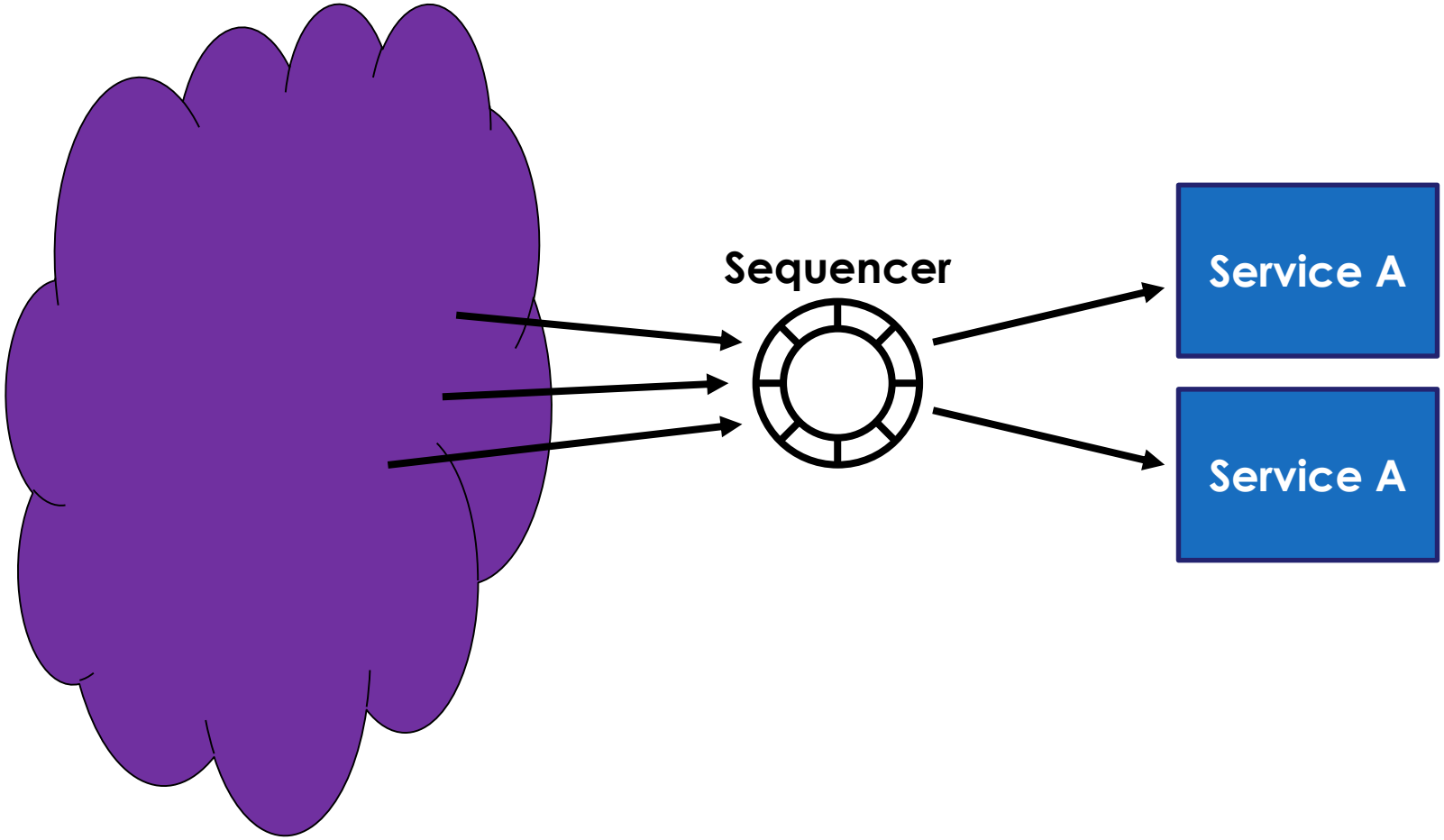
- Agree protocols of interaction
- Don't get a 3rd party involved, i.e. the OS
- Keep to user-space
- Beat the "notify()" problem

Observable State Machines

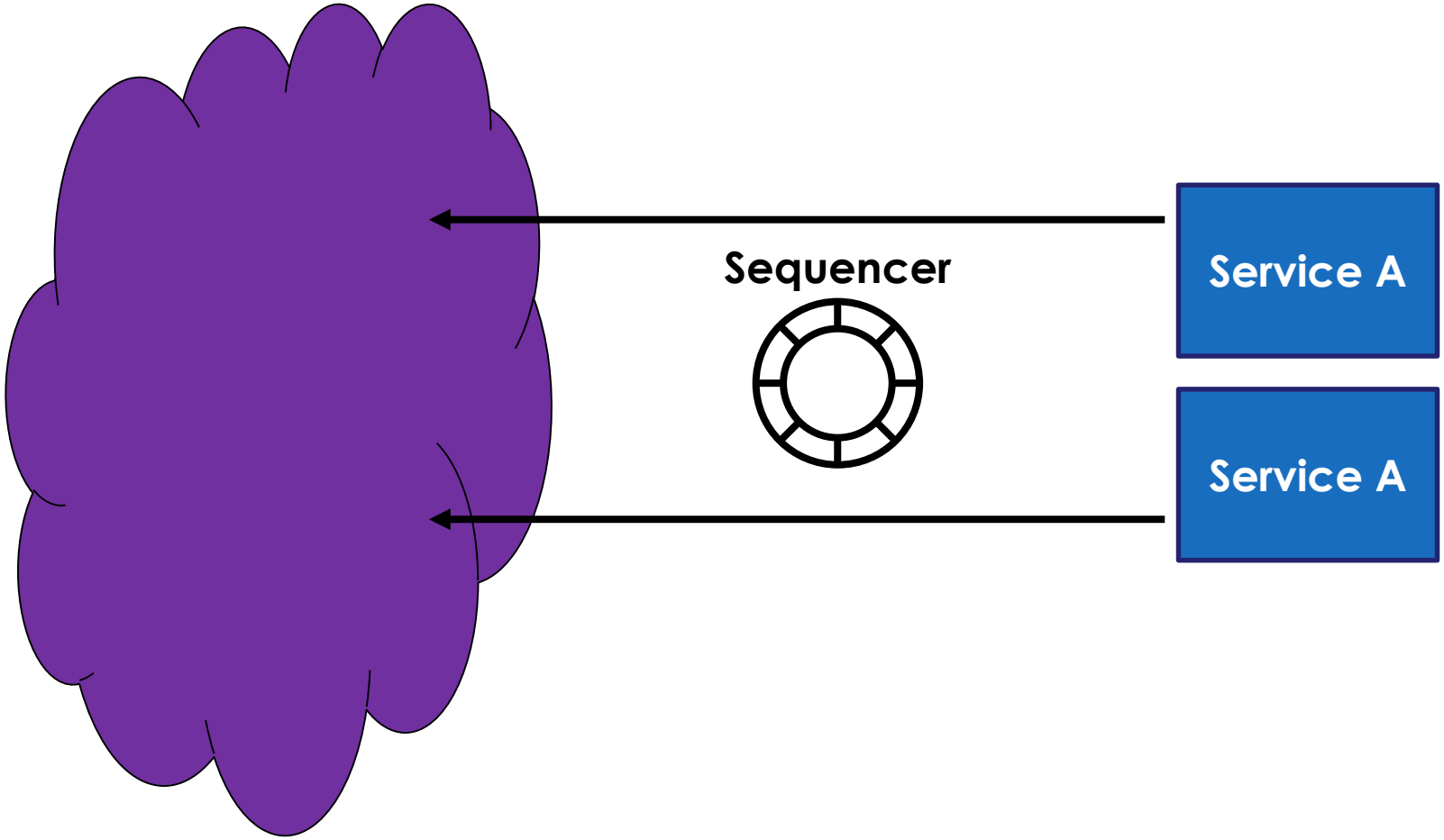


Pro Tip: Observable state machines make monitoring easy

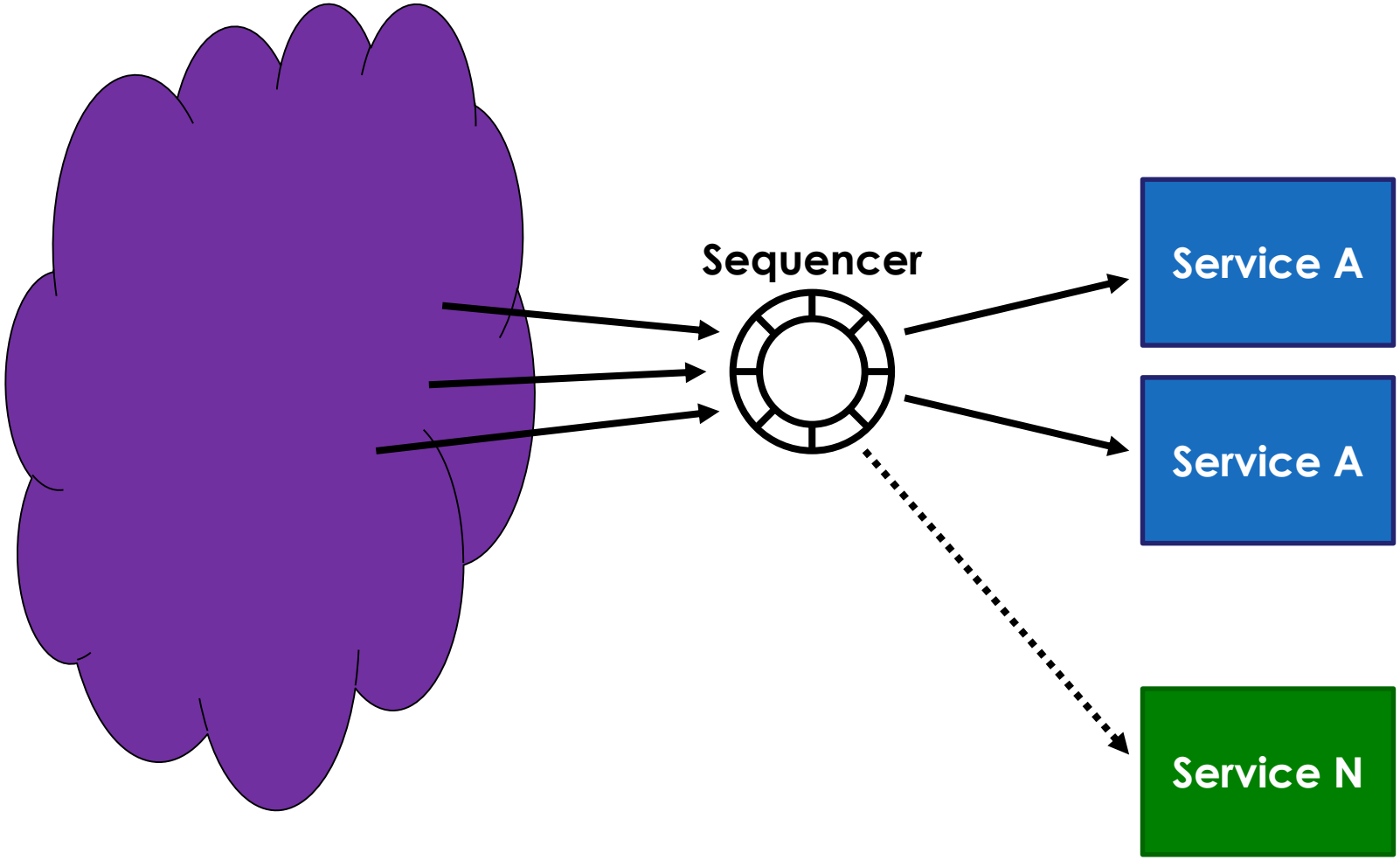
Cluster for Response and Resilience



Cluster for Response and Resilience



Cluster for Response and Resilience



Data Structures and O(?) Models

Is there a world beyond maps and lists?

In closing…


INCOMING!

WINTER IS COMING

The Internet of Things (IoT)

"There will be X connected devices by 2020..."

Where *X* is 20 to 75 Billion

If you cannot control arrival rates...

...you have to think hard about improving service times!

...and/or you have to think hard about removing all contention!

Questions?

Blog: http://mechanical-sympathy.blogspot.com/

Twitter: @mjpt777

"It does not matter how intelligent you are, if you guess and that guess cannot be backed up by experimental evidence – then it is still a guess."

- Richard Feynman