Lecture 1:
What do we measure?

PAMS'18

Zsolt István

zsolt.istvan@imdea.org

Slide contents heavily influenced by G. Alonso's Advanced Systems Lab lecture slides.

Why do we measure performance?

- Our goal is to understand the behavior of the system, predict its behavior
  - “We want to use Apache Kafka in our project. Can we deliver 10k operations/s to our clients if we use it?”
  - “Our application has 1 million users, uploading 1 photo/day. How many more users can the database handle before we need to upgrade our infrastructure?”
  - “The encryption module I am developing is slowing down the rest of the system, how do I figure out what part of the code needs redesign?”
  - …
- Modelling can help answer what-if questions.

All Models Are Wrong, Some Are Useful (G. Box)

- We should have a hypothesis in mind that we want to prove/disprove
- A model which is good in predicting one aspect, might not be useful for other aspects
- Avoid “overfitting” – should not have to redesign whole model when moving from from Xeon CPU to other…

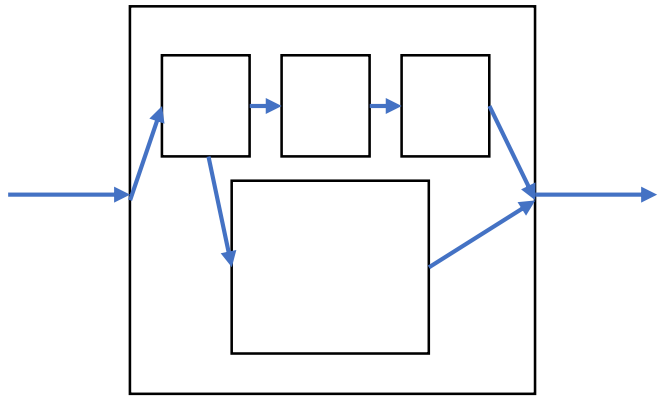
Ways of looking at the system

"Black box" modelling

- No knowledge/consideration of components
- Issue requests, measure how long answers take

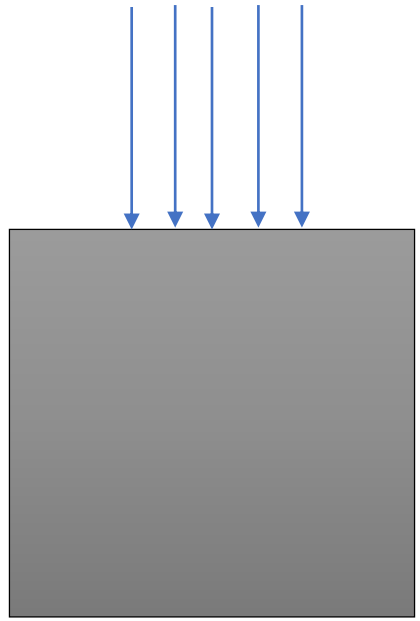
"White box" modelling

- Takes into account internal components
- Can become arbitrarily intricate
- Most complex model not always the best!



Open system

- Request can arrive “at any time”
  - Potentially infinite clients
- The rate at which requests arrive are not influenced by the server
- E.g. Web Server, (your email inbox)

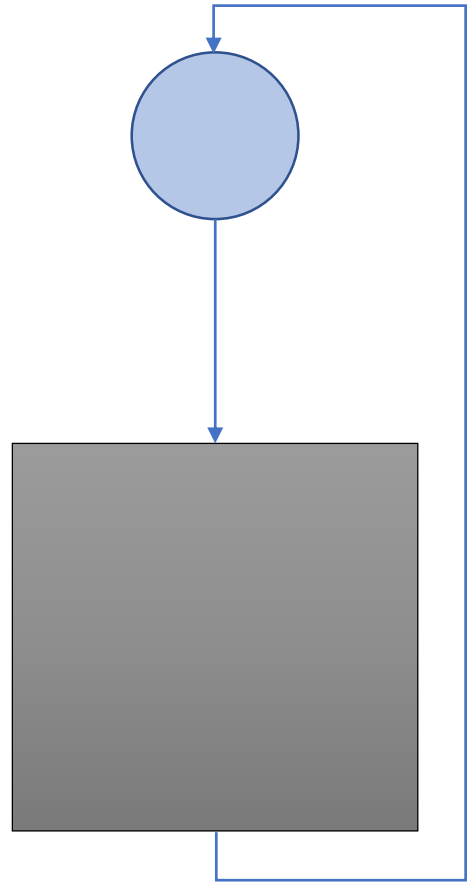


- Benchmarking:
  - Test the system with specific throughput levels
  - Test the system when pushed beyond its capacity

Closed system

- Limited number of clients
- Each client waits for a response before sending next request
- The load is self-adjusting
- E.g. database with local clients

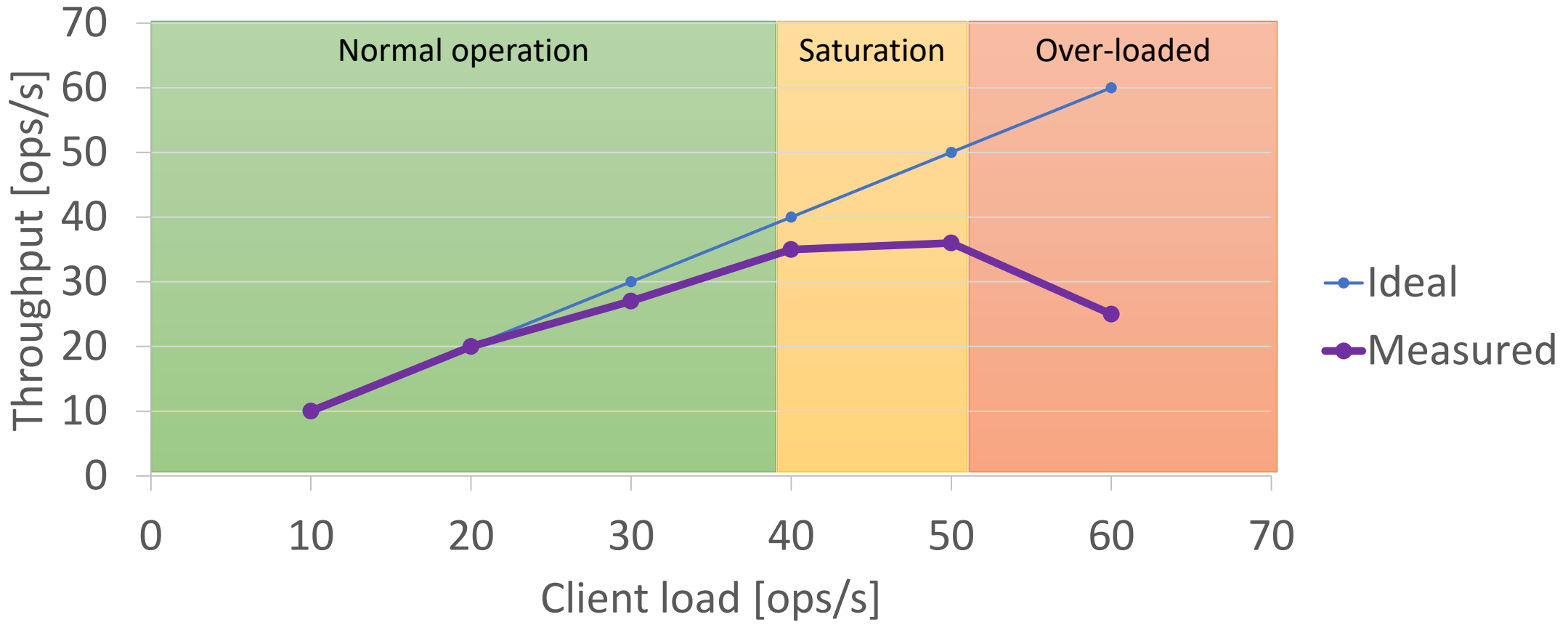
- Benchmarking:
  - Behavior with increasing number of clients
  - Verifying that the behavior is stable



Throughput

- Requests completed successfully per unit of time
  - e.g. Pizzas delivered per week, KVS accesses per second, etc.
  - Don’t count failed requests!
- Can be measured by clients or server – ideally the same
- We can talk of throughput in conjunction with a user workload
  - If we only send one request per hour, doesn’t mean the server couldn’t handle more!
  - (We’ll see some examples at the end)

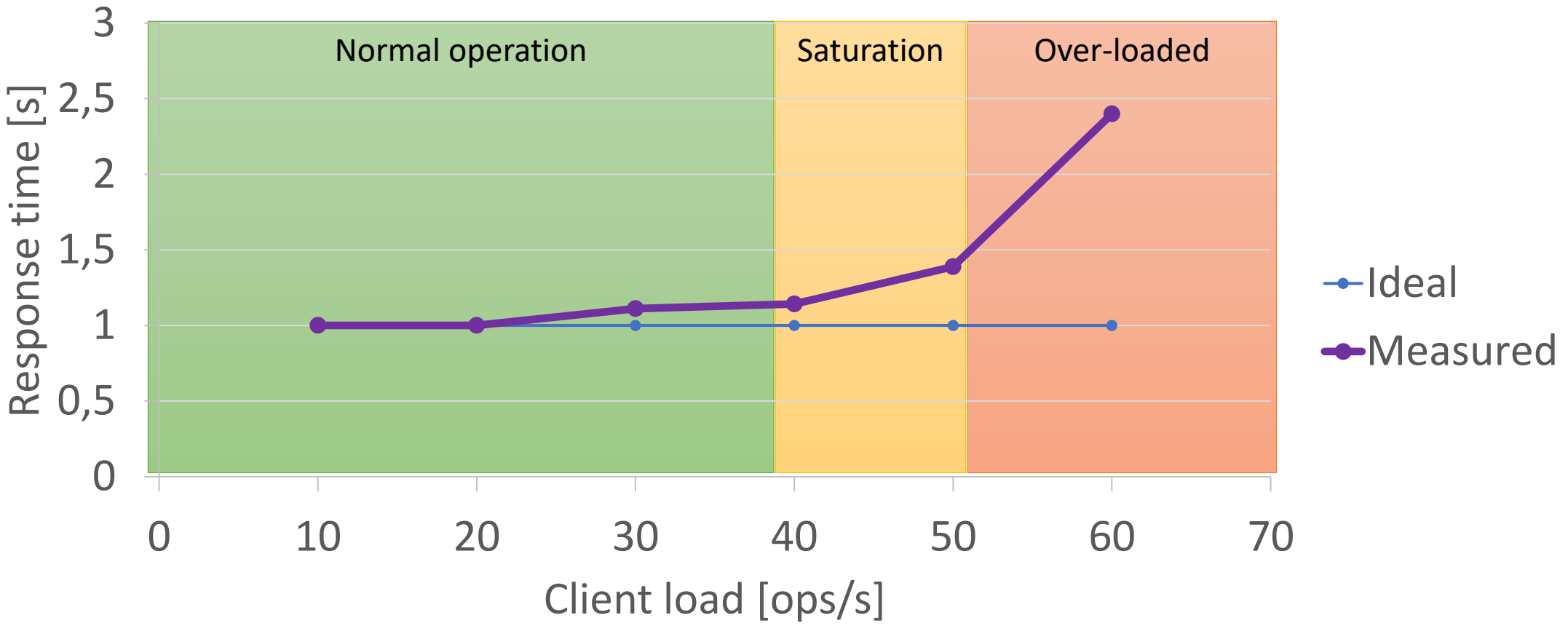
Throughput in practice



Response time

- Time it takes to handle a request and send back a response
  - Must define what we measure!
  - Only consider successful requests
- Average response time is common metric
  - But minimum/maximum, uniformity can be just as important!
  - Guarantee some behavior to users (SLAs)
- In a closed system: Throughput connected to average response time
  - Minimum recorder response time to determine upper bound for throughput*

Response time in practice



Interactive response time law

Can be applied to closed systems:

- Each user thinks for some time (Z), submits a request, waits for a response. Repeats.

- Throughput: X=Jobs/Time

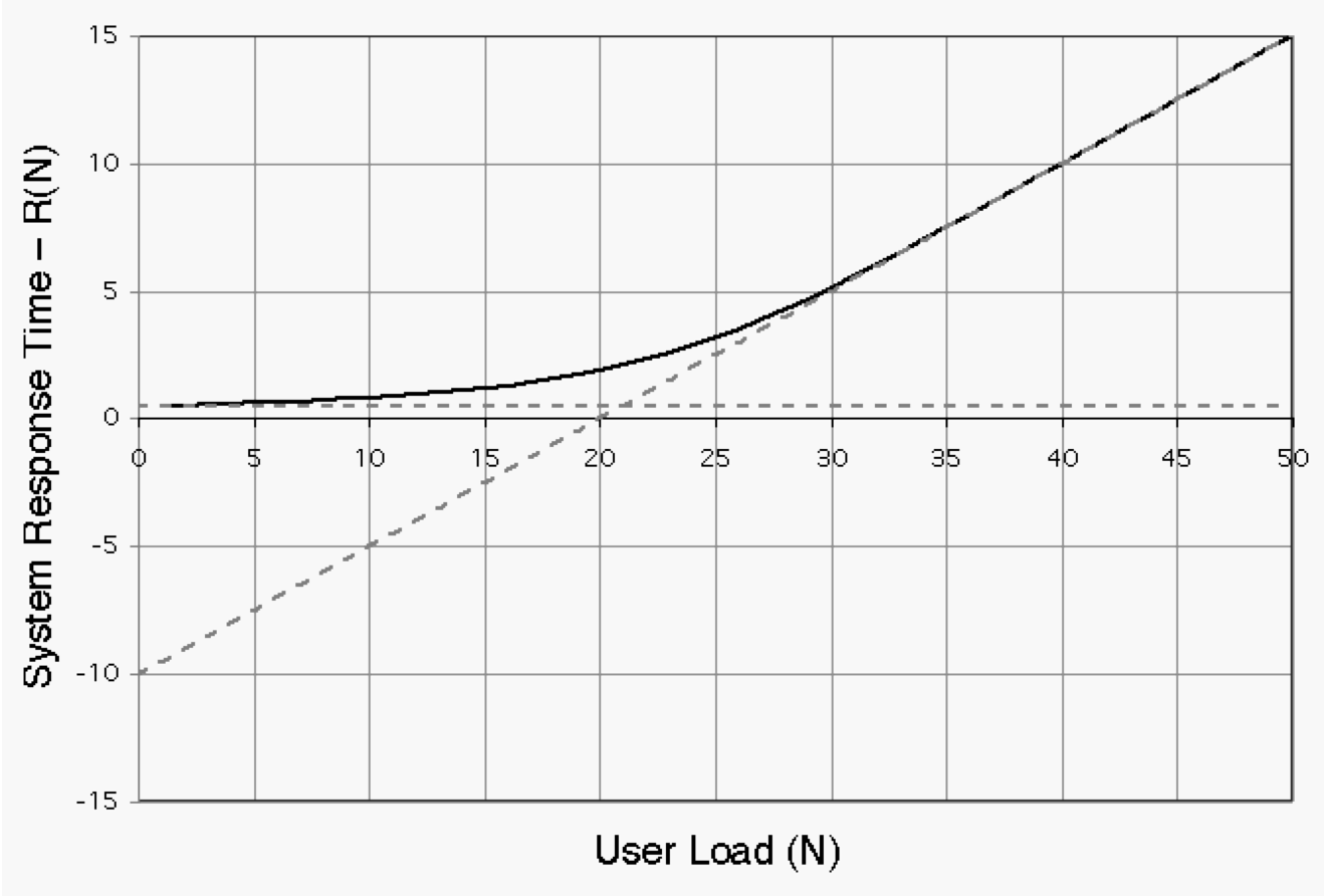
- How many jobs?

Interactive response time law

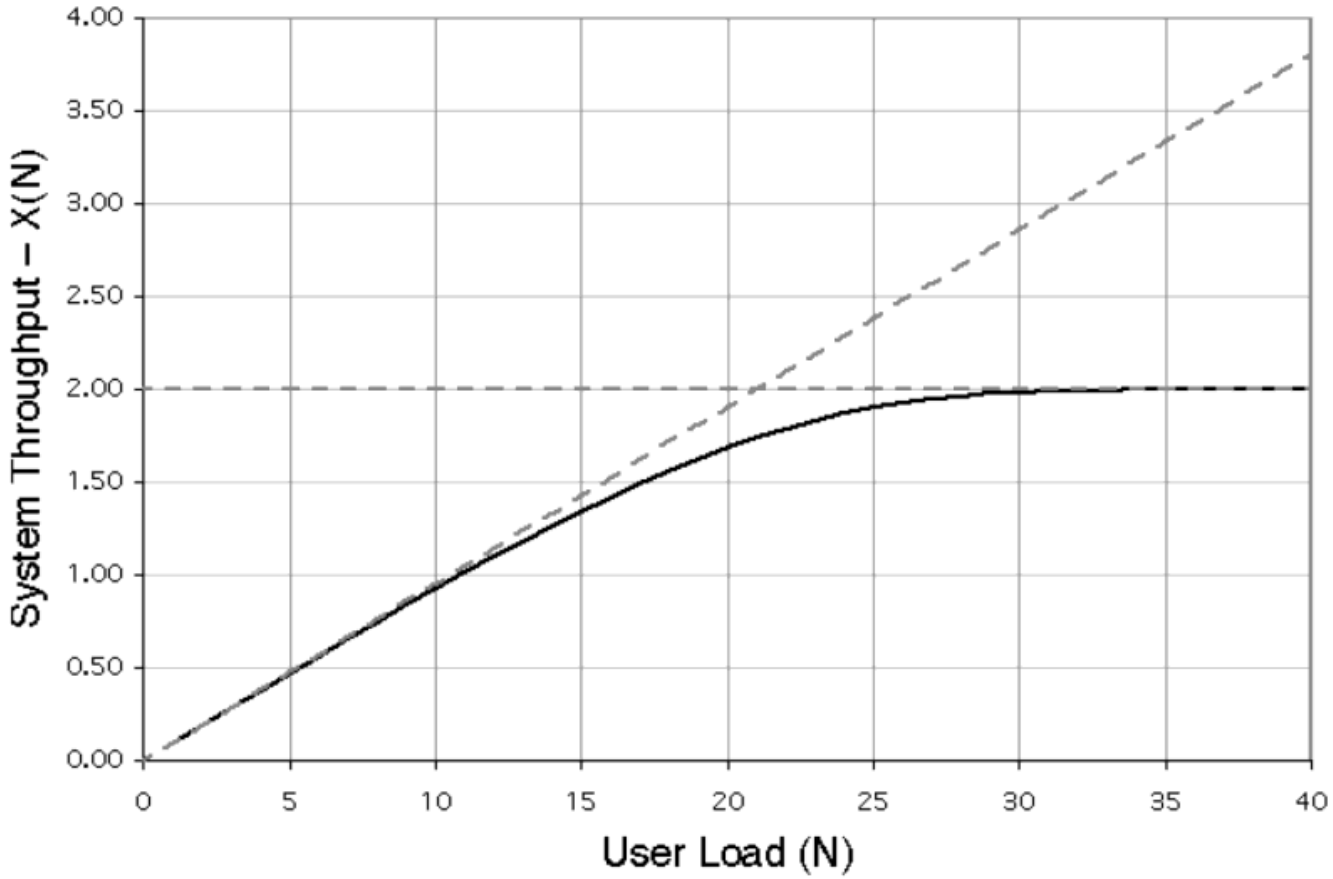
- Each client needs $Z + R$ (response time) time per request
  - Client's sending rate: $1/(R+Z)$
  - Number of jobs sent in time T: $T/(R+Z)$
- Rate for N clients: $N/(R+Z)$
  - Number of jobs sent in time T: $N*T/(R+Z)$

- ***$X = N / (R+Z)$***
- ***$R = (N/X) - Z$***

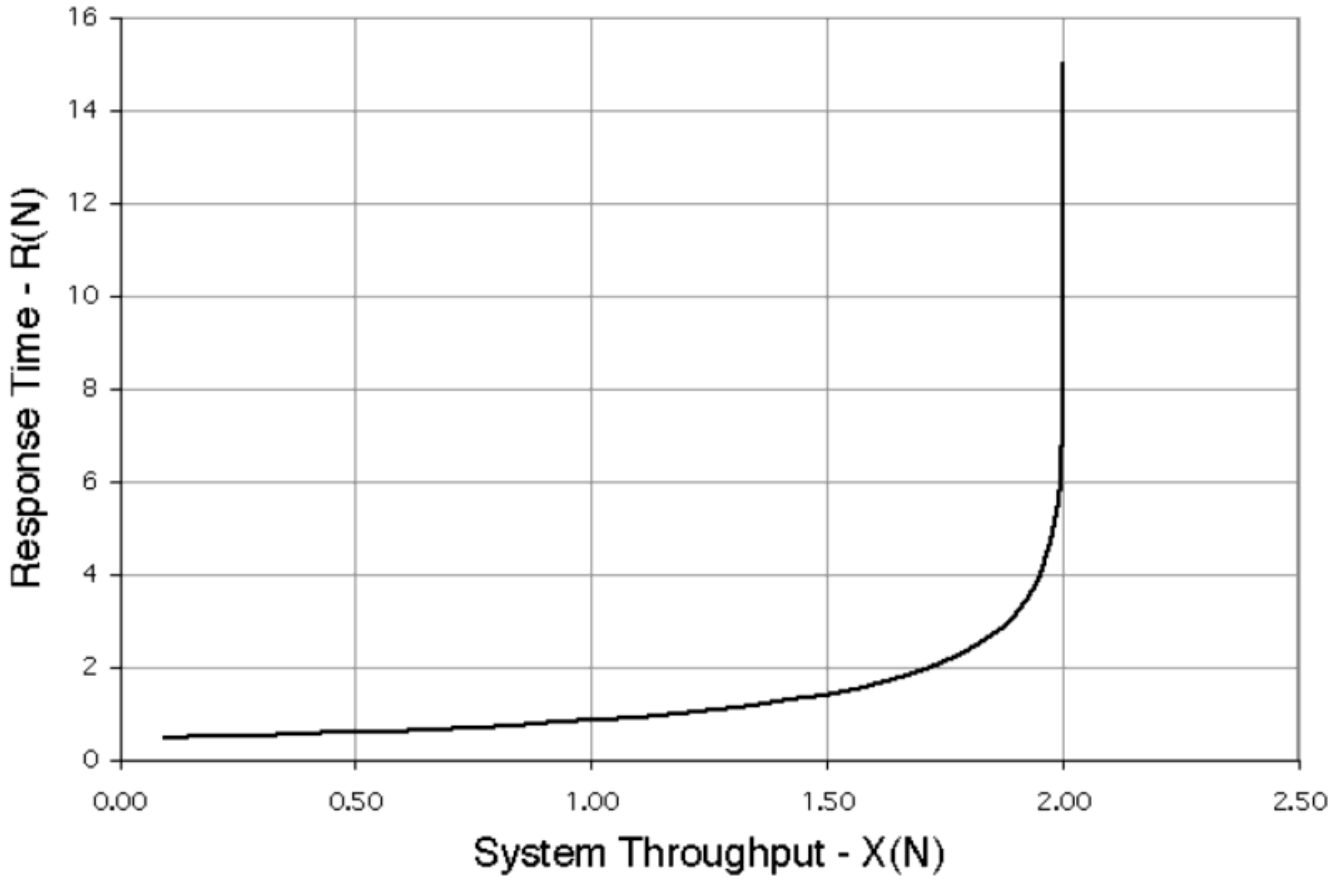
In plots…


System Response Time – R(N)
User Load (N)
15
10
5
0
-5
-10
-15
0
5
10
15
20
25
30
35
40
45
50

In plots…


System Throughput – X(N)
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
0
5
10
15
20
25
30
35
40
User Load (N)

In plots…


Response Time - R(N)
16
14
12
10
8
6
4
2
0
0.00
0.50
1.00
1.50
2.00
2.50
System Throughput - X(N)

Looks simple but…

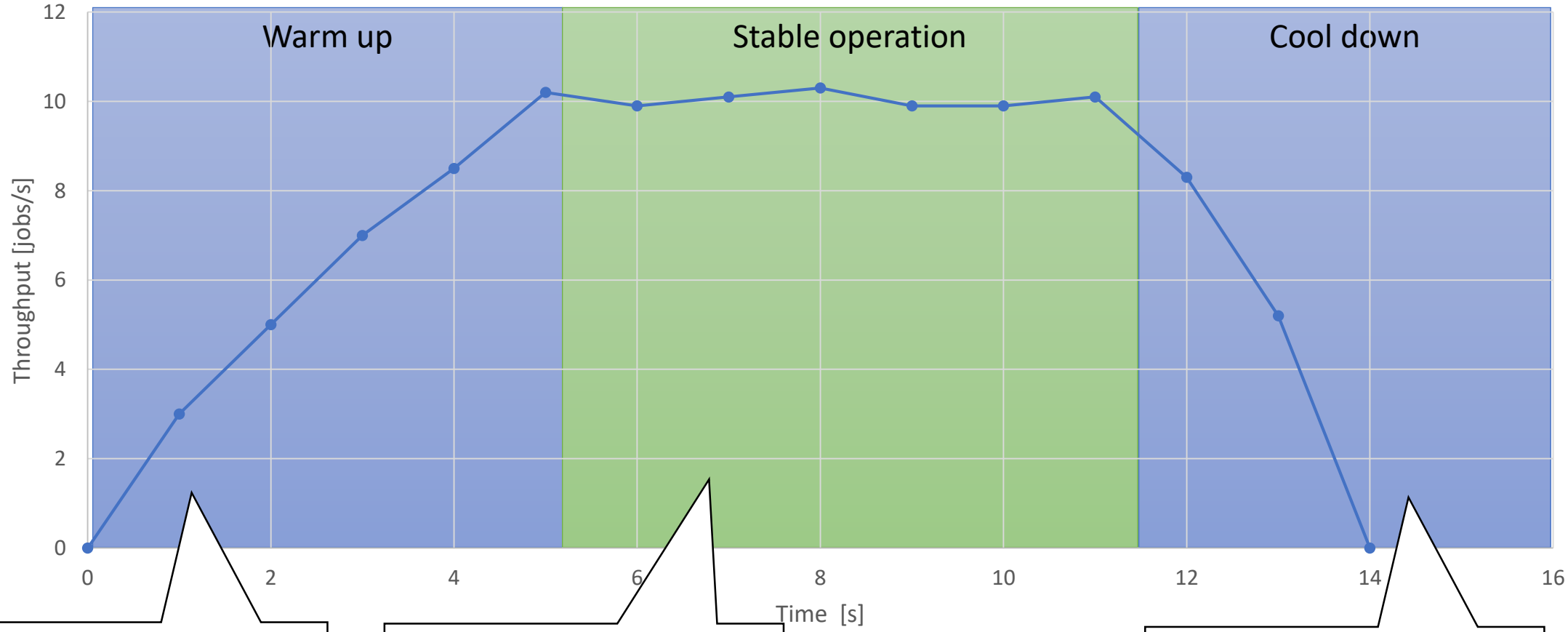
The model does not account for:

- Large variance in response times
- Different “types” of requests
- Communication delays and jitter
- Other overheads
- Failing requests, exceptions, stack overflows, etc…

System behavior over time

- When should we measure throughput/response time?

Experiment life-cycle

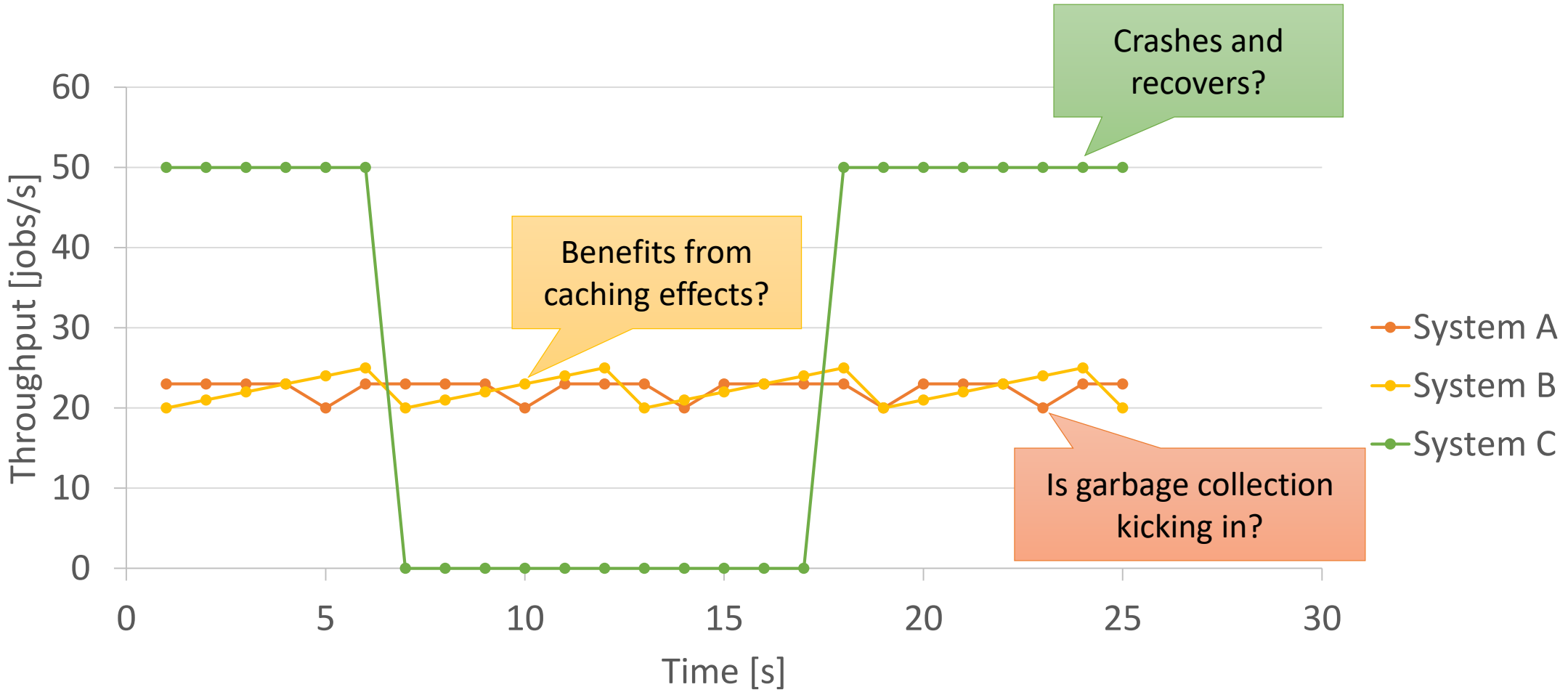


Clients starting, caches being populated, JIT compiler working in background, data is read from disk, etc.

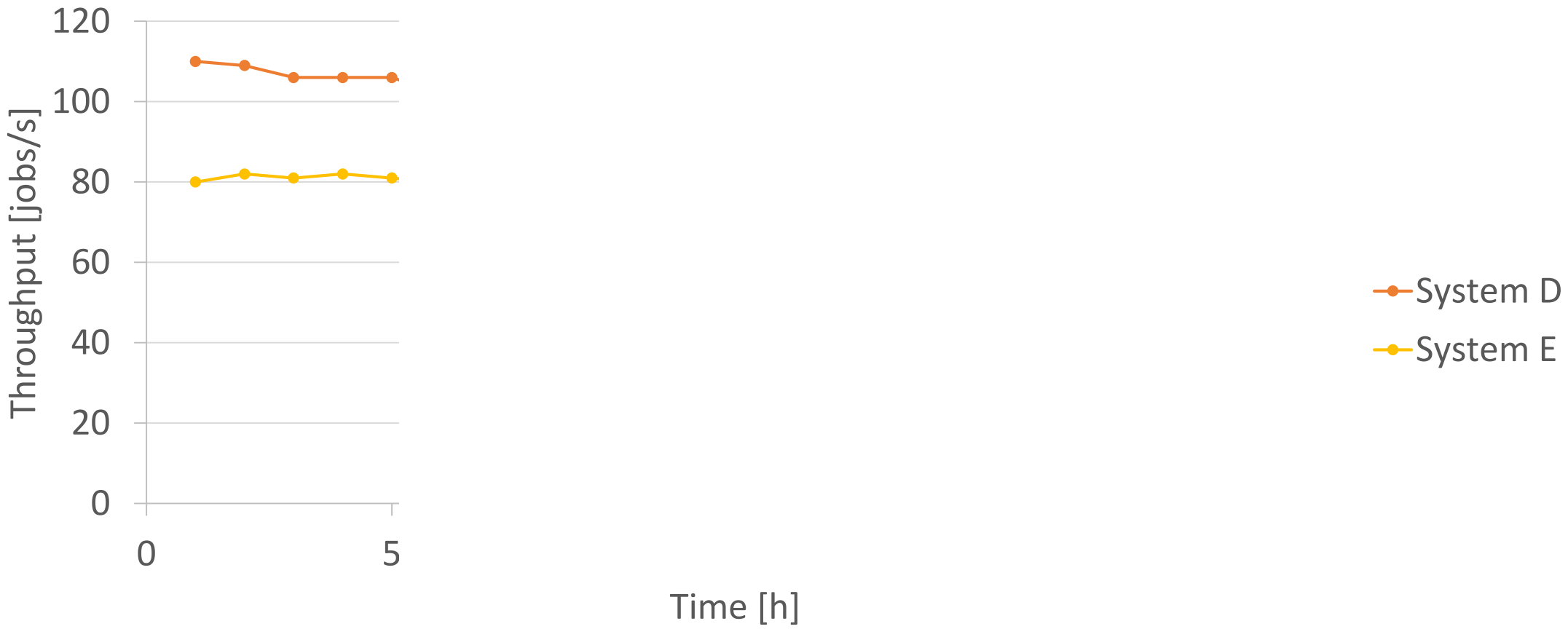
This part is the one we usually talk about when reasoning about throughput and response time!

Clients are stopping not at the same time (different classes of jobs, imperfect load balancing, etc.)

Observing a system in its stable state



Which database should we buy?



In conclusion

- We can discuss a system's behavior even if treated as black box
  - Interactive Response Time Law
  - (Later lecture: Queuing theory)
  - For deeper insights will have to consider what is inside
- Throughput/response time linked to each other in closed systems
  - Throughput is meaningful as a function of the workload
- Always aim to measure systems in steady state
  - Separate warm up and cool down phases
  - Validate that behavior is actually stable over time