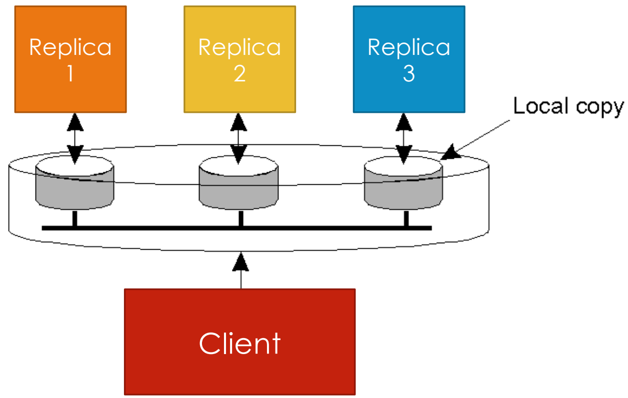
Client-Centric Consistency Models

- **Monotonic Reads**: Reads never go backwards
- **Monotonic Writes**: Writes never go backwards
- **Read your Writes**: My own writes must be visible
- **Writes follow reads**: If a write is based on a read, it must happen after it



What does the client see?

Client-Centric Consistency Models

- **Monotonic Reads**: If a process reads the value of a data item *X*, any subsequent read operation on *X* by that process will always return that same value or a more recent value

Example

Automatically reading your personal calendar updates from different servers. Monotonic Reads guarantees that the user sees all updates, no matter from which server the automatic reading takes place.

Example

Reading (not modifying) incoming mail while you are on the move. Each time you connect to a different e-mail server, that server fetches (at least) all the updates from the server you previously visited.

Client-Centric Consistency Models

- **Monotonic Writes**: A write operation by a process on a data item *X* is completed before any successive write operation on *X* by the same process.

Example

Updating a program at server S_2, and ensuring that all components on which compilation and linking depends, are also placed at S_2.

Example

Maintaining versions of replicated files in the correct order everywhere (propagate the previous version to the server where the newest version is installed).

Client-Centric Consistency Models

- **Read your Writes**: The effect of a write operation by a process on data item *X*, will always be seen by a successive read operation on *X* by the same process.

> **Example**
>
> Updating your Web page and guaranteeing that your Web browser shows the newest version instead of its cached copy.

Client-Centric Consistency Models

- **Writes follow reads**: A write operation by a process on a data item *X* following a previous read operation on *X* by the same process, is guaranteed to take place on the same or a more recent value of *X* that was read.

> Example
>
> If I read and then comment on an article, nobody should see my comment until after they see the article

Prof. Tim Wood & Prof. Roozbeh Haghnazar

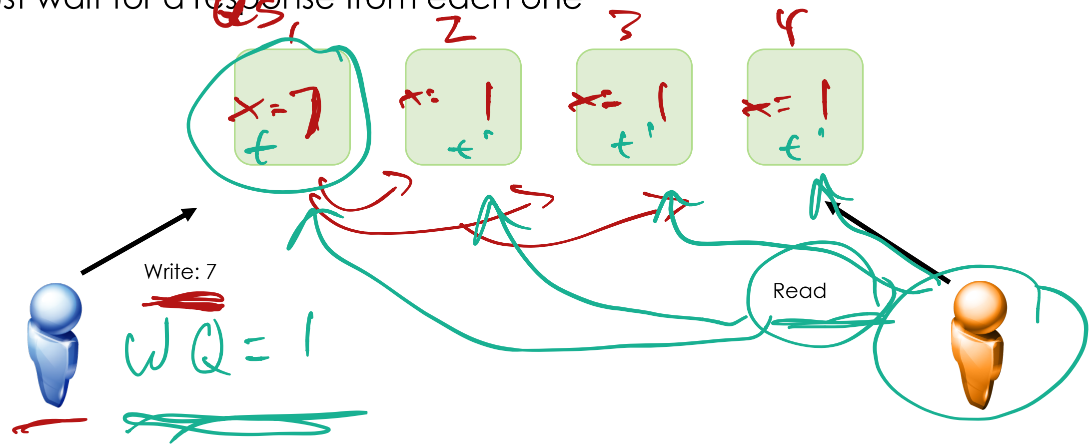
Quorum Replication

Quorum Based Systems

- Quorum: a set of responses that agree with each other of a particular size
- Crash fault tolerance: Need a quorum of **1**
  - **f** others can fail (thus need f+1 total replicas)
- Data fault tolerance: Need a quorum **f+1**
  - **f** others can fail (thus need 2f+1 total replicas)
  - Need a majority to determine correctness

Quorum

- 4 Replicas
  - Some nodes might be temporarily offline
- How many replicas to send to for a read or write?
  - Must wait for a response from each one



Dynamo DB

- Object Store from Amazon
  - Technical paper at SOSP 2007 conference (top OS conference)
- Stores **N** replicas of all objects
  - But a replica could be out of date!
  - Might be saved across multiple data centers
  - Gradually pushes updates to all replicas to keep in sync
- When you read, how many copies, **R**, should you read from before accepting a response?
- When you write, how many copies, **W**, should you write to before confirming the write?

Dynamo DB

- Read and Write Quorum size: how will the system behave?
  - R=1
  - W = 1
  - R = N/2+1
  - R=1, W = N
  - R=N, W=1

1 2 3 4 5 6 7 8 9 10

W=5 R=6

N=10

W=?

R+W>N

to gaurantee consistent reads

N=10, $R = \frac{10}{2} + 1 = 6$

$W = \frac{N}{2} = 5$

Dynamo DB

- Read and Write Quorum size:

- R=1 — fastest read performance, no consistency guarantees
- W = 1 — fast writes, reads may no be consistent
- R = N/2+1 (reading from majority)
- R=1, W = N - slow writes, but reads are consistent
- R=N, W=1 - slow reads, fast writes, consistent
- Standard: N=3, R=2, W=2
  - Ensures overlap

QUORUM

- How do N, R, and W affect:
- **Performance:** small R, W → better performance
- **Consistency:** if R + W > N → strong consistency
- **Durability:** if W ~ N → better durability
- **Availability:** larger N → more availability

DynamoDB lets the user tune these for their needs

Quorum

- How do N, R, and W affect:
- **Performance:**
  - low R or W -> higher performance
  - for a fixed R or W: higher N gives higher performance
  - higher N means more synchronization traffic
- **Consistency:**
  - R + W > N — guarantees consistency
  - R+w << N — much less likely to be consistent
- **Durability:**
  - N=1 vs N=100, more N = more durability
- **Availability:**
  - Higher N or W => higher availability

Distributed Systems
CS6421
Performance

Prof. Tim Wood and Prof. Roozbeh Haghnazar

Includes material adapted from Van Steen and Tanenbaum's Distributed Systems book

Final Project

Questions?

- Implementation phase!
  - Get coding!
  - Keep your code in GitHub
- Schedule meetings with us!
  - Especially if you realize you can't achieve what you originally planned

- Timeline
  - Milestone 0: Form a Team - 10/12
  - Milestone 1: Select a Topic - 10/19
  - Milestone 2: Literature Survey - 10/29
  - Milestone 3: Design Document - 11/8
  - **Milestone 4: Final Presentation - 12/14**

https://gwdistsys20.github.io/project/

Last Time...

- Replication and Consistency
  - Why replicate
  - What is consistency?
  - Consistency Models
  - Quorum Replication
- Exam
  - Avg: 90%

This Time...

- Performance in Dist. Systems
  - Introduction
  - Performance metrics
  - Models
  - Architectures

But first we need to finish a bit of consistency!

DistSys Challenges

- **Heterogeneity**
- Openness
- Security
- Failure Handling
- Concurrency
- **Quality of Service**
- **Scalability**
- Transparency

Performance Challenges

Problem

- Amazon: 100 ms extra latency costs 1% in sales
- Bing: 2s slowdown would reduce the revenue by 4.3%
- Google: (August 16 2013) 5 minute down time cost 545k$
- Increasing 400ms web search latency caused a decrease of about 0.74% in Google's search frequency

What is Performance?

- Marriam-webster:
  - the execution of an action
  - the fulfillment of a claim, promise, or request
  - the manner of reacting to stimuli

- Wikipedia
  - In computing, computer performance is the amount of useful work accomplished by a computer system. Outside of specific contexts, computer performance is estimated in terms of accuracy, efficiency and speed of executing computer program instructions.

What is Performance?

- Performance considers:
  - Latency (transmission delay)
  - Bandwidth (maximal transmission capacity)
  - Throughput (average transmission rate)
  - Response time (time to see result of action)

What is the first step???

How can we choose the best service as a client?

How can we make sure that we have provide the best service as a provider?

How can we understand that what are we going to design as an engineer?

QoS & SLA

Service Level Agreements (SLA)

- Service Level Agreements are fundamental to an effective cloud utilization and especially business customers need them to ensure risks and service qualities are prevented respectively provided in the way they want.
- The confirmed SLAs serve as a basis for compliance and monitoring of the QoS.
- Due to the dynamic cloud character, the QoS attributes must be monitored and managed consistently

How can I describe and measure the QoS?

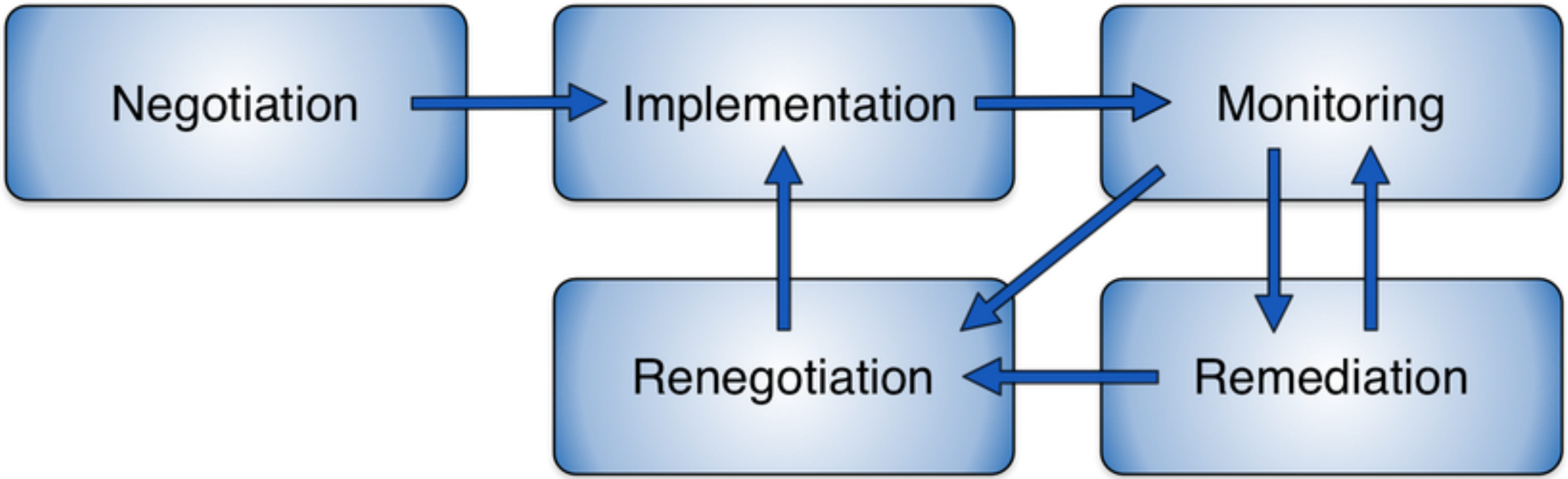
KPI

How can we define SLA

- NIST has pointed out the necessity of SLAs, SLA management, definition of contracts, orientation of monitoring on Service Level Objects (SLOs) and how to enforce them. (https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication500-292.pdf)

- There are two major specification for describing SLAs
  - Web Service Level Agreement (WSLA) Language Specification, was developed by IBM with the focus on performance and availability metrics.
  - WS-Agreement (WS-A) was developed by the Open Grid Forum in 2007.
  - The newest update, which is based on the work of the European SLA@SOI project.

SLA Life cycle



Service Level Objectives

- Service Level Objectives (SLOs) are a central element of every service level agreements (SLA), which include:
  - negotiated service qualities (service level)
  - corresponding Key Performance Indicators.
- SLOs contain the specific and measurable properties of the service, such as ***availability***, ***throughput or response time*** and often consist of ***combined or composed attributes***

SLO Characteristics

- SLOs should thereby have the following characteristics:
  - Repeatable
  - Measurable
  - Understandable
  - Significant
  - Controllable
  - Affordable
  - Mutually acceptable
  - Influential

We need KPI

SLO Characteristics

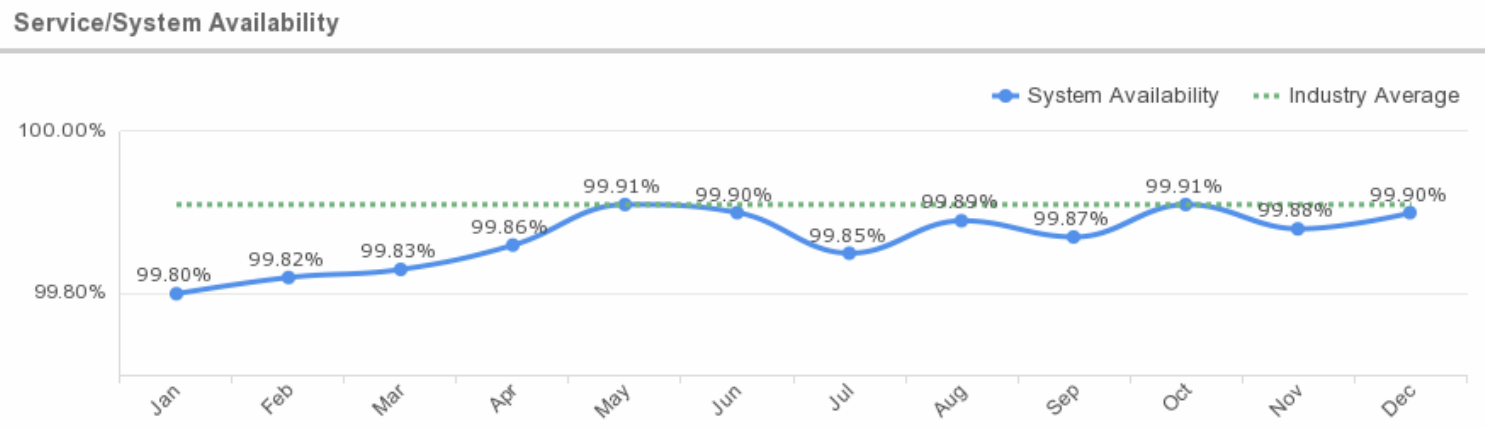
- A valid SLO specification might, for instance, look like this:
  - *The IT system should achieve an availability of 98% over the measurement period of one month. The availability represents thereby the ratio of the time in which the service works with a response time of less than 100ms plus the planned downtime to the total service time, measured at the server itself.*

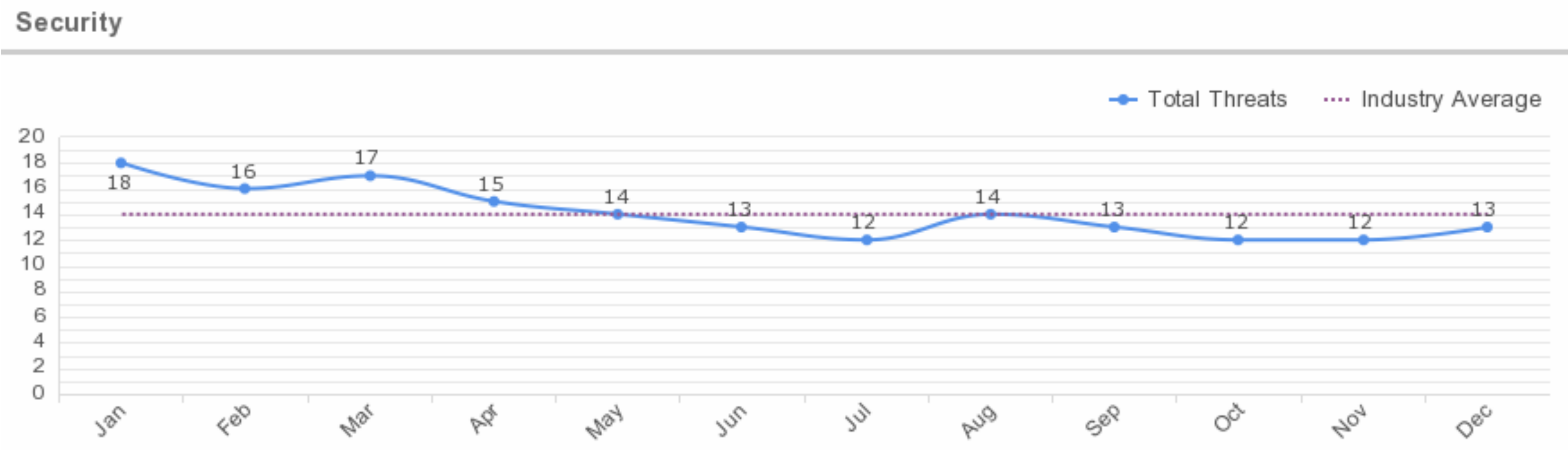
Key performance metrics

- General Service KPIs
- Network Service KPIs
- Cloud Storage KPIs
- Backup and Restore KPIs
- Infrastructure as a Service KPIs

General Service KPIs

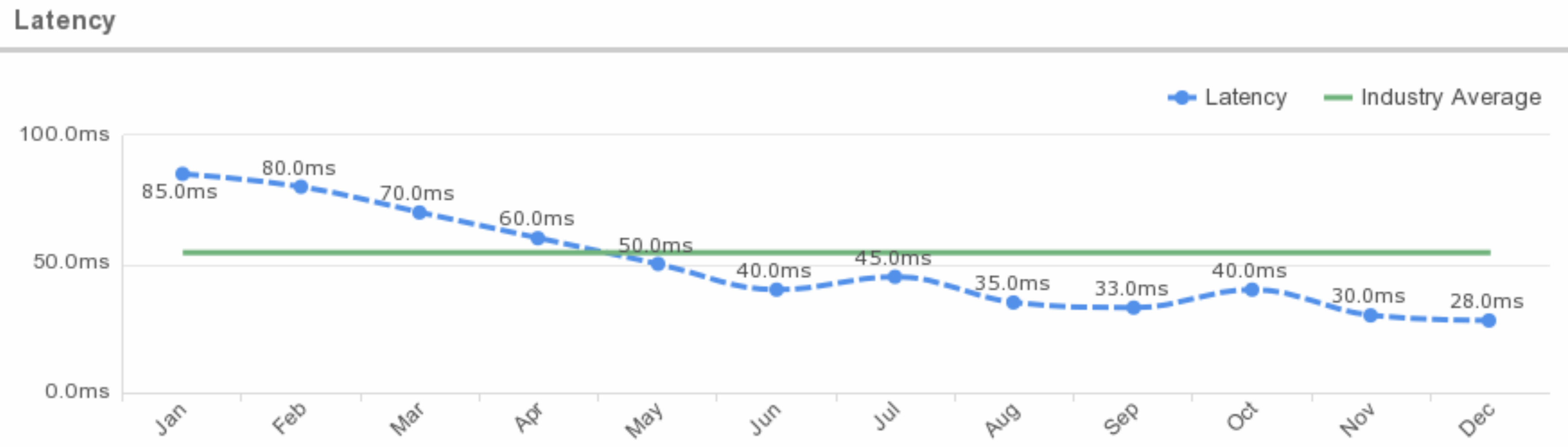
- Basic Services
- Security
- Service and Helpdesk
- Monitoring
- Etc.

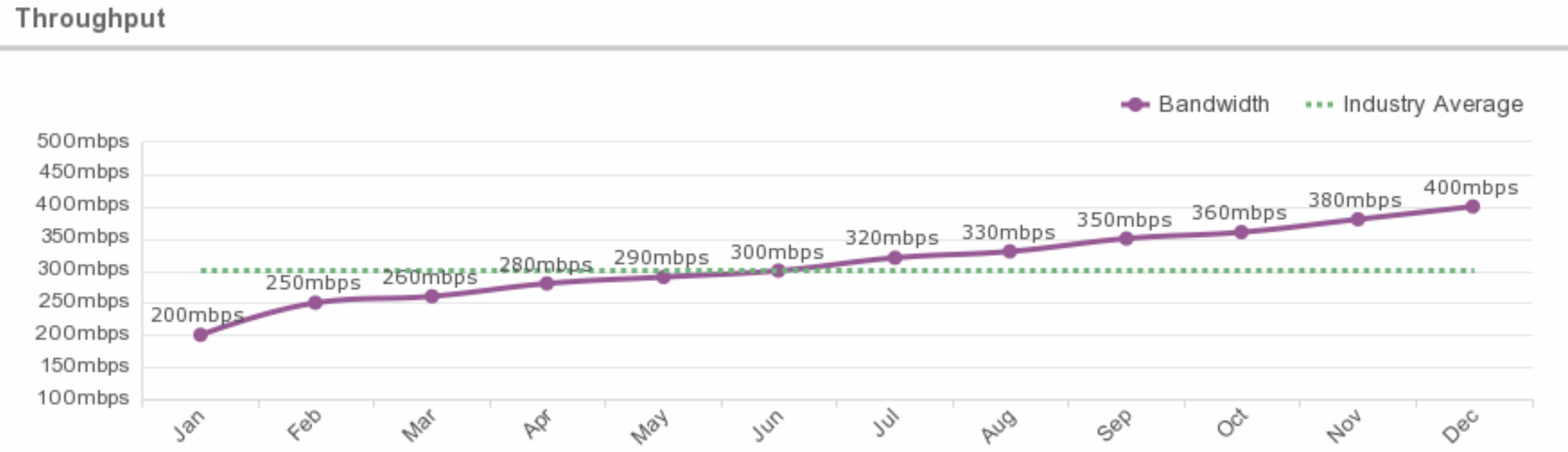




Network Service KPIs

- Round Trip Time
- Response Time
- Packet Loss
- Bandwidth
- Throughput
- Network Utilization
- Latency
- Etc.





Cloud Storage KPIs

- Response Time
- Throughput
- Average Read Speed
- Average Write Speed
- Random Input / Outputs per second (IOPS)
- Sequential Input / Outputs per second (IOPS)
- Free Disk Space
- Provisioning Type
- Average Provisioning Time

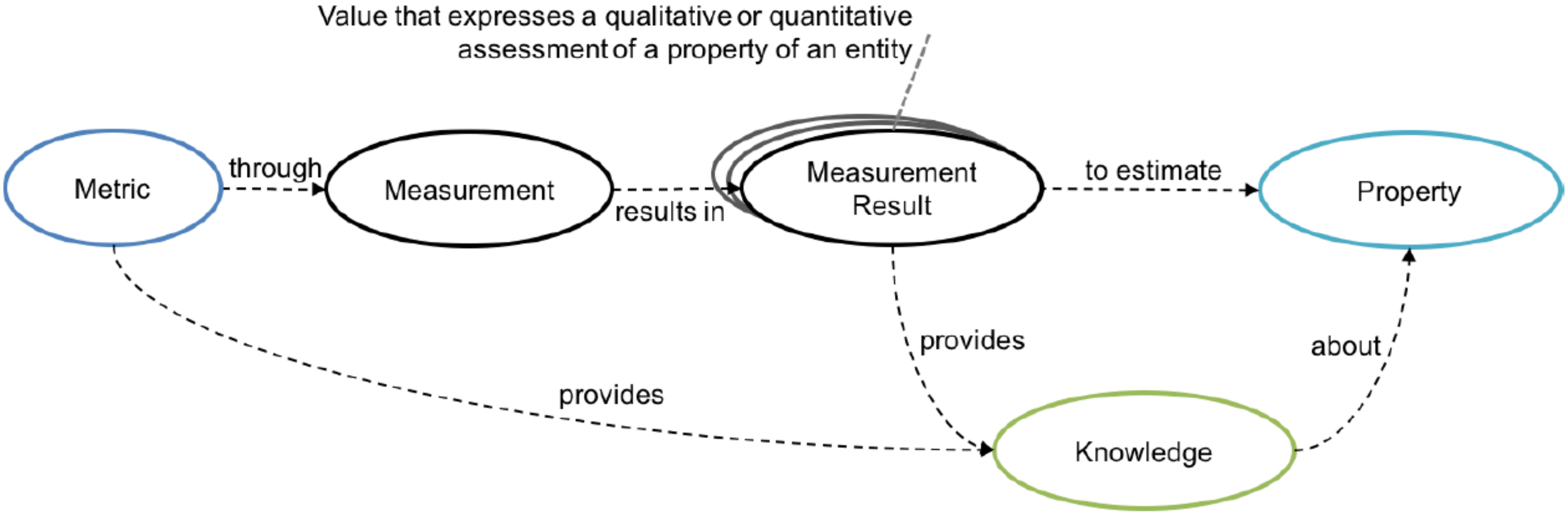
Backup and Restore KPIs

- Backup Interval
- Backup Type
- Time To Recovery
- Backup Media
- Backup Archive

Infrastructure as a Service KPIs

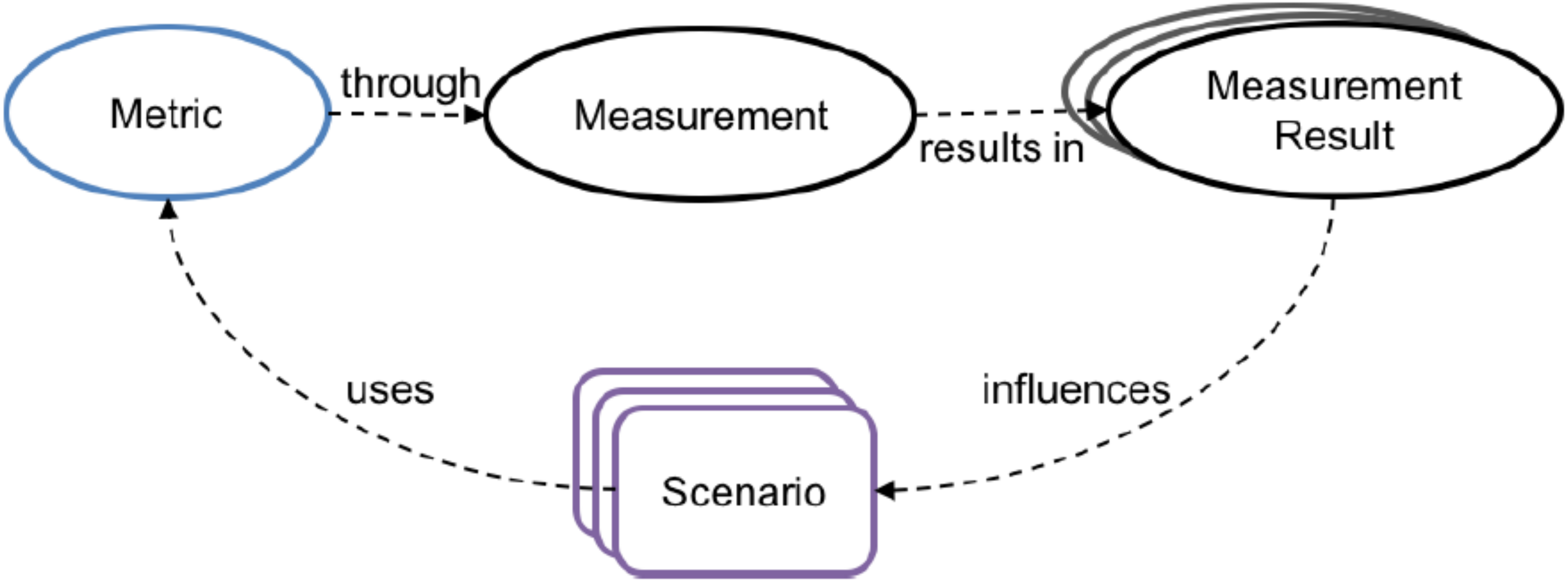
- VM CPUs
- CPU Utilization
- VM Memory
- Memory Utilization
- Minimum Number of VMs
- Migration Time
- Migration Interruption Time
- Logging

Metric and property



https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.500-307.pdf

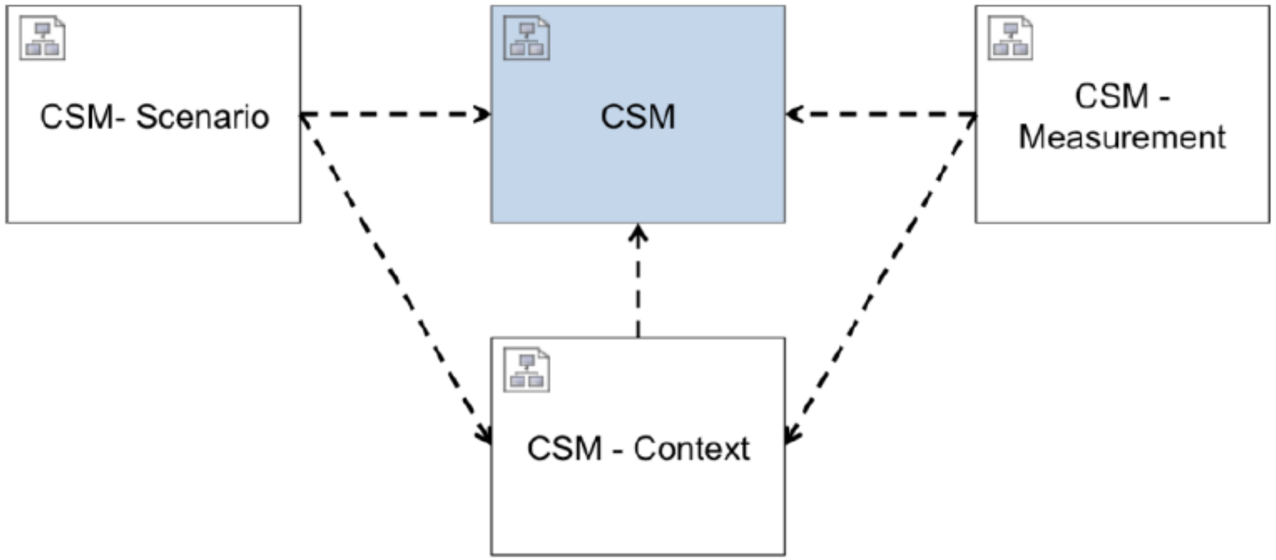
Scenario and metric



https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.500-307.pdf

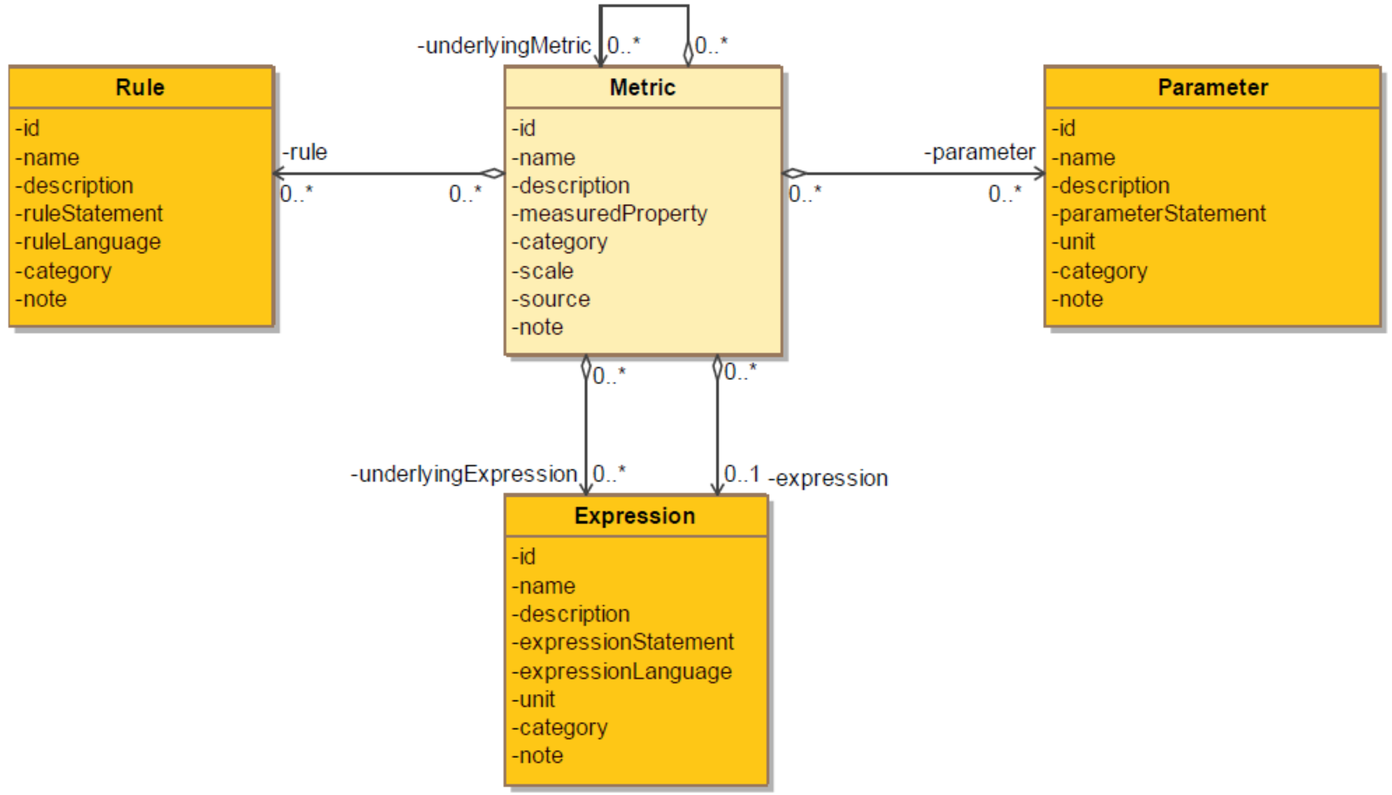
Cloud Service Metric Ecosystem Model

- CSM: The description and definition of a standard of measurement (e.g. metric for customer response time)
- CSM Context: The context related to using the CSM in a specific scenario. (e.g. objectives and applicability conditions of the customer response time metric)
- CSM Measurement: The use of the CSM to make measurements (e.g. the measurement of response time property based on the customer response time metric)
- CSM Scenario: The use of the CSM in a scenario (e.g. the selection and use of the customer response time metric in an SLA)



https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.500-307.pdf

Cloud Service Metric (CSM)



https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.500-307.pdf

SUMMARY

- SLA: The agreement you make with the clients and users
- SLO: The objectives your team must hit to meet that agreement
- KPI: Key performance indicators

References:

- http://www.thinkmind.org/articles/emerging_2013_3_30_40082.pdf
- https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.500-307.pdf
- https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication500-292.pdf
- https://www.cocop-spire.eu/content/key-performance-indicator-kpi-and-impact-evaluation-distributed-production-systems-%E2%80%93-importance-feedback
- https://www.atlassian.com/incident-management/kpis
- https://www.atlassian.com/incident-management/kpis/sla-vs-slo-vs-sli
- https://cloud.google.com/blog/products/gcp/sre-fundamentals-slis-slas-and-slos
- https://cloud.google.com/blog/products/gcp/availability-part-deux-CRE-life-lessons

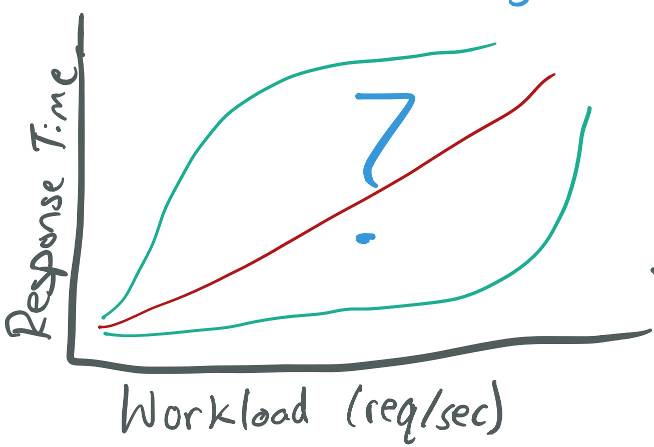
Prof. Tim Wood & Prof. Roozbeh Haghnazar

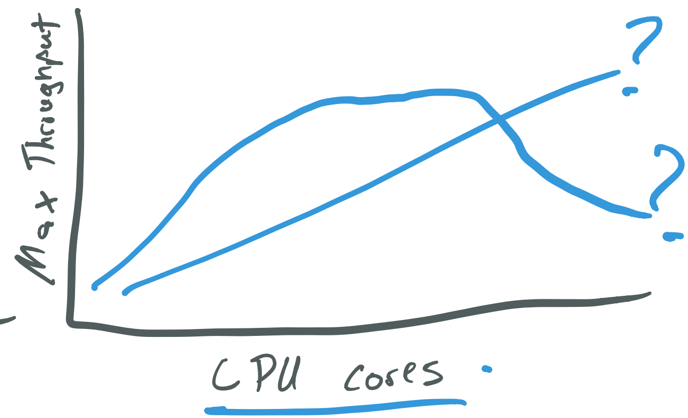
Performance Modeling

Performance Models

- Measuring performance is not always enough
- We want to **Predict** these metrics in advance

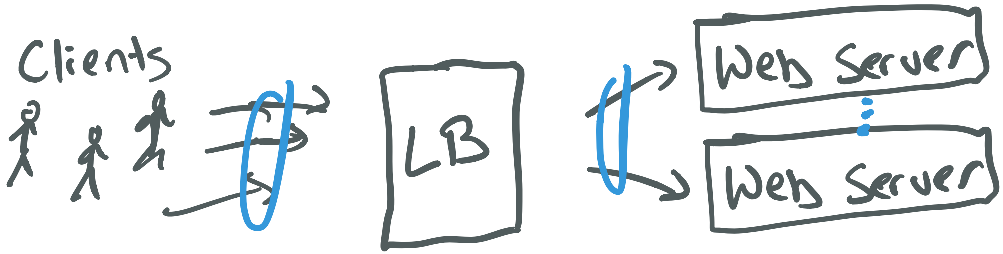
  - How will response time change if my workload doubles?
  - How much memory / CPU do I need to get a target throughput?





Model Scenario

Consider this web application:

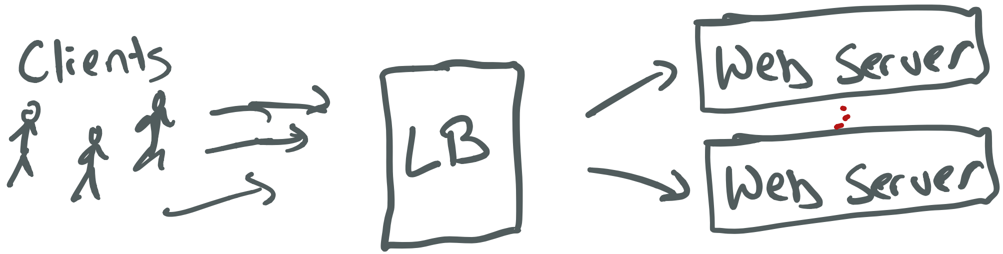


What will affect its performance?

- Clients & workload — # clients, req/sec, type of req
- Network - BW & Latency
- App details
- Server details
- LB algorithm
- # Servers

Model Scenario

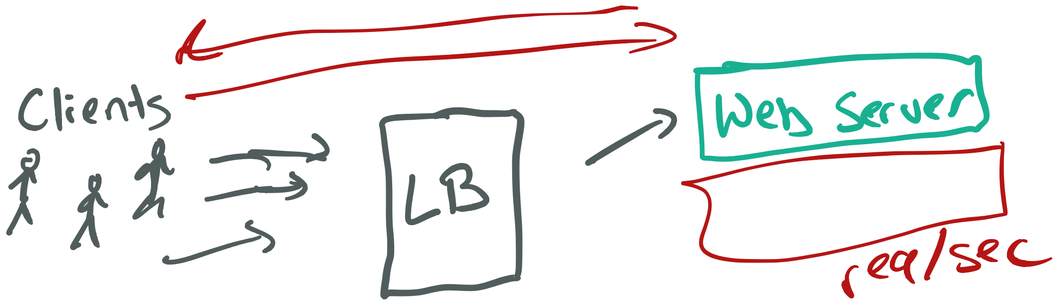
Consider this web application:



What will affect its performance?

- How many replicas? How many resources?
- Dependencies between components?
- Incoming workload characteristics
- Effectiveness of Load Balancer

Throughput Modeling



- How can we predict the maximum throughput of the system?
- Suppose each request takes

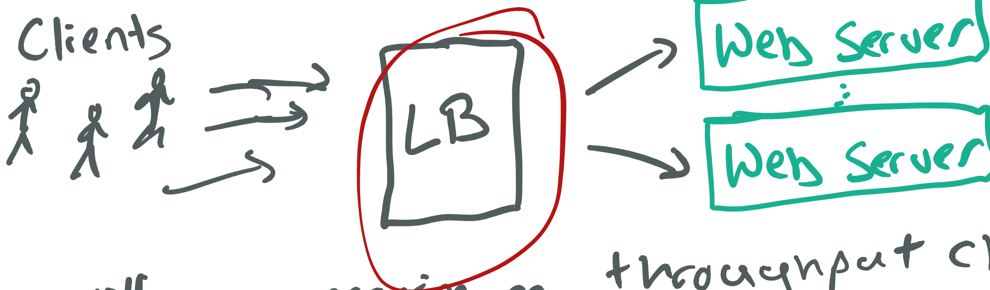
  ~~5~~ ½ sec millisec on Web server

  ↑ Service time

Throughput ≈ # threads / service time

per replica

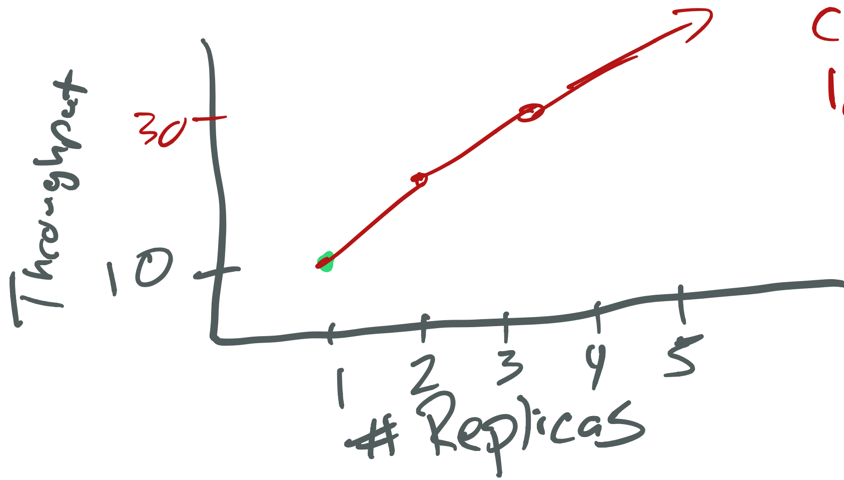
Throughput Modeling



S.T = 100 ms = 0.1s

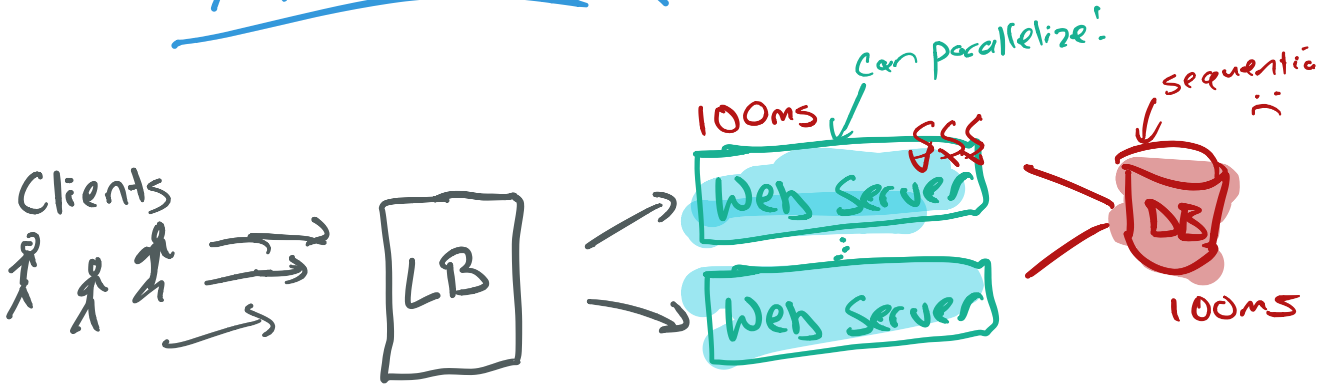
$\frac{1}{0.1} = 10 \frac{req}{sec}$

- How will our maximum throughput change when we add more replicas?



Can I always linearly scale?

Amdahl's Law

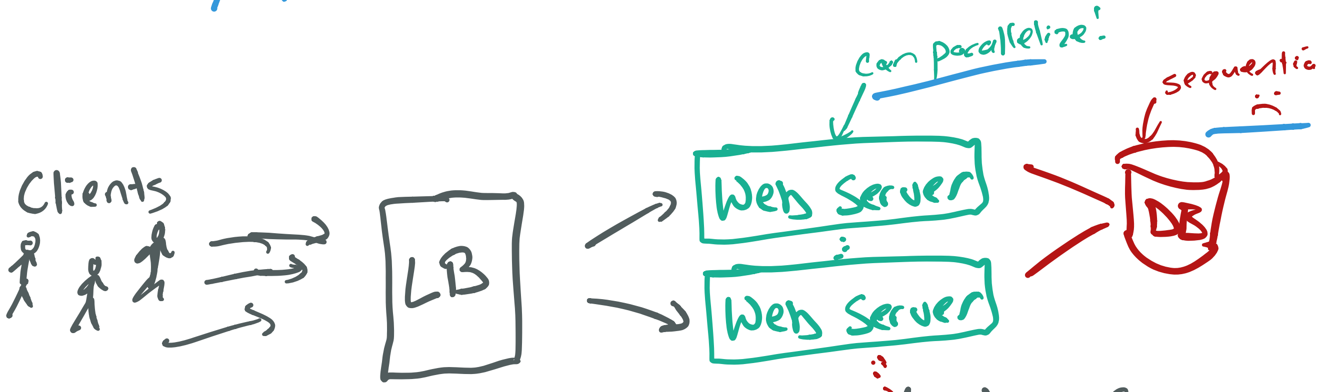


- How will our maximum throughput change if we add a database?

WS + DB

100ms + 100ms

Amdahl's Law

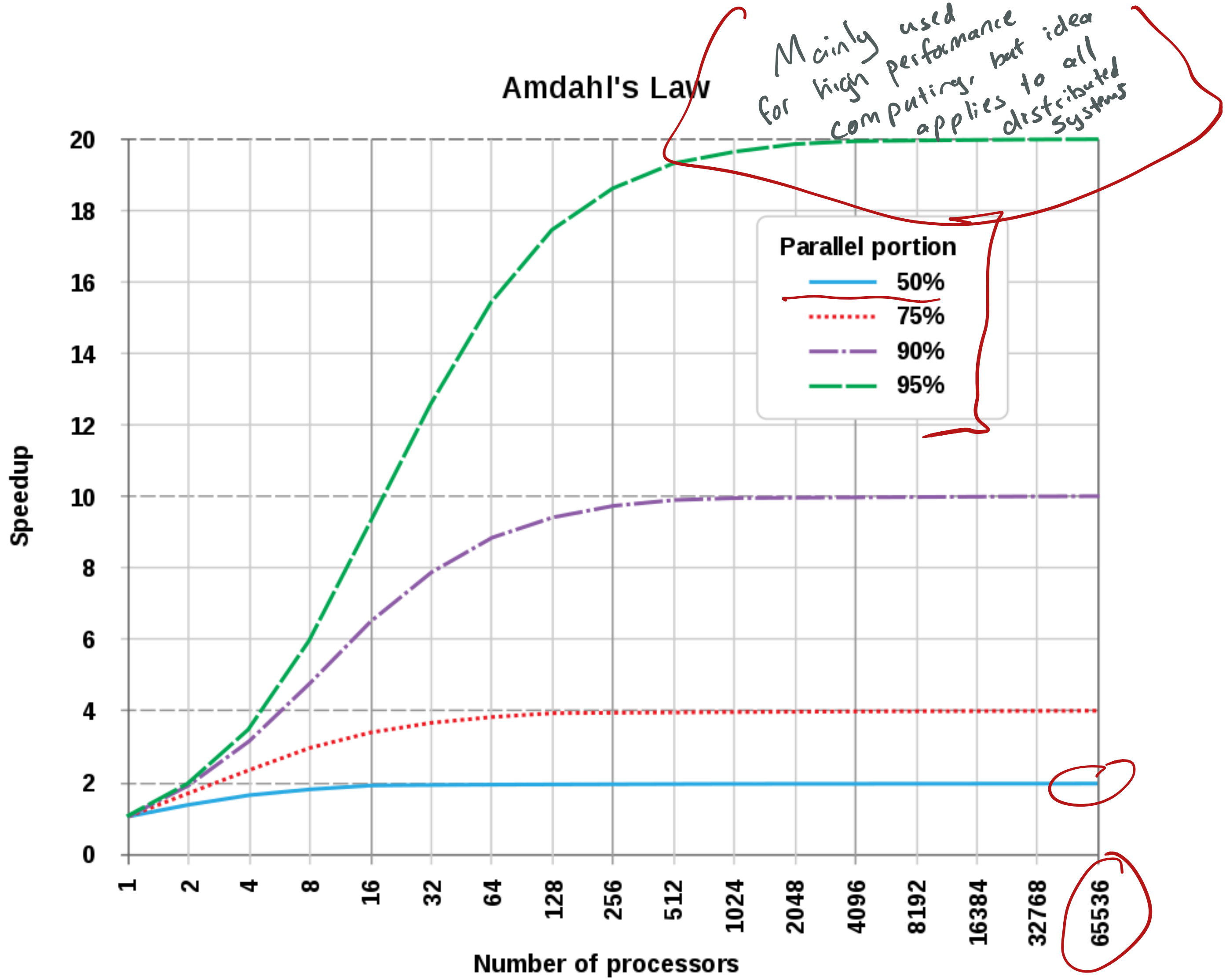


- How will our maximum throughput change if we add a database?

$$\text{Speedup} \le \frac{1}{1 - p}$$

p = proportion of the program that can be parallelized

If 95% of processing is in web server, then max speedup is $\frac{1}{1 - .95} = \frac{1}{.05} = 20x$


Amdahl's Law
Mainly used for high performance computing, but idea applies to all distributed systems
Parallel portion
50%
75%
90%
95%
Speedup
0
2
4
6
8
10
12
14
16
18
20
Number of processors
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536

Response Time

- Is Response Time the same as Service Time?

$$R = S + N + Q$$

R: Response Time

S: Service Time

N: network RTT + Transfer time

Q: Queuing Delay → Waiting, Load Dependent

constants

Response Time

- Is Response Time the same as Service Time?

$$R = S + Q + N$$

R: Response Time

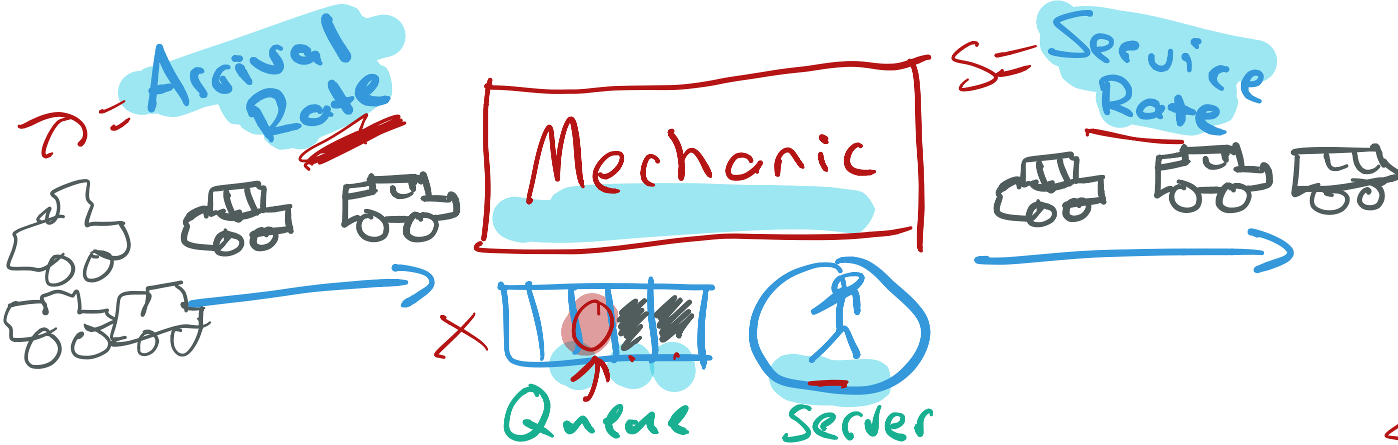
S: Service Time (Constant)

Q: Queuing Delay (Depends on Workload)

N: Network Delay (Constant)

Queuing Delay

- Q = time a request spends waiting to be served
- L = # of requests in queue



- If Arrival Rate << Service Rate: $L \approx 0$ (small)
- If Arrival Rate >> Service Rate: $L \approx \infty$

$$Q = (L + 1) \times S$$

(L: length; S: service time)

Queuing Theory

- Branch of Mathematics / Operations Research Studying systems of Queues

- Many equations to help predict performance of Systems that can be modeled as queues and processors

Capacity = max TPut

Load < Capacity

$$Q_{\text{Delay}} = \frac{1}{\text{capacity} - \text{load}} =$$

$$R = S + Q + N$$

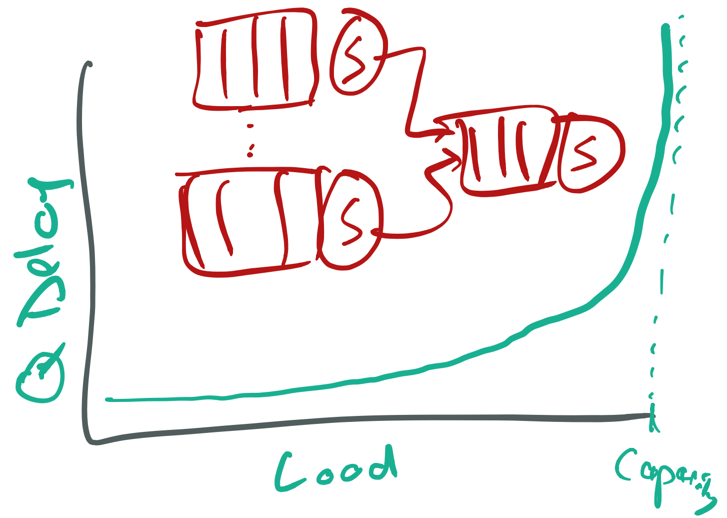
Queuing Theory

- Branch of Mathematics / Operations Research Studying systems of Queues

- Many equations to help predict performance of Systems that can be modeled as queues and processors

$$\underset{\text{Delay}}{Q} = \frac{1}{\text{capacity} - \text{load}}$$

$$R = S + Q + N$$



Little's Law

$$L = \lambda \times R$$

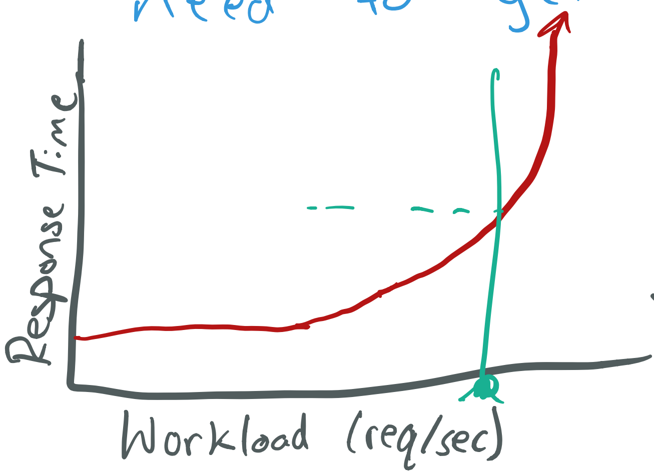
L: Q length; λ: Arrival Rate; R: Response Time

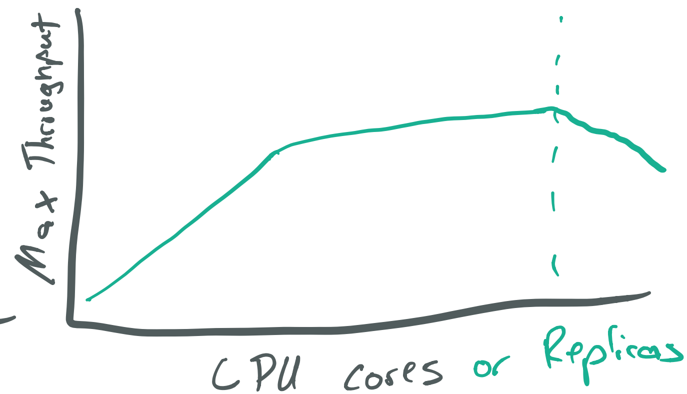
- Suppose 100 req/sec arrive on average + each request takes 5ms to complete

$$L = 100 \frac{req}{sec} \times .005\, sec = .5$$

Performance Models

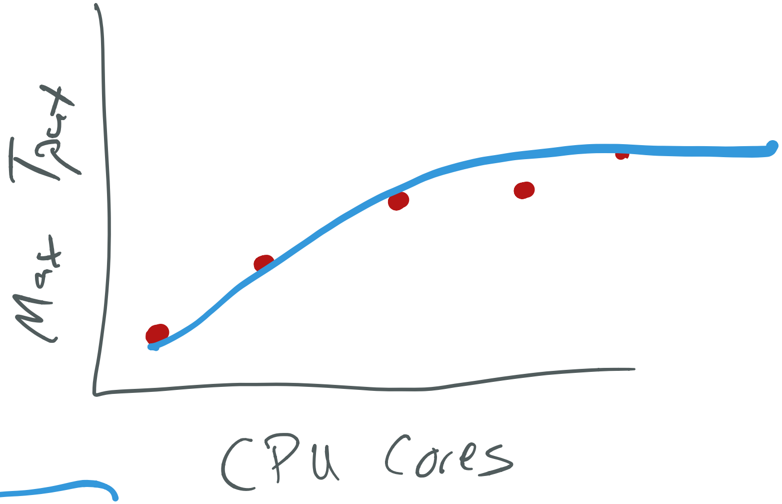
- Measuring performance is not always enough
- We want to **Predict** these metrics in advance
  - How will response time change if my workload doubles?
  - How much memory/CPU do I need to get a target throughput?

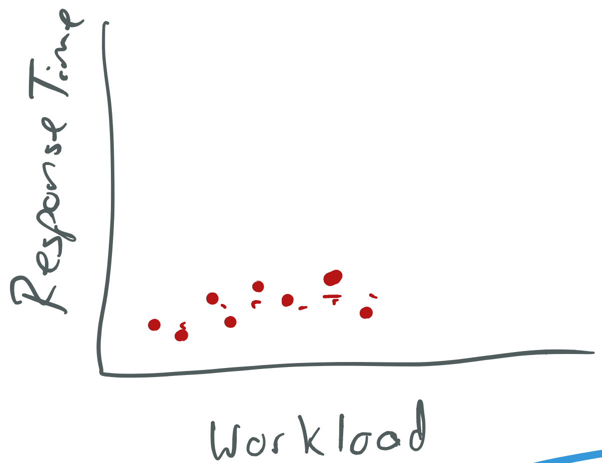




ML Models

- Data-driven performance models
- Gather system metrics over time + predict future behavior





+ Captures true system details, interference, etc

– Non-linear behavior is difficult to predict

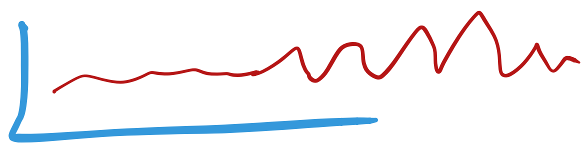
Workload Models

- How do we characterize the requests coming in to a system?

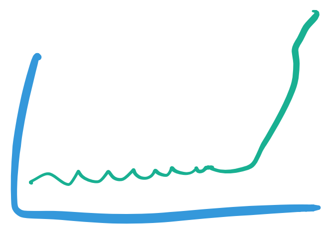
Uniform - consistent pattern or rate over time ← 10 req/sec

Diurnal - traffic affected by day/night cycle

Bursty - unpredictable surges in traffic



Flash Crowds - extreme & unpredictable bursts



Skewed - uneven mix of request types ← 90% reads, 10% writes

☆ Workload type will affect performance