Methodology

Andrea Marin

a.a. 2024/2025

Università Ca' Foscari Venezia
MSc in Computer Science
Software performance and scalability

What is performance evaluation?

What is performance evaluation?

- Performance evaluation involves quantifying the service delivered by a computer or a communication system or a software architecture
- Examples
  - What is the expected response time of a web service under a certain work condition?
  - What's the amount of power consumed by a certain app for Android?
  - What's the throughput of a certain Internet router?

Key factors: load, metrics, goals

- The performance of a system depends on the workload (load)
- The load characterises the quantity and the nature of requests submitted to the system
- Example: the number of requests received by a web server in a unit of time is the *intensity of the workload*
- Common misconception: The performance degrades linearly with the workload
    - This is a huge mistake!

Performance vs. Scalability

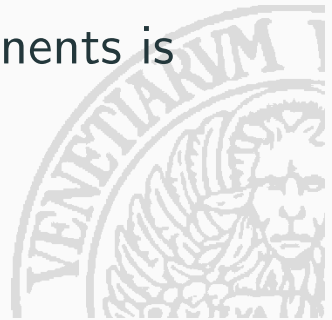
- Performance measures how fast and efficiently a software system can complete certain computing tasks
- Scalability measures the trend of performance with increasing load
- SLA: Service Level Agreements, i.e., the set of quantitative indices that a software/hardware systems must satisfy
  - Example: the response to a request must be given within x seconds with 99% of probability
- QoS: The quality of service are non-functional properties of the software systems, including the response time, the throughput, the energy consumption, etc.

Metric

- A performance metric is a measurable quantity that precisely captures what we want to measure
  - Depends on the systems and on the goals
- For each metric we may be interested in the average, a percentile, the worst case and so on

Examples of metrics

- Response time
  - The time elapsed from the moment in which a task arrives at a system to the moment in which the task is served
- Throughput
  - The number of tasks served in the unit of time
- Energy consumption
  - The energy consumed by the system for serving a task or per unit of time
- Utilisation
  - The fraction of time that a system or one of its components is busy
- A combination of the previous ones
  - multi-dimensional metric



Goals

- Comparison of different system's configurations
  - Under the same workload, which performs better?
    - No need of a precise estimation of the workload since it is hypothetical
- System dimensioning
  - In order to guarantee a certain Quality of Service (QoS) how shuld we design a computer system?
  - The workload must be estimated precisely because it strongly influences the conclusions of the analysis

Factors

- Given the goals, the metric, and the load we must define the factors
- The factors are the elements of the characteristics of the load that affect the performance metric
- Examples:
  - Number of users
  - Mobility of the smartphones
  - Types of the connections of the users of a web server
  - ...

Evaluation methods

- Measurements
  - Difficult to perform without changing the system, sometimes impossible
- Discrete event simulation
  - Works on general hypothesis, usually precise, requires high skills and is very time consuming
- Analytical methods
  - Usually much faster than simulation, more restrictive assumptions, mathematically bases

Performance patterns

Performance patterns

- Performance patterns are traits that are common in many scenarios
- The can be summarised in 4 different types:
  1. Bottlenecks
  2. Congestion collapse
  3. Competition side effects
  4. Latent congestion collapse

Bottlenecks

- Remember: *The speed of a convoy is that of the slowest vessel*
- The performance of a system is often limited by that of its weakest components. The weakest component is the bottleneck
- Example: A Linux-Apache-Mysql-PHP web server consists of two independent servers with Apache/PHP and one server with shared database. The former two can serve 10 requests/s each and the database 12 requests/s. When the number of requests increases the database will be the bottleneck.
- Identifying bottlenecks is not always simple

Congestion collapse

- Congestion happens when the intensity of the workload is higher than the system capacity
- Congestion collapse is a frequent reaction of the system to the congestion. As the load increases the utility of the congested system decreases (instead of remaining at its maximum)
- Example: in a server, the requests which do not receive a reply in x seconds will be newly submitted. As the load increases, the requests will wait longer for being served so they will newly submitted. However, their siblings are still in the server's queue and will be served eventually. This causes a waste of the computational power that causes the congestion collapse

Competition side effects

- Users compete for system resources. This pattern happens when by increasing certain resources some users are able to increase their load, which in turn it decreases the performance of competing users
- Example: a server receives the requests on two connection lines with capacity C_1 and C_2. If the second line is replaced with a line with capacity $C_3 >> C_2$, does the system performance improve? From the point of the of the customers using line 1, the effect will be of competing with more customers and hence experiencing worse performance

Latent congestion collapse

- This pattern happens as a consequence of removing a bottleneck of the system
- A consequence of the removal is that a new bottleneck may appear that causes a congestion collapse
- Basically the first bottleneck was acting as an implicit admission control protecting the system from the collapse

Summing up

1. Define your goal
2. Identify the factors
3. Define your metrics
4. Define the workload
5. Know your bottleneck (focus your attention on them)
6. Know your system well