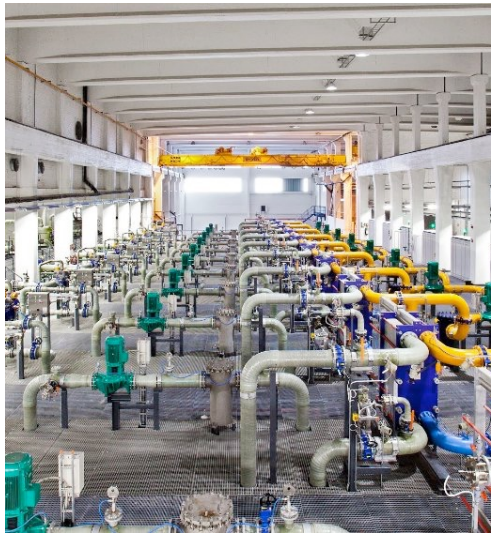
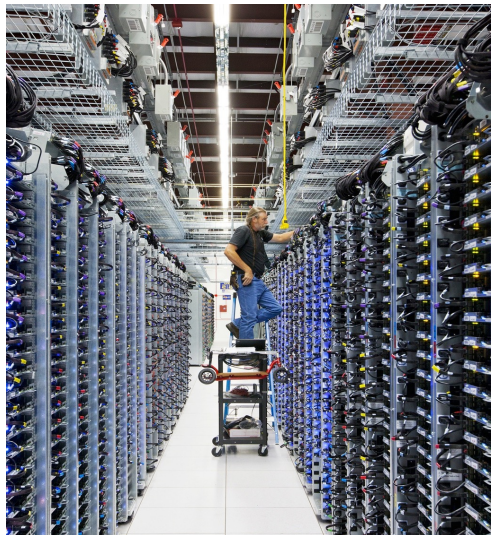
heig-vd

HEIG-VD | TIC – Technologies de l'Information et de la Communication

Administration IT — High performance systems


Academic year 2015/16

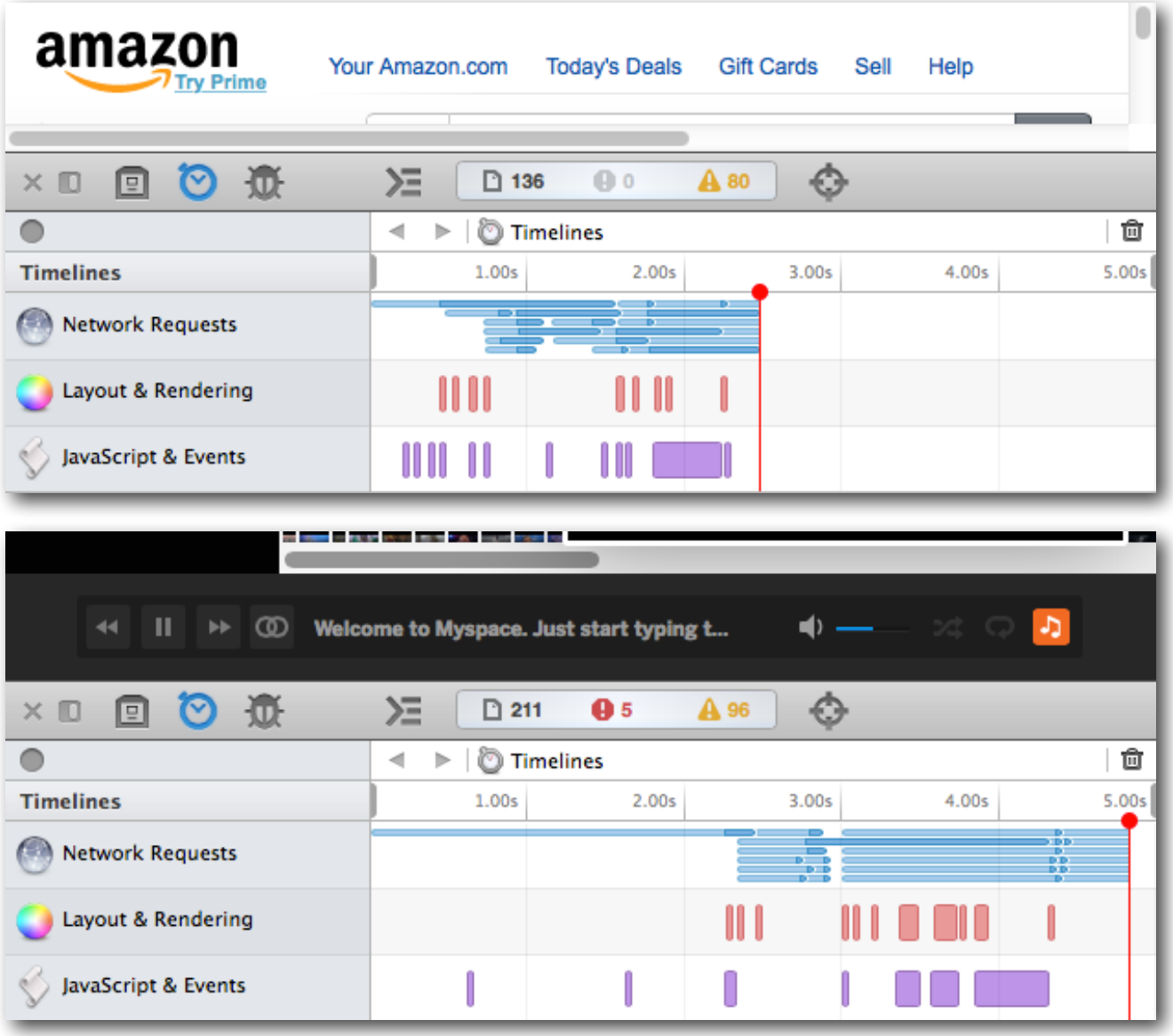
High performance systems

Introduction

- The most popular web sites like Google Maps, Yahoo! Mail, Facebook, YouTube and Amazon are examples of scalable high performance systems.
  - When you access these sites you get a high-quality user experience: the sites respond very quickly. Interaction with the site is fluid. You don't have to wait for pages to download, they appear almost instantly. The sites are **high performance**.
  - They are also built to **scale**. They access petabytes of data and they send terabits per second to millions of users worldwide.

High performance systems

amazon.com vs. myspace.com



High performance systems

Human perception

- In the field of User Experience it is established that there are three main time limits which are determined by human perceptual abilities.
  - **0.1 second** is about the limit for having the user feel that the system is **reacting instantaneously**, meaning that no special feedback is necessary except to display the result.
  - **1.0 second** is about the limit for the **user's flow of thought** to stay uninterrupted, even though the user will notice the delay. Normally, no special feedback is necessary during delays of more than 0.1 but less than 1.0 second, but the user does lose the feeling of operating directly on the data.
  - **10 seconds** is about the limit for **keeping the user's attention** focused on the dialogue. For longer delays, users will want to perform other tasks while waiting for the computer to finish, so they should be given feedback indicating when the computer expects to be done. Feedback during the delay is especially important if the response time is likely to be highly variable, since users will then not know what to expect.

Source: Jakob Nielsen — Usability Engineering

High performance systems

The cost of latency

- Performance is taken very seriously by big web sites because it has a measurable impact on the bottom line.
- Google Search and Microsoft Bing conducted latency experiments in 2009
  - For a set of users artificially increase server response time for a duration of four weeks.
  - Results:
    - With increased delay users click less and search less → Less ad revenue
    - Even after the delay was removed, the users still had -0.21% fewer searches → A slow user experience affects long-term behavior.
  - The results were so bad that the study was ended prematurely.
- Shopzilla reported in 2009 on a year-long performance redesign
  - Improved page loading time by 5 seconds (from ~7 seconds to ~2 seconds)
  - Results:
    - 25% increase in page views
    - 7-12% increase in revenue
    - 50% reduction in hardware

High performance systems

The cost of latency

Impact of delays at Bing search server on user behavior

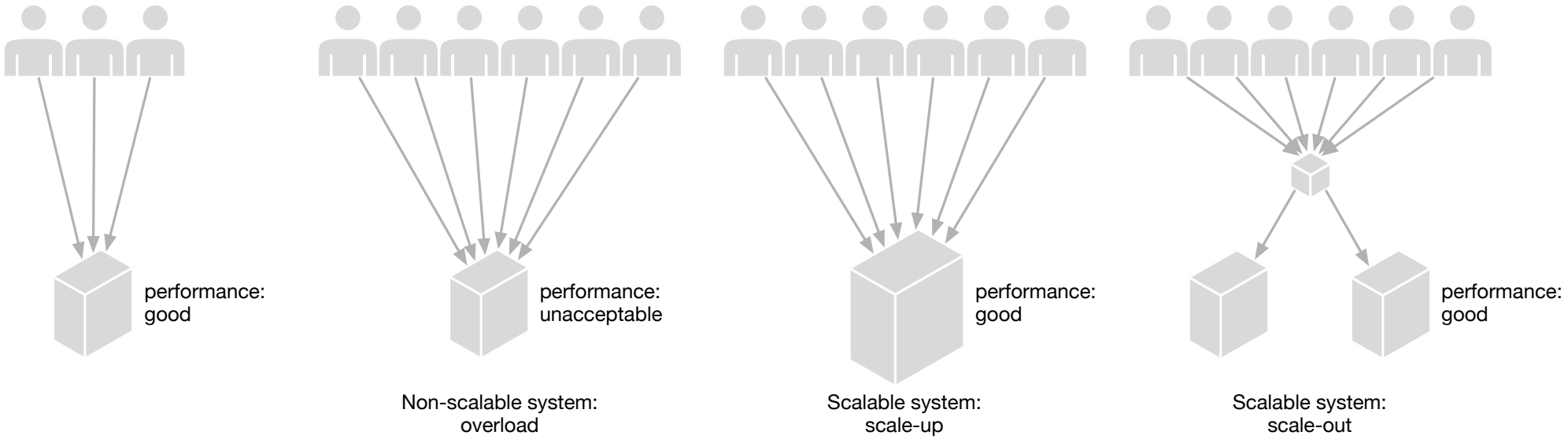
Server delay (ms)	Increased time to next click	Queries / user	Any clicks / user	User satisfaction	Revenue / user
50	—	—	—	—	—
200	500	—	-0.3%	-0.4%	—
500	1200	—	-1%	-0.9%	-1.2%
1000	1900	-0.7%	-1.9%	-1.6%	-2.8%
2000	3100	-1.8%	-4.4%	-3.8%	-4.3%

Source: Eric Schurman, Jake Brutlag, "Performance Related Changes and their User Impact", Velocity 2009

Scalability

Definition

- A system is called **scalable** if it is able to handle a growing amount of work (load, traffic) in a capable manner (continuing to perform satisfactorily), or if it can be enlarged to accommodate that growth.



Non-scalable system:
overload

Scalable system:
scale-up

Scalable system:
scale-out

Scalability

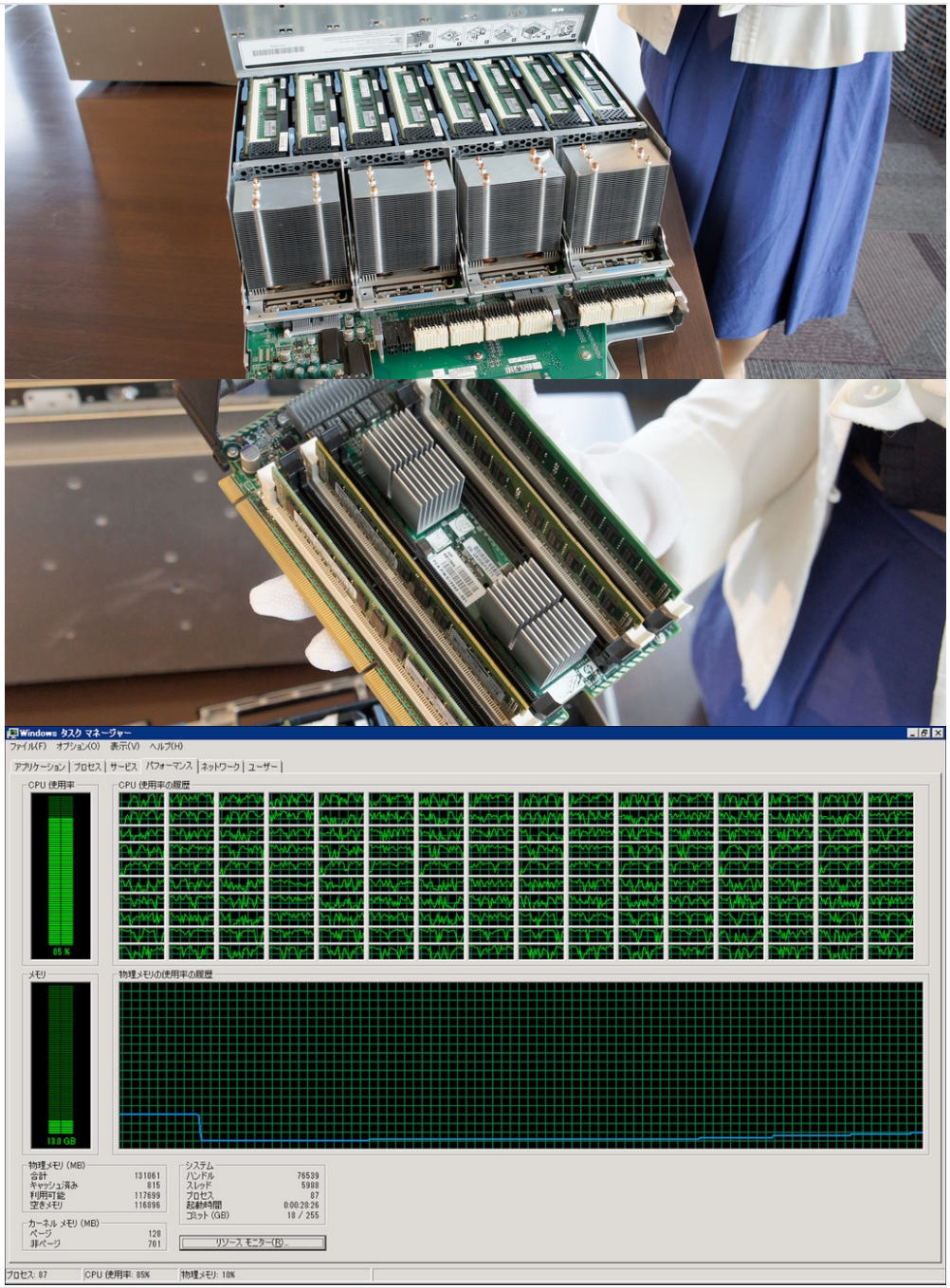
Vertical scaling (scale-up) and horizontal scaling (scale-out)

- Two approaches to scale a system:
  - Use a more capable computer: faster CPU, more cores, more memory, bigger disk, ...
    - Aka vertical scalability or scale-up
    - Gets quickly very expensive
    - Limited by computer architecture
  - Use more computers that work in parallel
    - Aka horizontal scalability or scale-out
    - Enables use of cheap commodity hardware
    - Almost unlimited scalability

Scalability

Vertical scaling example

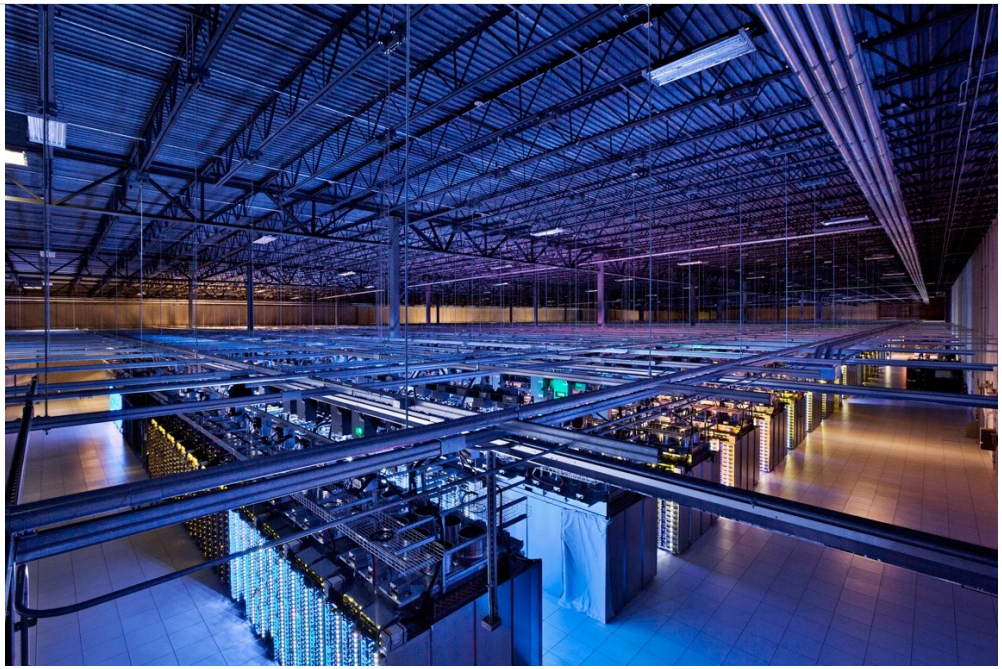
- Example: Most powerful x86 servers as of 2012
  - Bull bullion
  - HP ProLiant DL980 G7
  - Fujitsu PRIMEQUEST 1800EC2
  - Oracle Sun Fire X4800
  - IBM System x3850 X5
  - Cisco UCS C460 M2
  - Maximal specs:
    - 16 Intel Xeon E7 processors
      - 2.4 GHz
      - 10 cores per processor
    - 4 TB RAM
    - 28 TB hard disk storage
  - Cost between $20k and $500k

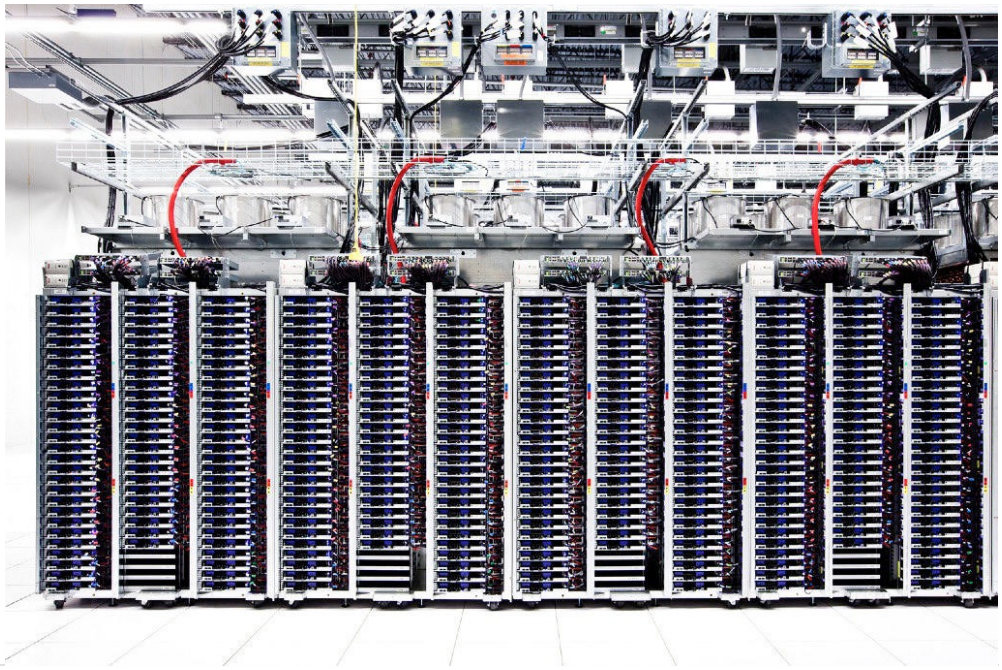


Scalability

Horizontal scaling example

- Modern datacenter (Google, Amazon, Facebook, Apple, ...). Typical numbers:
  - Datacenter contains ~50'000 servers
  - One server has
    - 2 CPUs
    - 16 GB RAM
    - 2 TB hard disk storage
  - Cost per server ~$1.5k





Scalability

Scalability vs. high performance

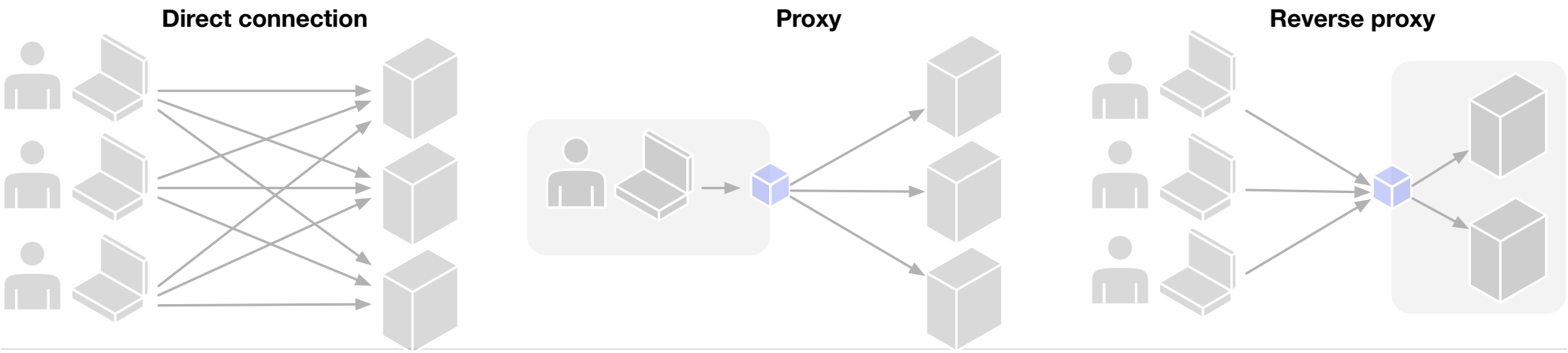
- Scalability and performance are related. Are they synonyms? Is a scalable system automatically a high performance system?
  - Example: Imagine a web site with a load balancer and a server. The site consists only of a single static web page. When the server is asked to serve the page it sleeps for a second, then serves the page.
    - The web site is not high performance because of the one second response time.
    - But it is scalable because servers can be added in case the traffic increases.

Load balancing

Proxy and reverse proxy

- In a client-server system a **proxy** is a host deployed on the client side that
  - appears to the client as if it were a server,
  - appears to the server as if it were a client,
  - forwards
    - requests it receives from the client to the server,
    - answers from the server to the client,
  - is often used for caching and filtering.

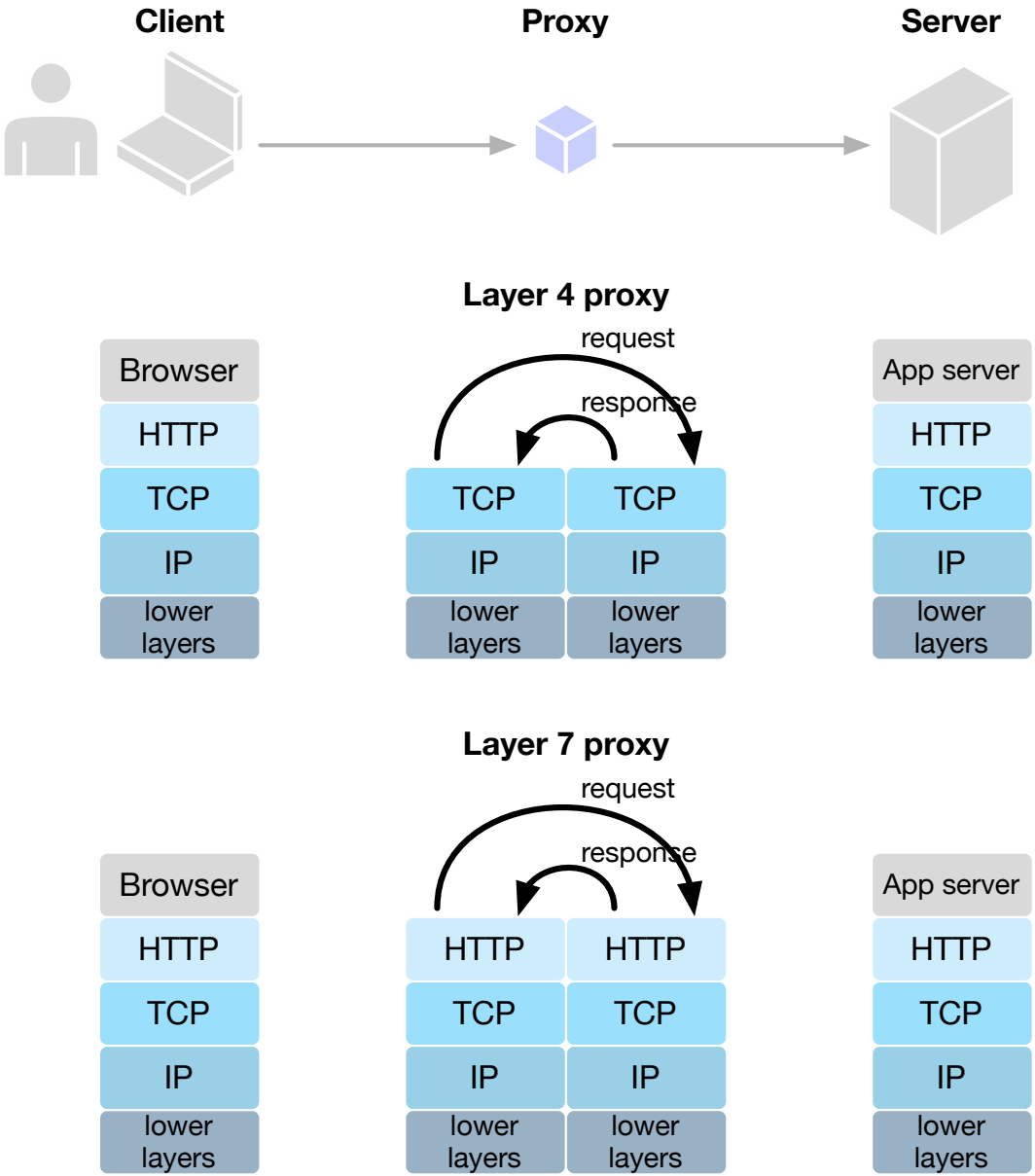
- A **reverse proxy** is a proxy deployed by a web site that
  - receives all requests from the clients (the DNS name of the site resolves to it),
  - forwards them to a number of servers to distribute the load → load balancing,
  - is often also used for caching and filtering.



Load balancing

Types of load balancers

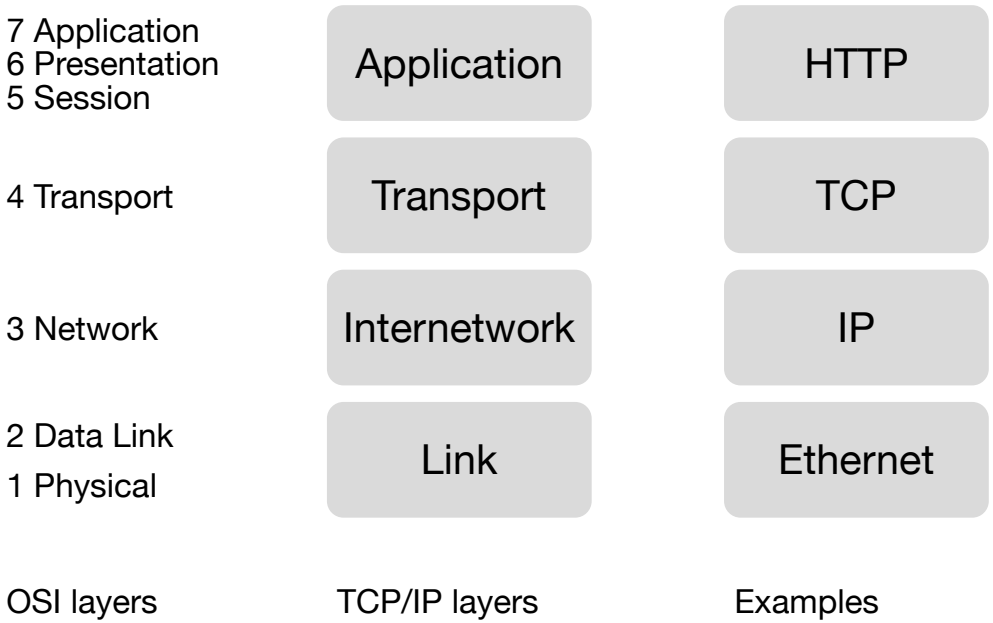
- A load balancer is a reverse proxy to distribute requests evenly across multiple servers.
- There are two common types of load balancers:
  - Layer 4 proxy
    - Intercepts TCP connections.
    - Works with many protocols that use TCP: HTTP, FTP, SIP, ...
  - Layer 7 proxy
    - Intercepts HTTP requests/responses.
    - Can be more intelligent because it knows more information.
      - For example it sees HTTP cookies.



Layered architectures for web applications

Reminder: Layered architectures

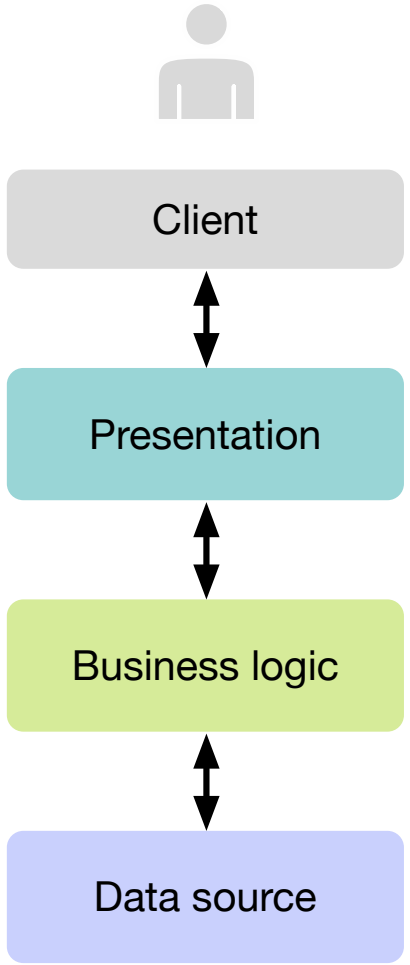
- A layered architecture helps to manage complexity:
  - The upper layers depend on the lower layers.
  - The lower layers **do not depend** on the upper layers.
- Example
  - Networking protocol layers



Layered architectures for web applications

Typical layers

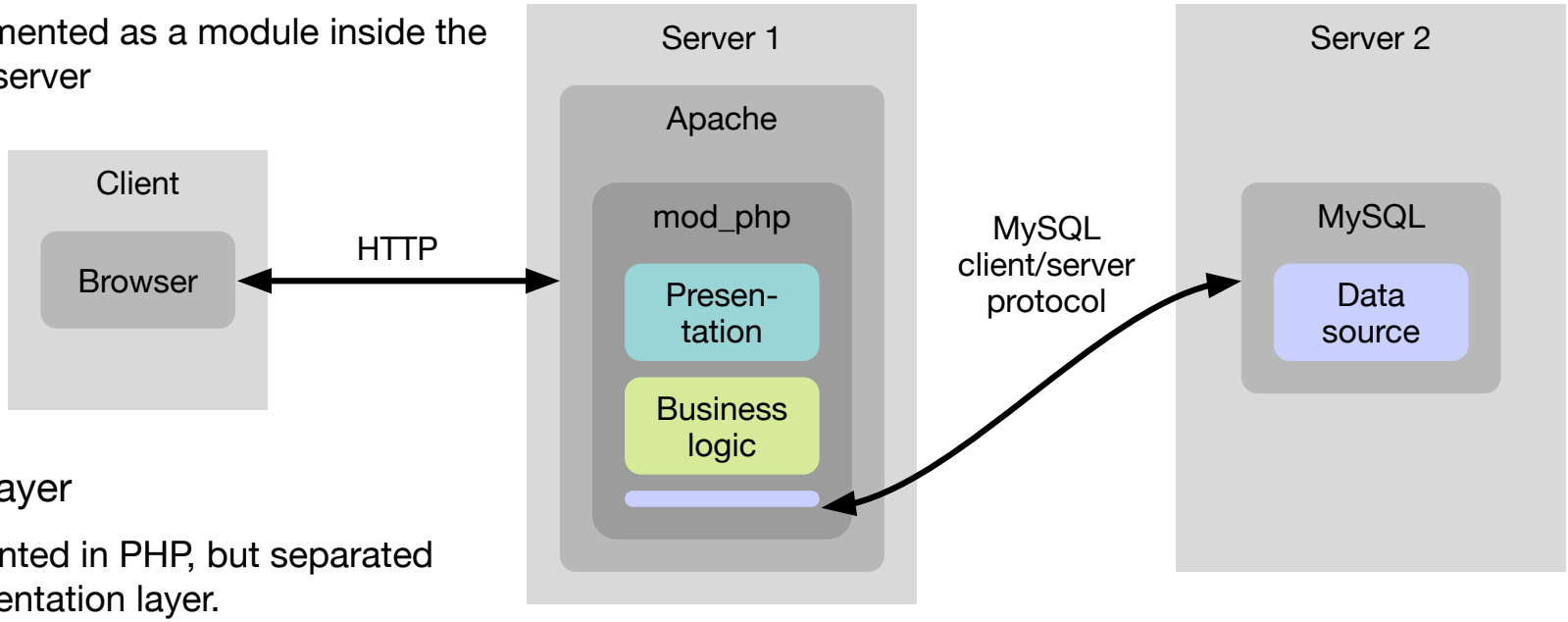
- The **presentation layer**
  - Manages the interaction between the user and the software.
  - Is written in HTML (+ JavaScript / AJAX)
- The **business logic** or **domain logic** layer
  - Makes computations, validations and management
  - Can encapsulate the data source, but this is normally not the case
- The **data source** layer
  - Communicates with other systems to receive and send data.
  - Typically includes a Database Management System (DBMS) for storing data persistently



Layered architectures for web applications

Example

- Example of a LAMP (Linux / Apache / MySQL / PHP) application
  - Presentation layer
    - Implemented in PHP
    - PHP is implemented as a module inside the Apache web server

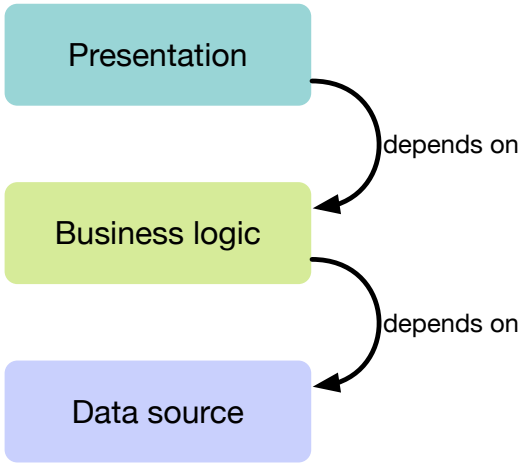


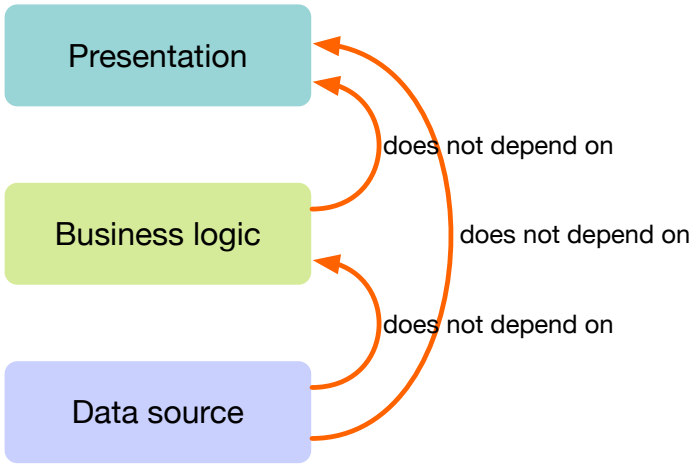
  - Business logic layer
    - Also implemented in PHP, but separated from the presentation layer.
  - Data source layer
    - Essentially implemented by the MySQL DBMS

Layered architectures for web applications

Typical layers

- The dependencies are important: for example the business logic layer and the data source layer should not have to know anything about the presentation layer.
- By maintaining this separation we can change our web interface by a new one whenever we want, without having to change the business logic or the data model.
- In terms of physical placement of layers:
  - All layers typically run on the server side.
  - With Ajax and related technologies a piece of the presentation layer may run on the client side.

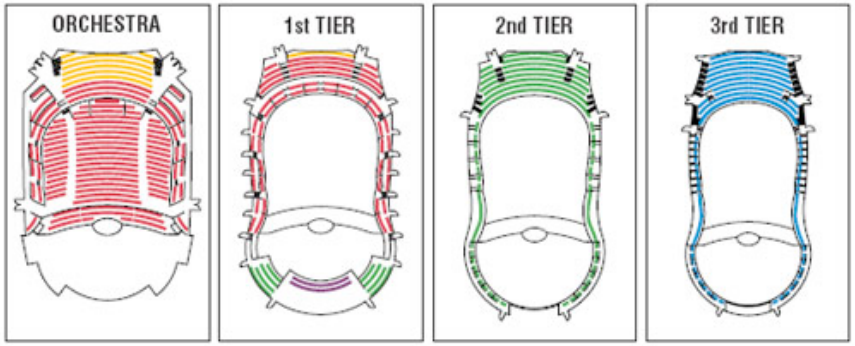


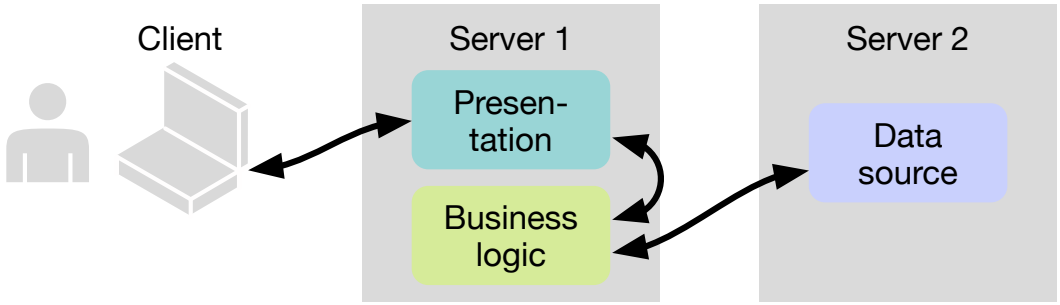


Layered architectures for web applications

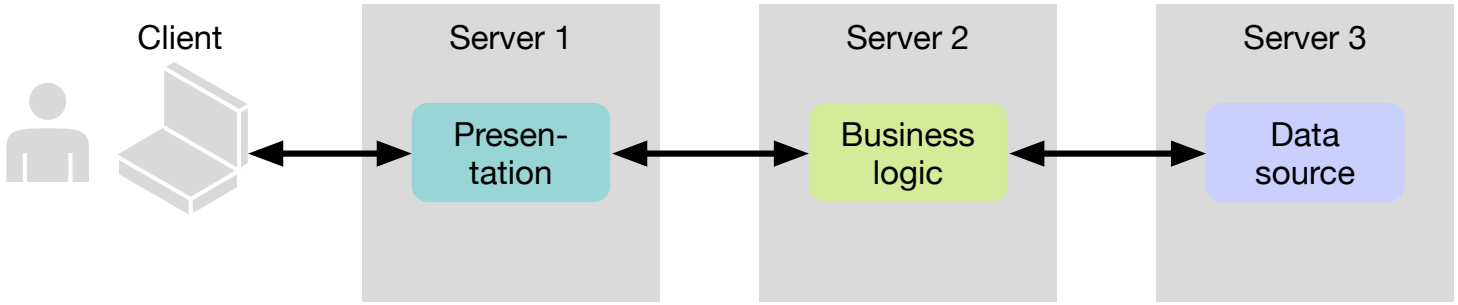
Distribution across servers

- Sometimes one uses the term *tier* instead of the term *layer*.
- The term tier generally implies a physical separation, while a layer is only logical.





Distribution over two tiers



Distribution over three tiers: **Three-Tier Architecture**

Scalability — Stateless modules

Introduction

- The HTTP protocol is an example of a *stateless* protocol.
  - This concept applies as well to the modules that constitute a web application.
- Definition
  - A web application module is *stateless* if the processing of an HTTP request from the client is independent from the preceding requests.
  - The module does not need to manage a session or conversation.
  - A module that manages state is called *stateful*.
- Advantages
  - Simplicity
  - Scalability
  - Allows to use caching
- In practice
  - Many simple web applications do not need state.
  - All applications that ask the user to log in need state.
- Sometimes one can make a module stateless by storing the state on the client side.
  - In cookies
  - In parameters of the URI
  - In hidden fields of a form
  - Doesn't work well for large data volumes.

Scalability

Parallel processing (scale-out)

- It is very easy to make a stateless module scalable.
  - Increase the number of servers.
  - Process requests in parallel.
  - Distribute the load.
- Example: Scale-out of the presentation layer.