Network Programming

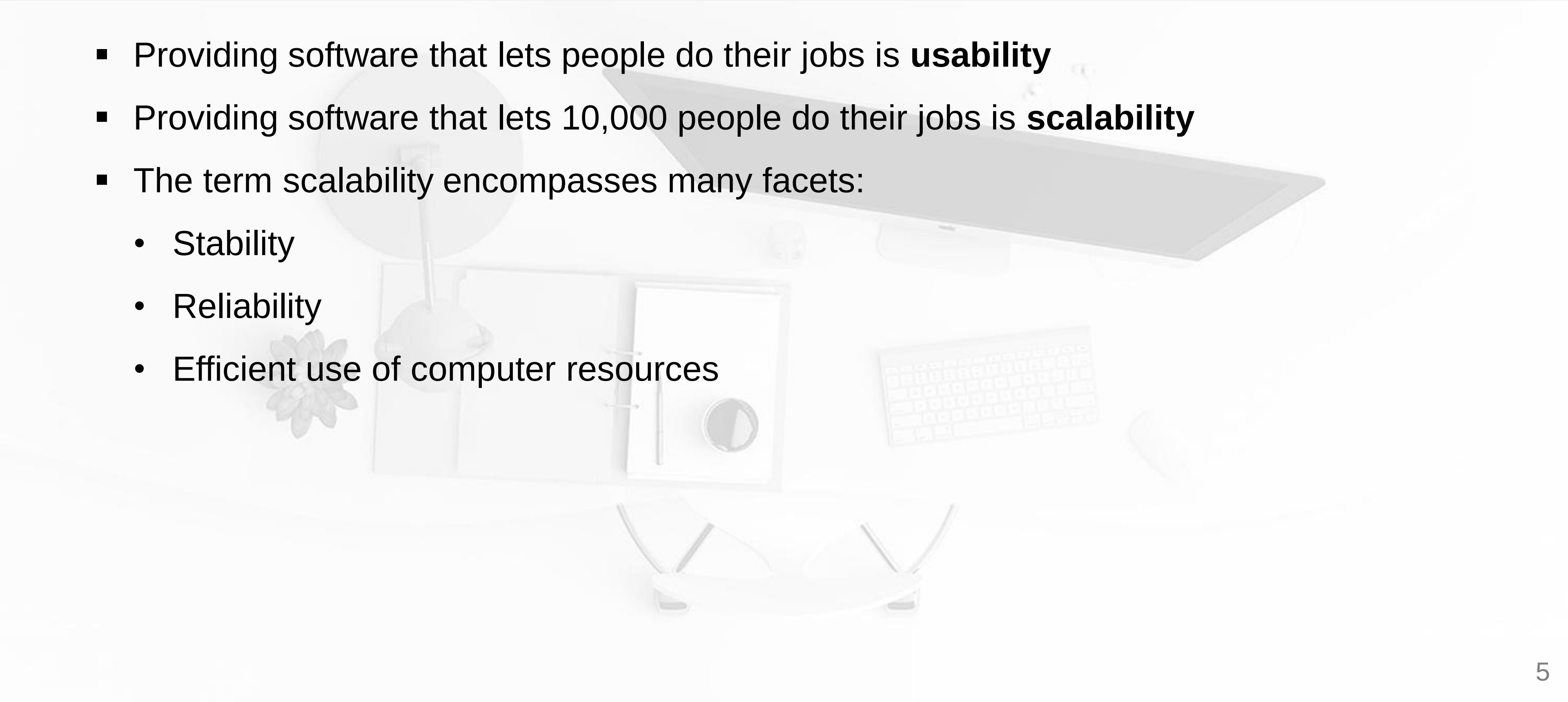
Ung Văn Giàu
Email: giau.ung@eiu.edu.vn

Programming for Scalability

Content

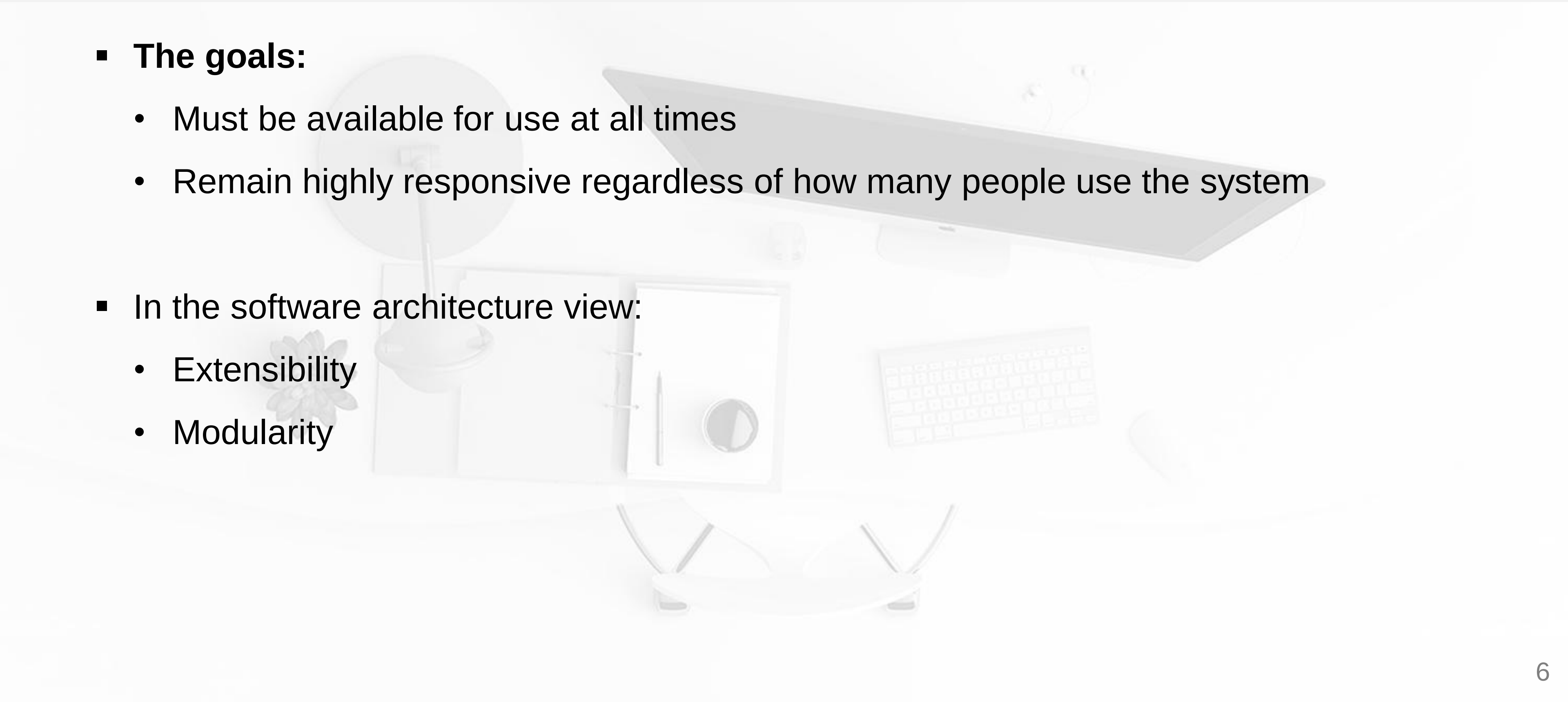
1. Scalability

Scalability



- Providing software that lets people do their jobs is **usability**
- Providing software that lets 10,000 people do their jobs is **scalability**
- The term scalability encompasses many facets:
  - Stability
  - Reliability
  - Efficient use of computer resources

Scalability



- **The goals:**
  - Must be available for use at all times
  - Remain highly responsive regardless of how many people use the system

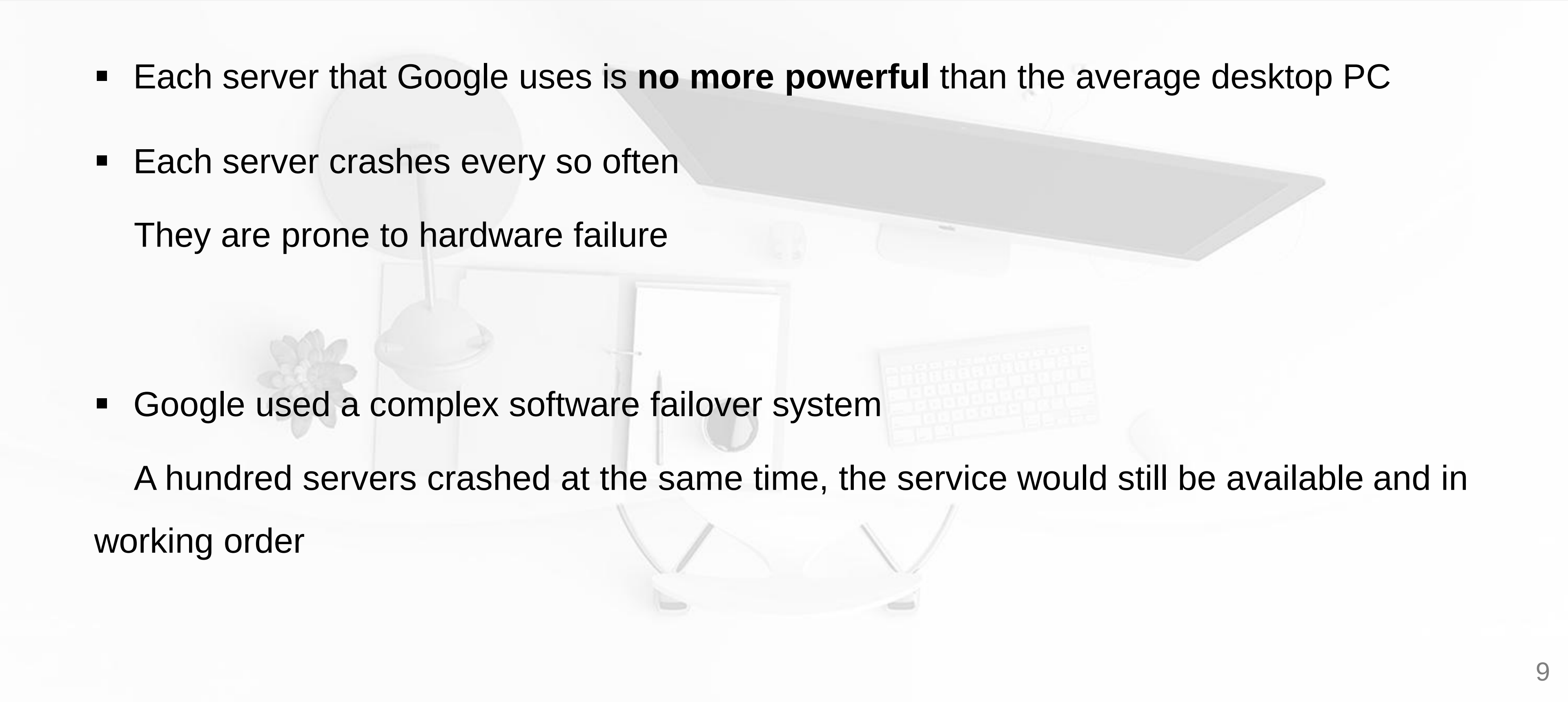
- In the software architecture view:
  - Extensibility
  - Modularity

2. Google search engine

Google search engine

- Google is certainly the Internet's largest search engine
- It serves more than 3 billion request per day
- It had more than 2,5 million servers distributed worldwide (June 2016)
- One of the most scalable Internet services

Google search engine



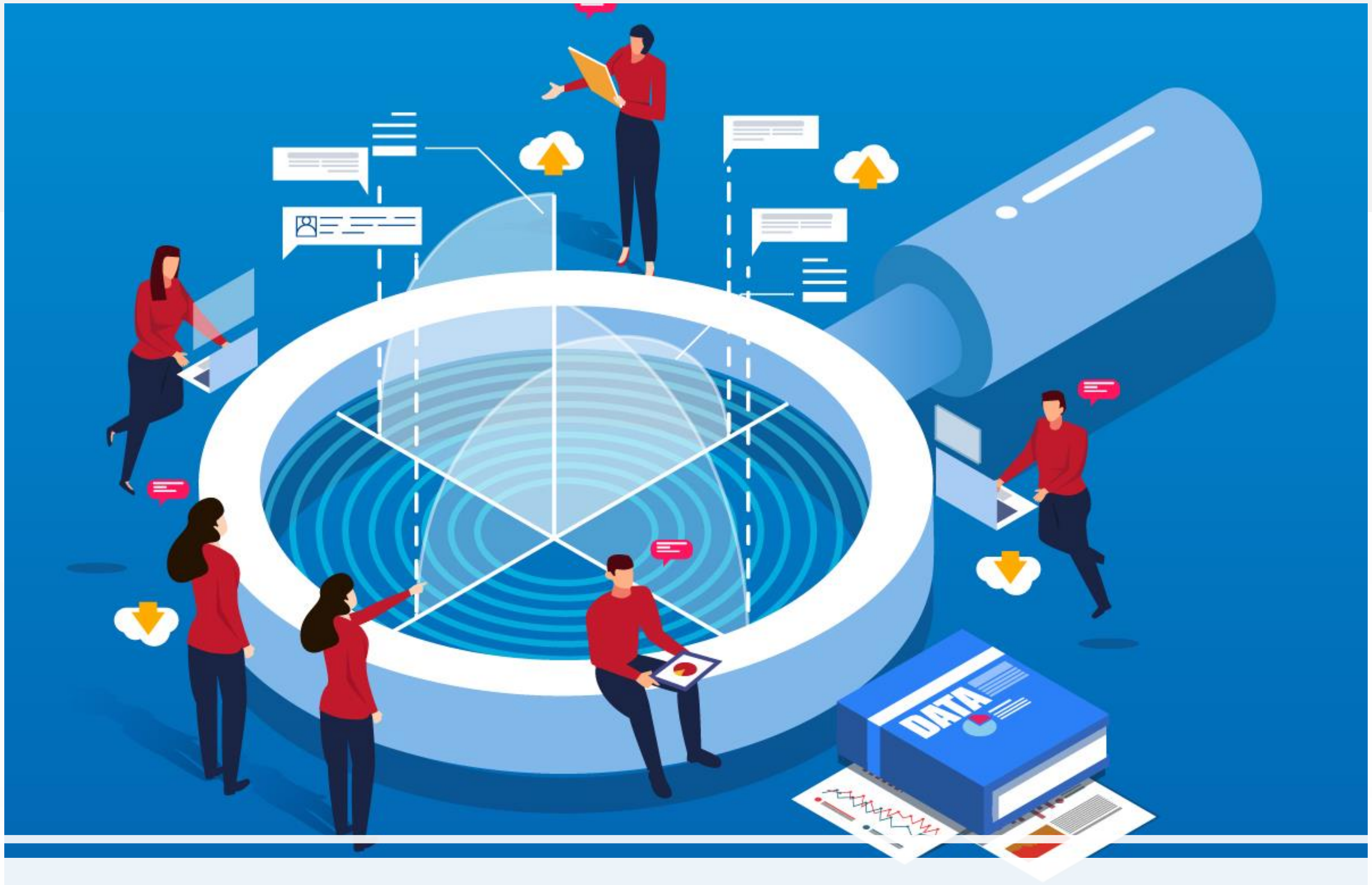
- Each server that Google uses is **no more powerful** than the average desktop PC

- Each server crashes every so often

  They are prone to hardware failure

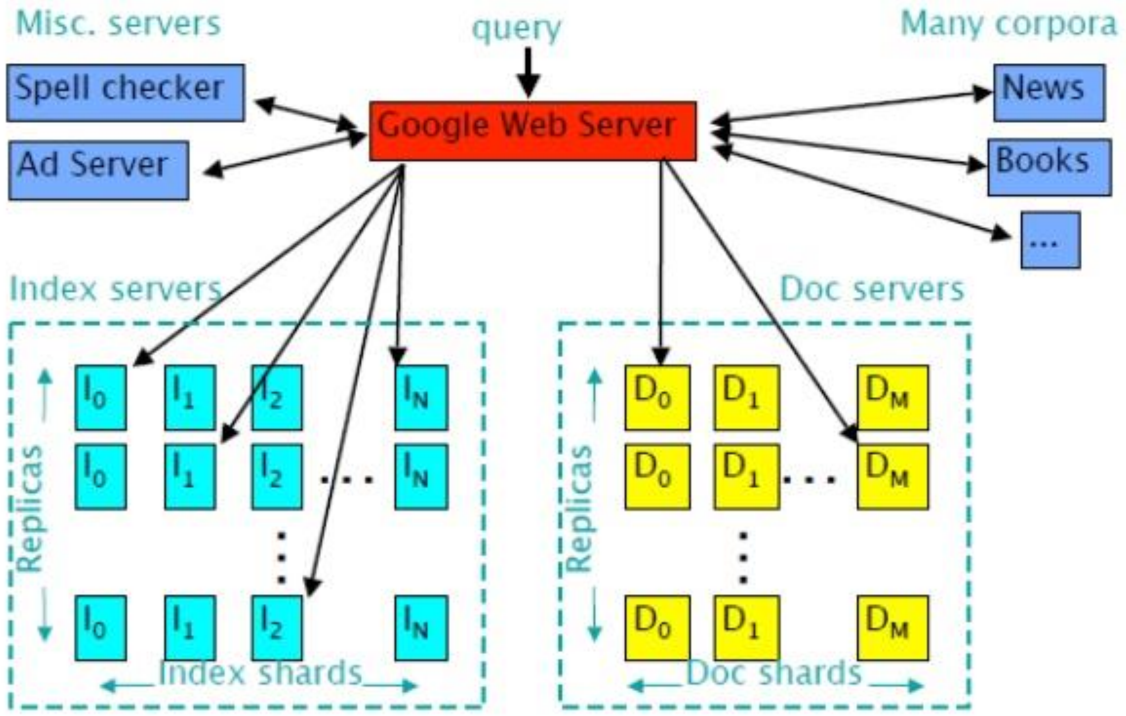
- Google used a complex software failover system

  A hundred servers crashed at the same time, the service would still be available and in working order



Big data

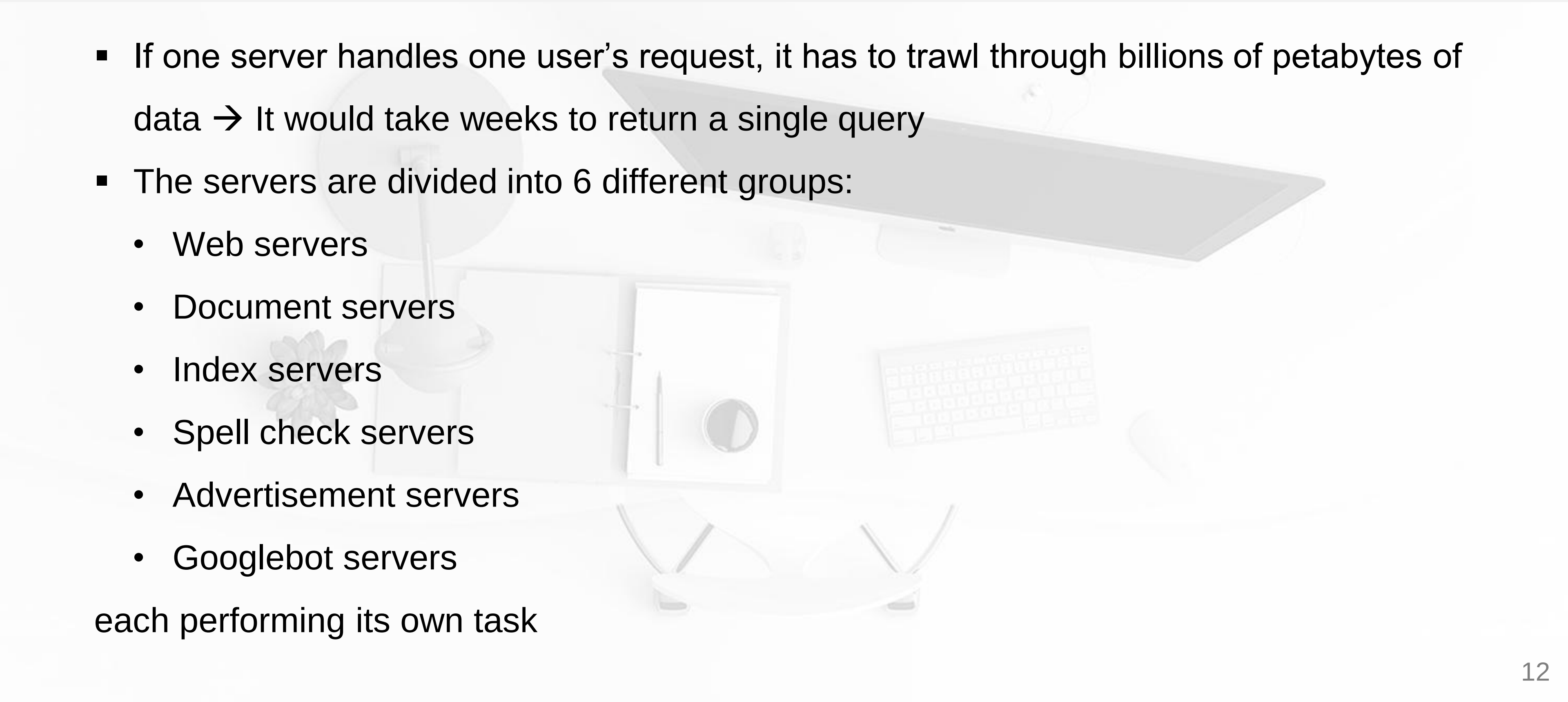
Google Query Serving Infrastructure



Elapsed time: 0.25s, machines involved: 1000s+

3

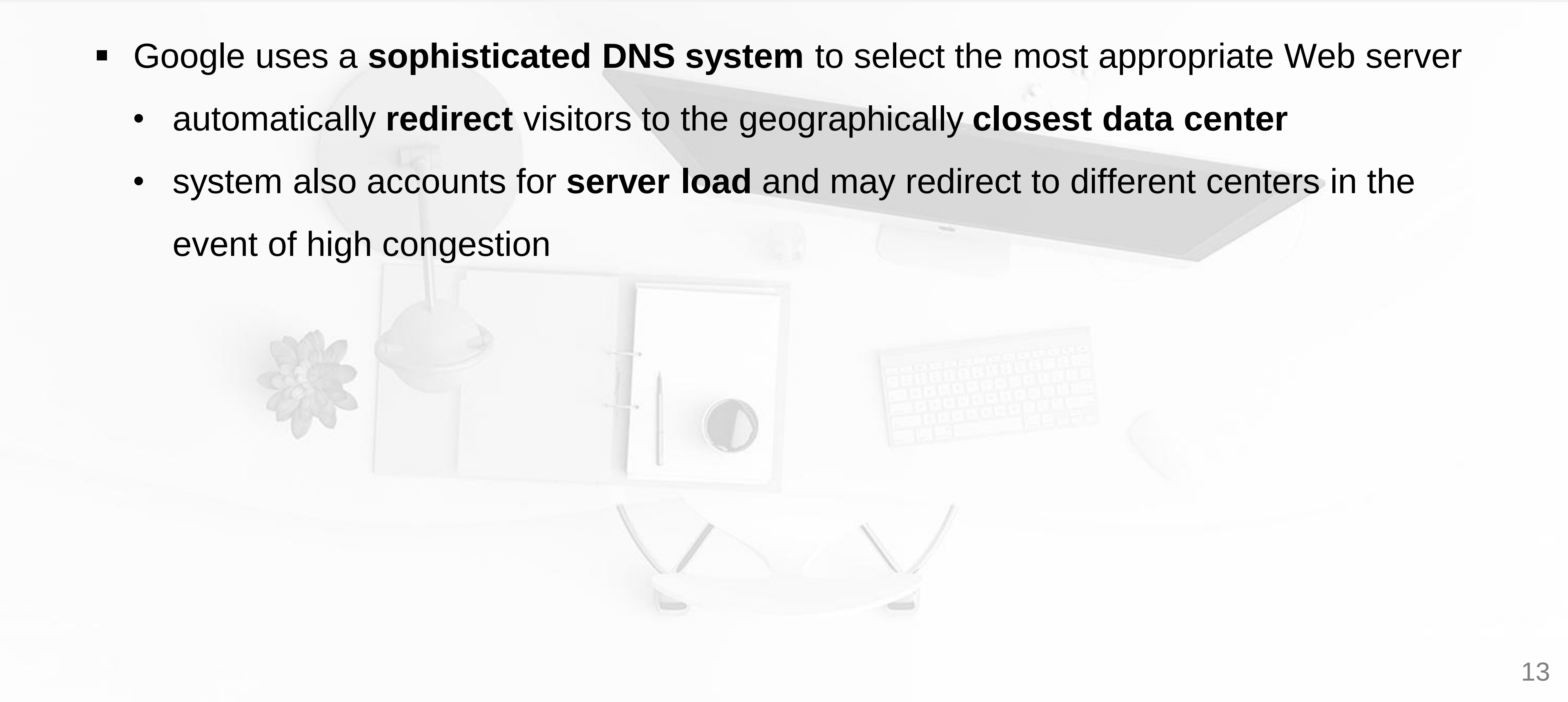
Google search engine



- If one server handles one user’s request, it has to trawl through billions of petabytes of data → It would take weeks to return a single query
- The servers are divided into 6 different groups:
  - Web servers
  - Document servers
  - Index servers
  - Spell check servers
  - Advertisement servers
  - Googlebot servers

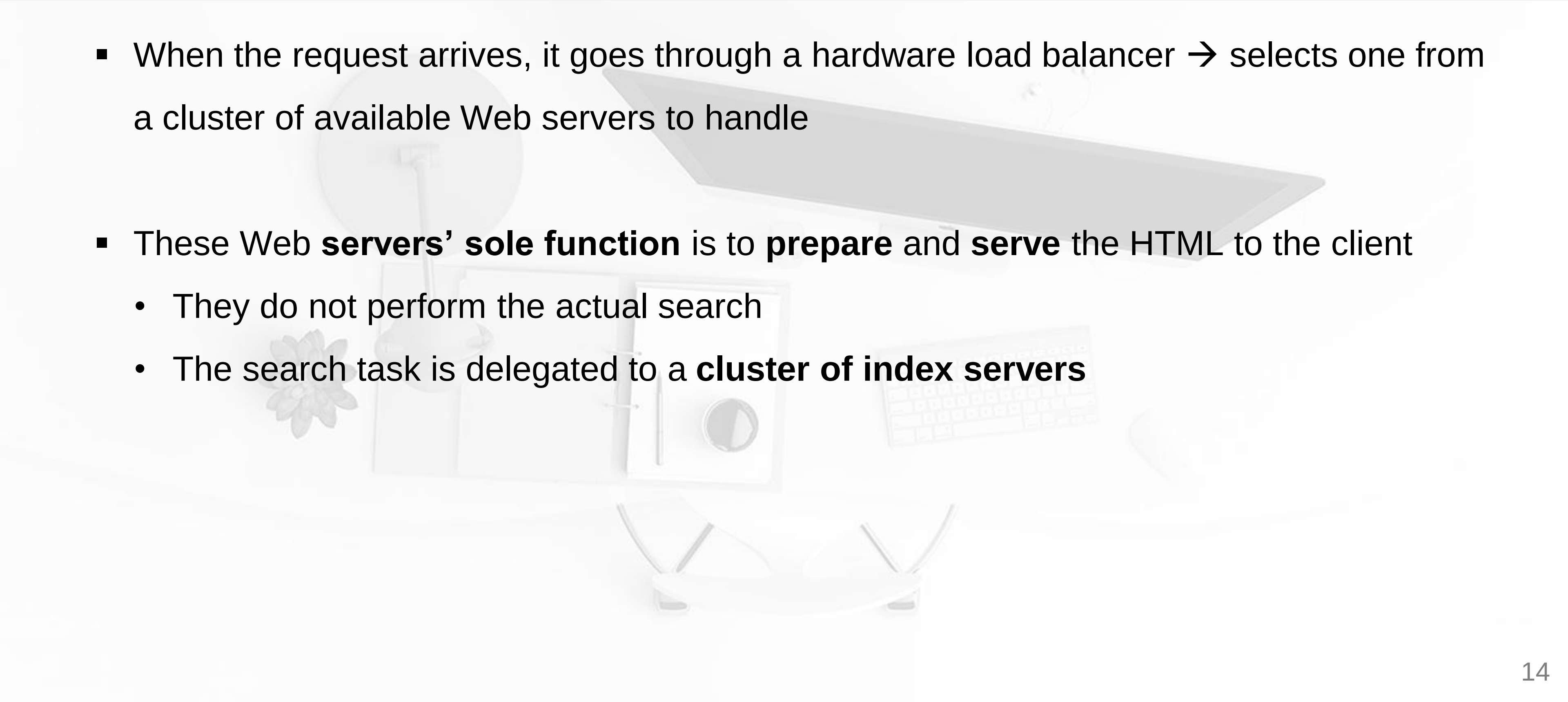
each performing its own task

Google search engine



- Google uses a **sophisticated DNS system** to select the most appropriate Web server
  - automatically **redirect** visitors to the geographically **closest data center**
  - system also accounts for **server load** and may redirect to different centers in the event of high congestion

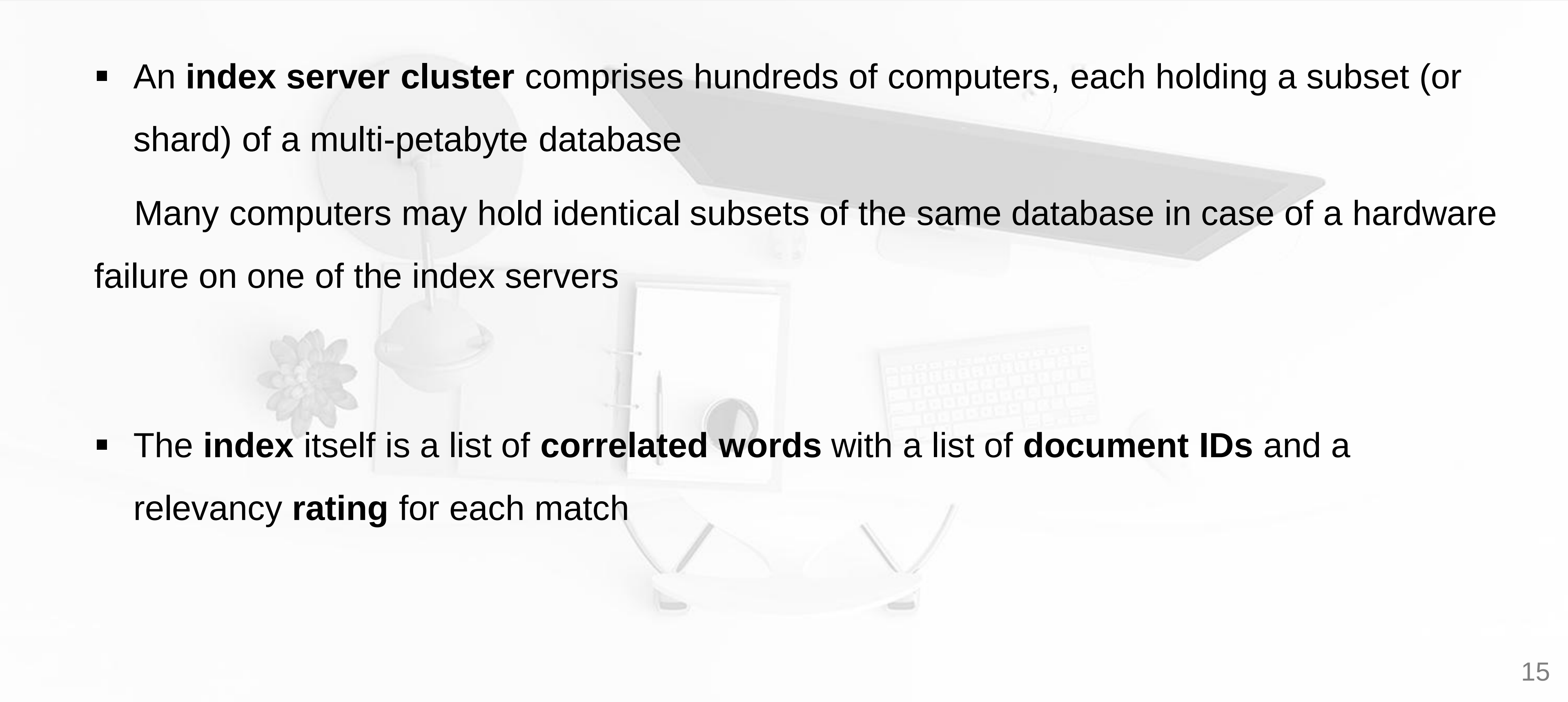
Google search engine



- When the request arrives, it goes through a hardware load balancer → selects one from a cluster of available Web servers to handle

- These Web **servers' sole function** is to **prepare** and **serve** the HTML to the client
  - They do not perform the actual search
  - The search task is delegated to a **cluster of index servers**

Google search engine

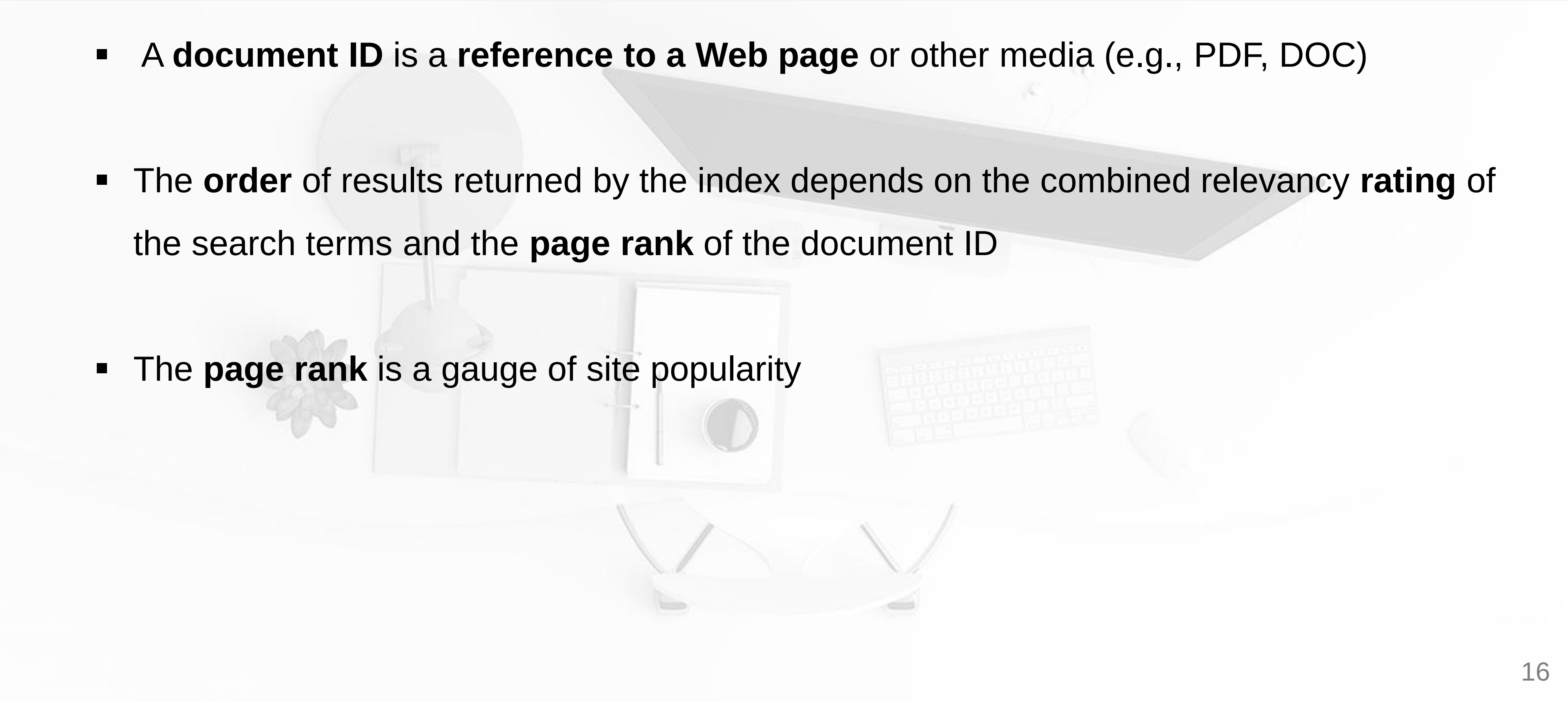


- An **index server cluster** comprises hundreds of computers, each holding a subset (or shard) of a multi-petabyte database

  Many computers may hold identical subsets of the same database in case of a hardware failure on one of the index servers

- The **index** itself is a list of **correlated words** with a list of **document IDs** and a relevancy **rating** for each match

Google search engine



- A **document ID** is a **reference to a Web page** or other media (e.g., PDF, DOC)

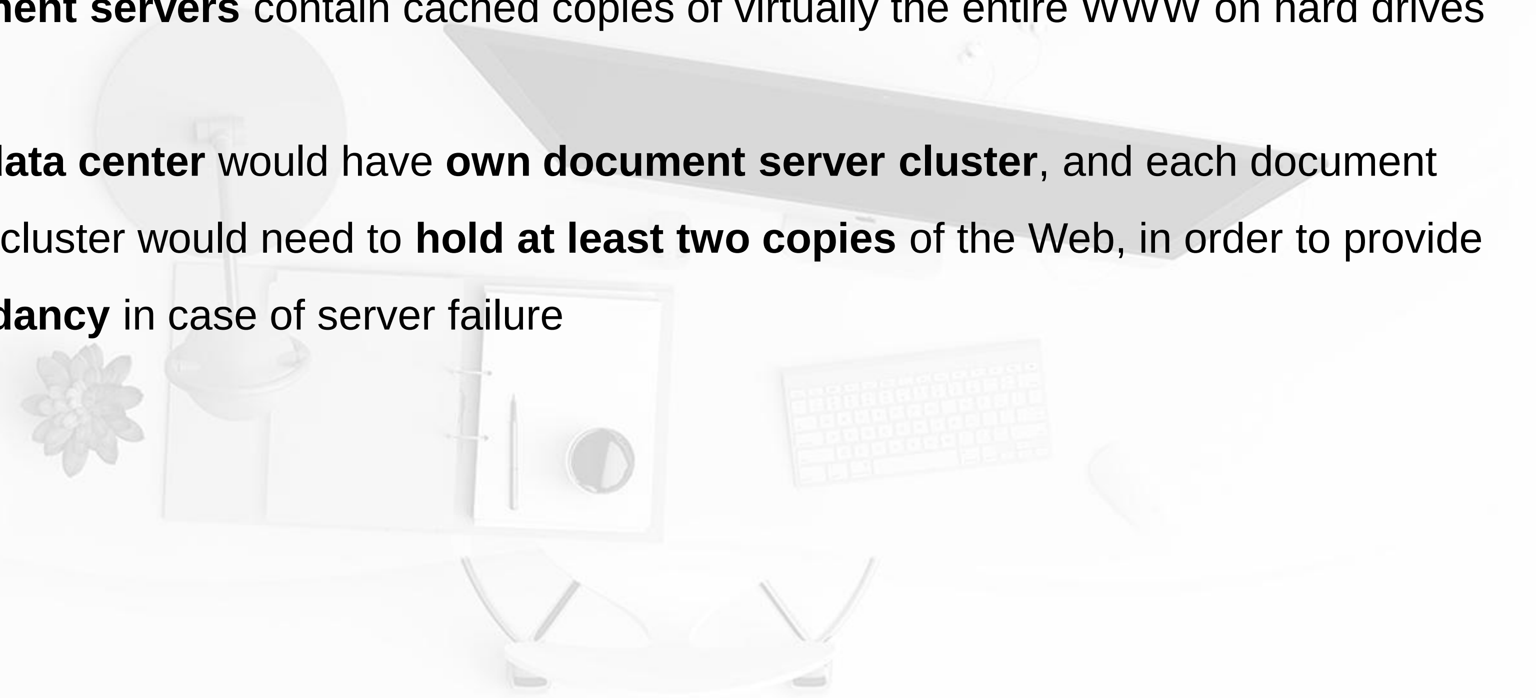
- The **order** of results returned by the index depends on the combined relevancy **rating** of the search terms and the **page rank** of the document ID

- The **page rank** is a gauge of site popularity

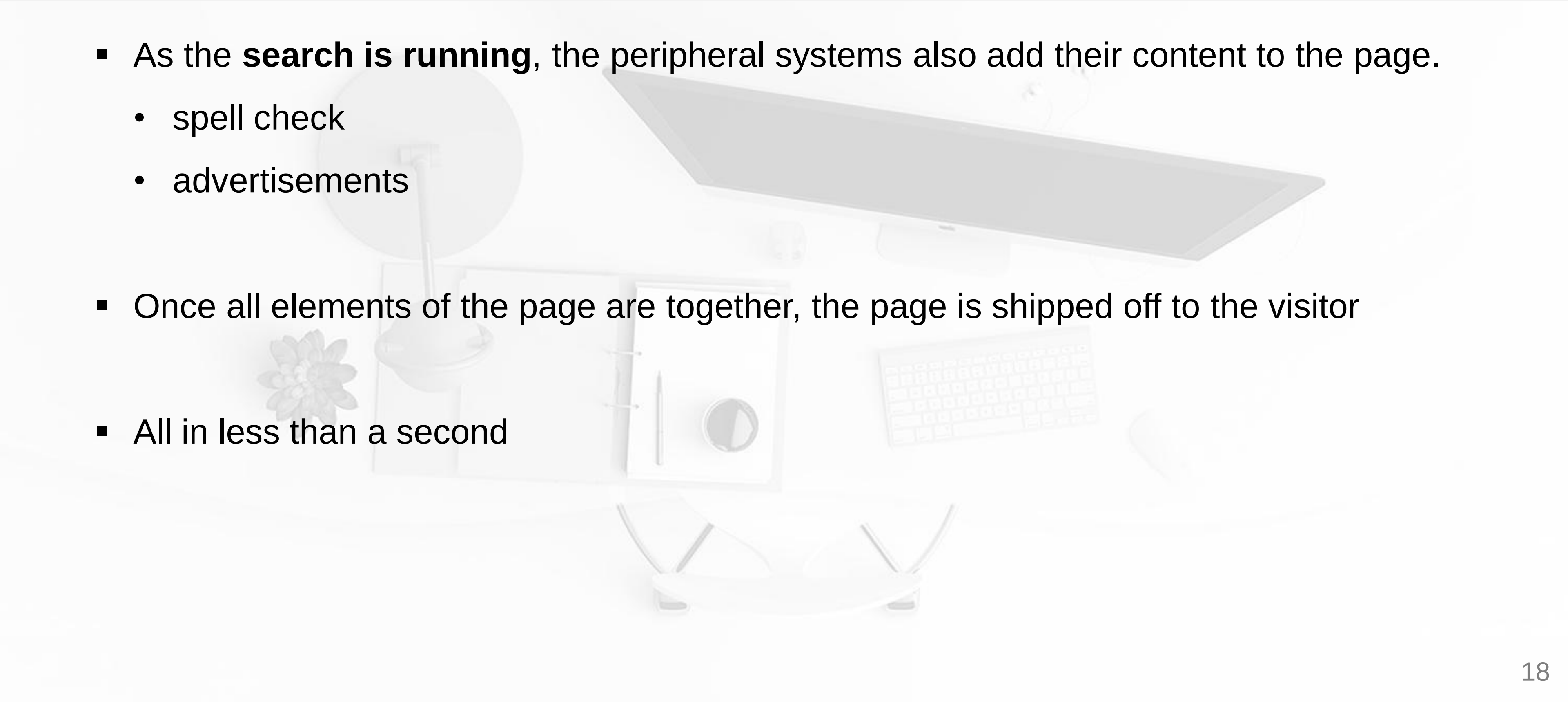
Google search engine

- **Document servers** contain cached copies of virtually the entire WWW on hard drives

- Each **data center** would have **own document server cluster**, and each document server cluster would need to **hold at least two copies** of the Web, in order to provide **redundancy** in case of server failure



Google search engine



- As the **search is running**, the peripheral systems also add their content to the page.
  - spell check
  - advertisements

- Once all elements of the page are together, the page is shipped off to the visitor

- All in less than a second

Google search engine

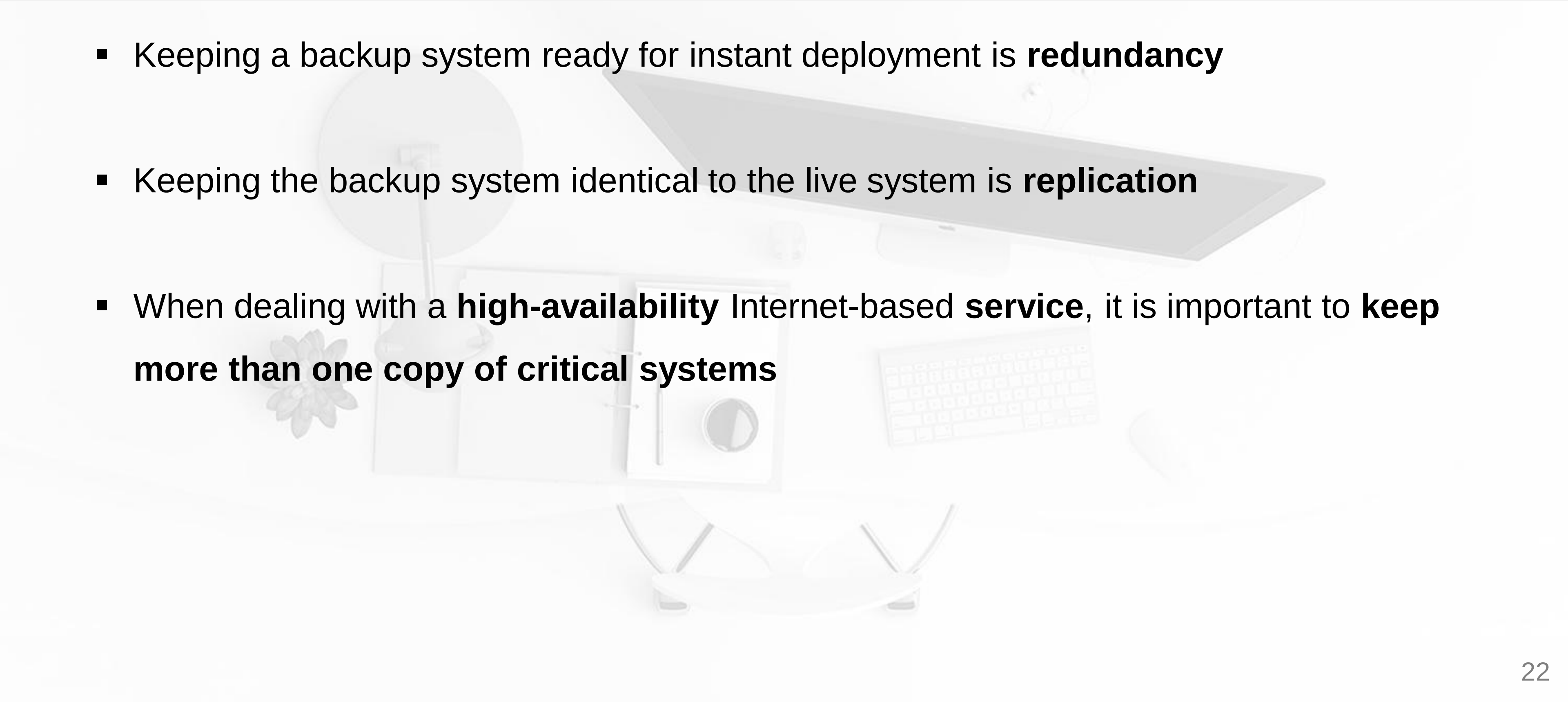
- Google bot (spider) is software, running on thousands of PCs simultaneously, and trawls the Web continuously

- Google bot stores the content in the document servers and updates the index servers

The Google architecture is one of the best in the world and is the pinnacle of scalability

3. Replication & redundancy

Replication & redundancy



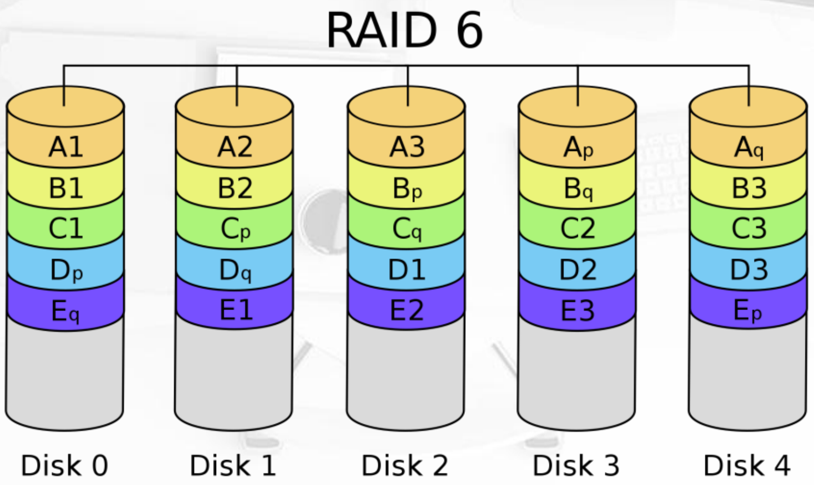
- Keeping a backup system ready for instant deployment is **redundancy**
- Keeping the backup system identical to the live system is **replication**
- When dealing with a **high-availability** Internet-based **service**, it is important to **keep more than one copy of critical systems**

Replication & redundancy

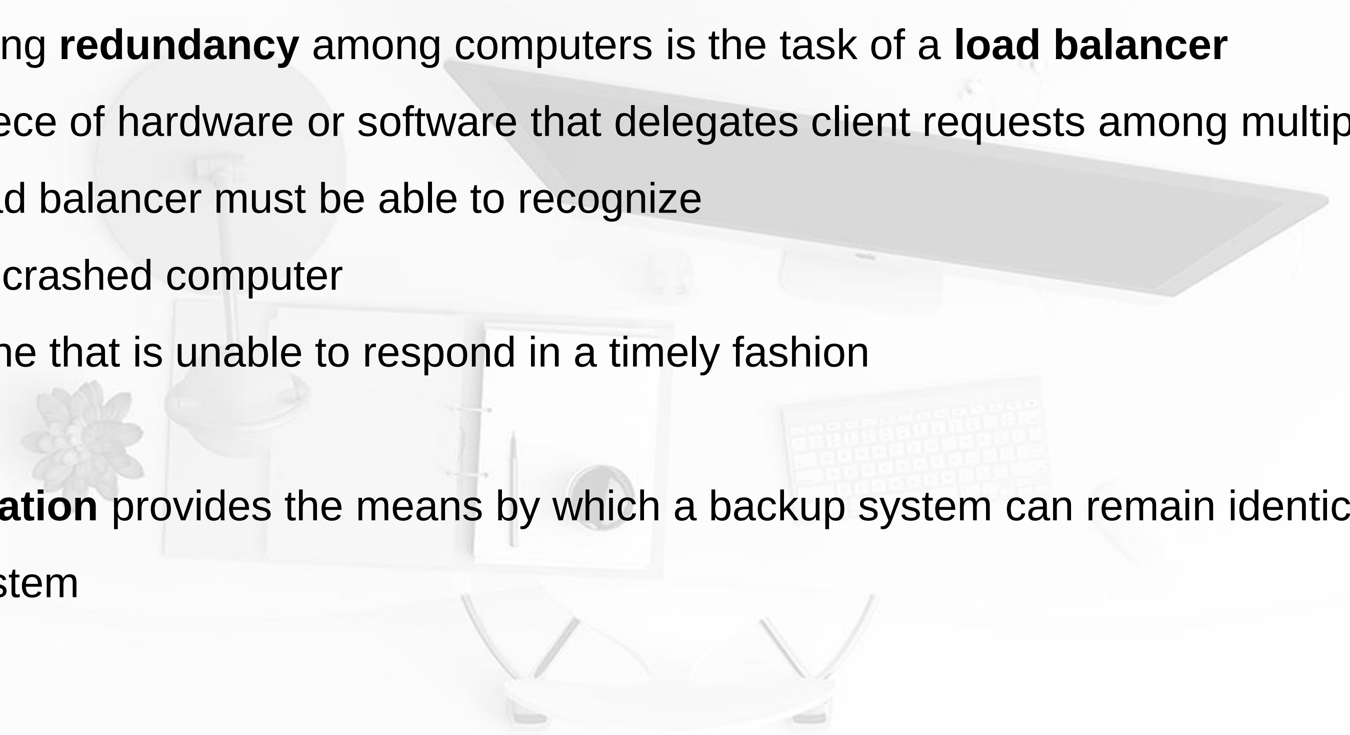
- Backup systems do not need to be kept on separate machines

  Use a redundant array of inexpensive disks (RAID) array.

  Many computers can read from a RAID array at once but only one computer can write at the same time.



Replication & redundancy



- Providing **redundancy** among computers is the task of a **load balancer**

    A piece of hardware or software that delegates client requests among multiple servers

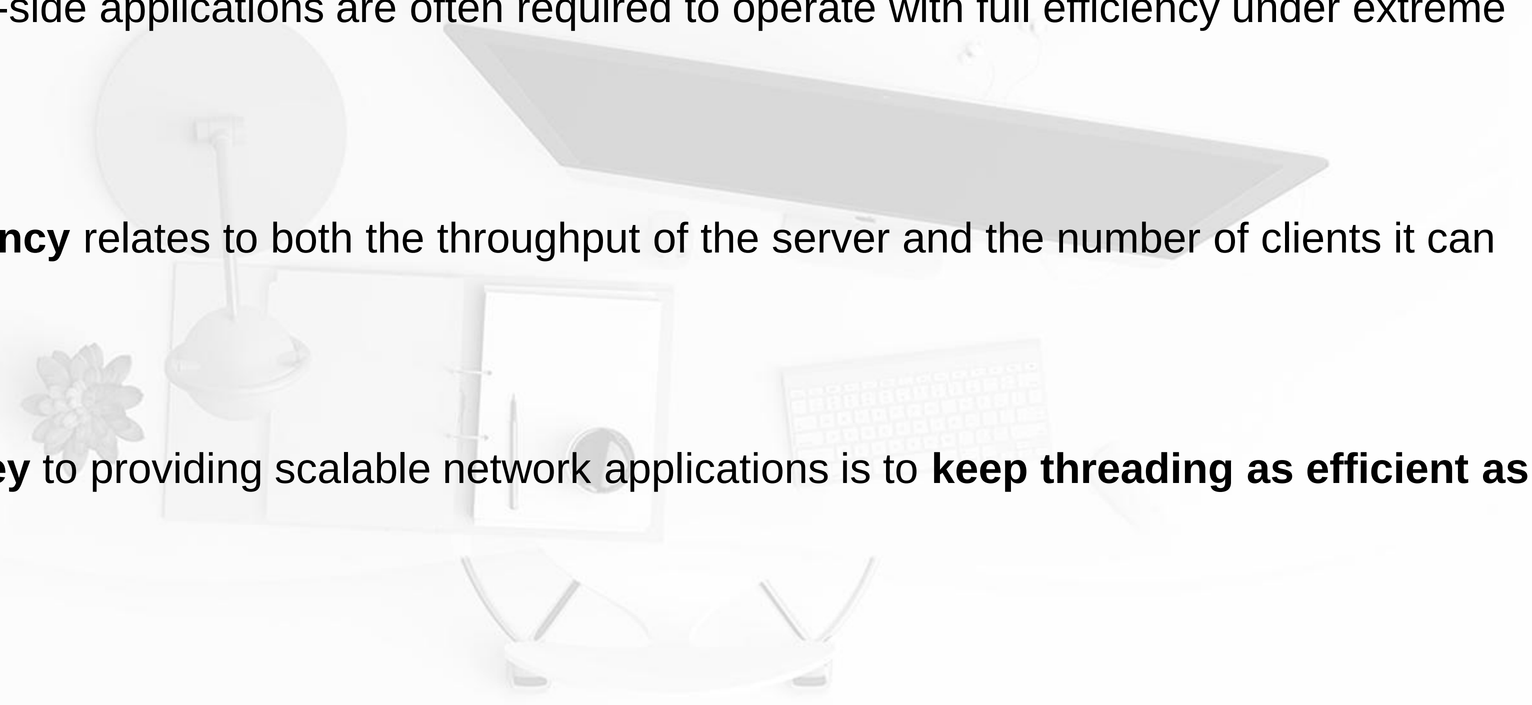
→ the load balancer must be able to recognize

  - ✓ a crashed computer
  - ✓ one that is unable to respond in a timely fashion

- **Replication** provides the means by which a backup system can remain identical to the live system

4. Scalable network applications

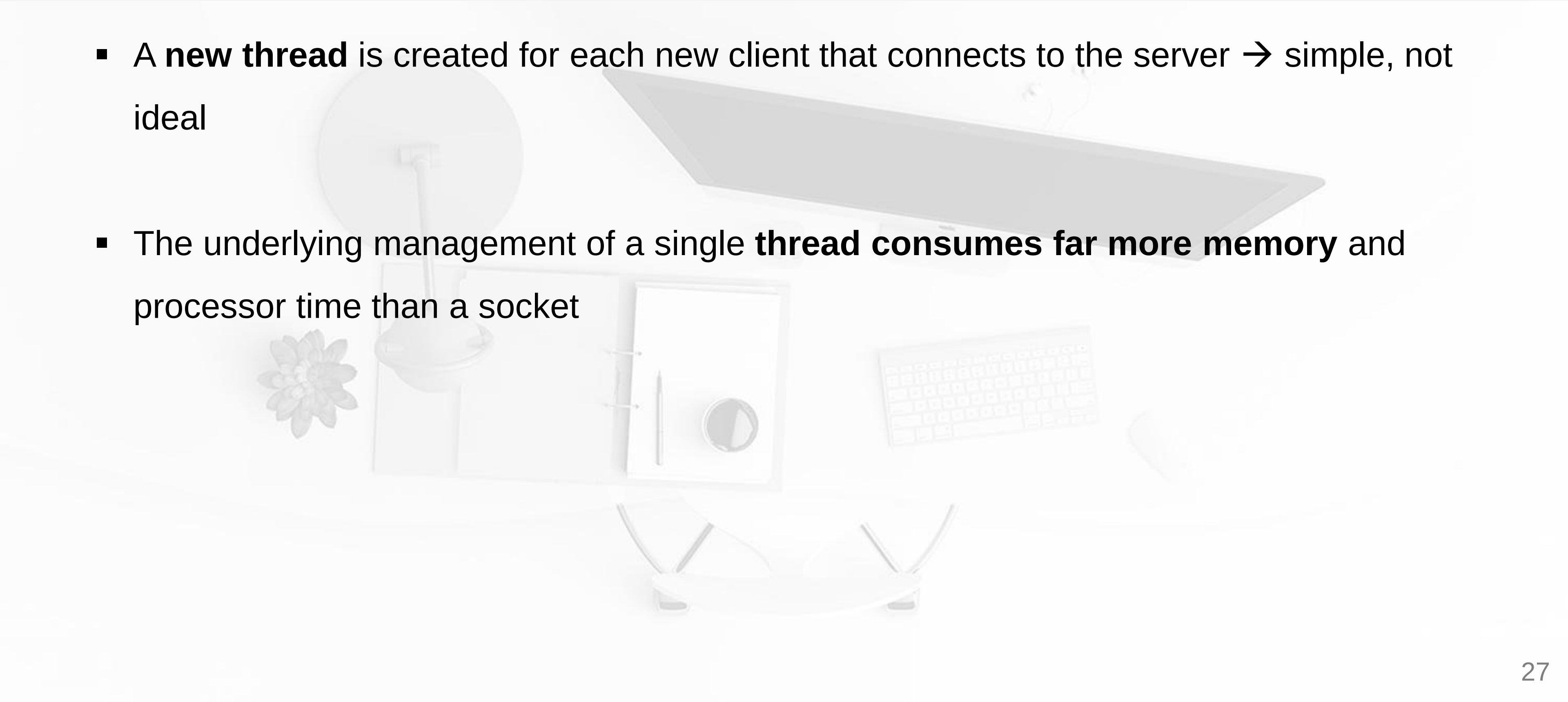
Scalable network applications



- Server-side applications are often required to operate with full efficiency under extreme load
- **Efficiency** relates to both the throughput of the server and the number of clients it can handle

→ The **key** to providing scalable network applications is to **keep threading as efficient as possible**

Scalable network applications



- A **new thread** is created for each new client that connects to the server → simple, not ideal

- The underlying management of a single **thread consumes far more memory** and processor time than a socket

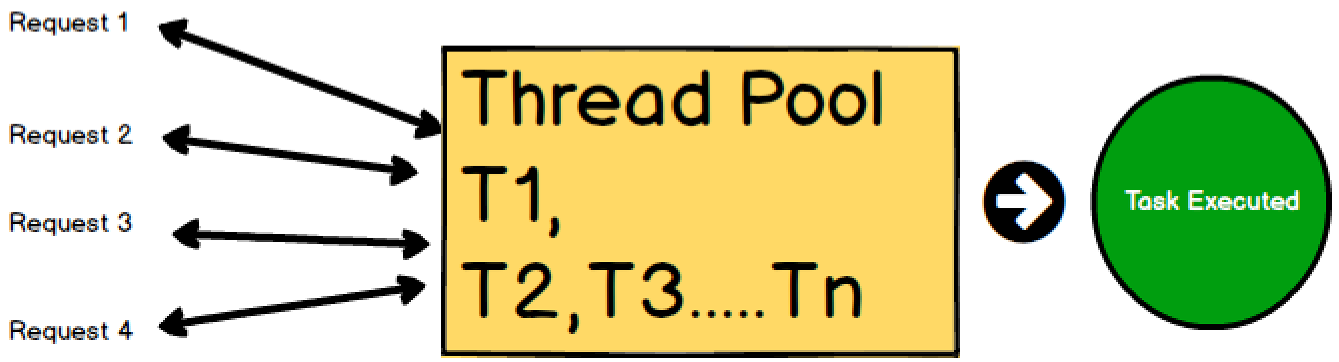
Scalable network applications

To better manage thread creation, a technique known as **thread pooling** can be employed

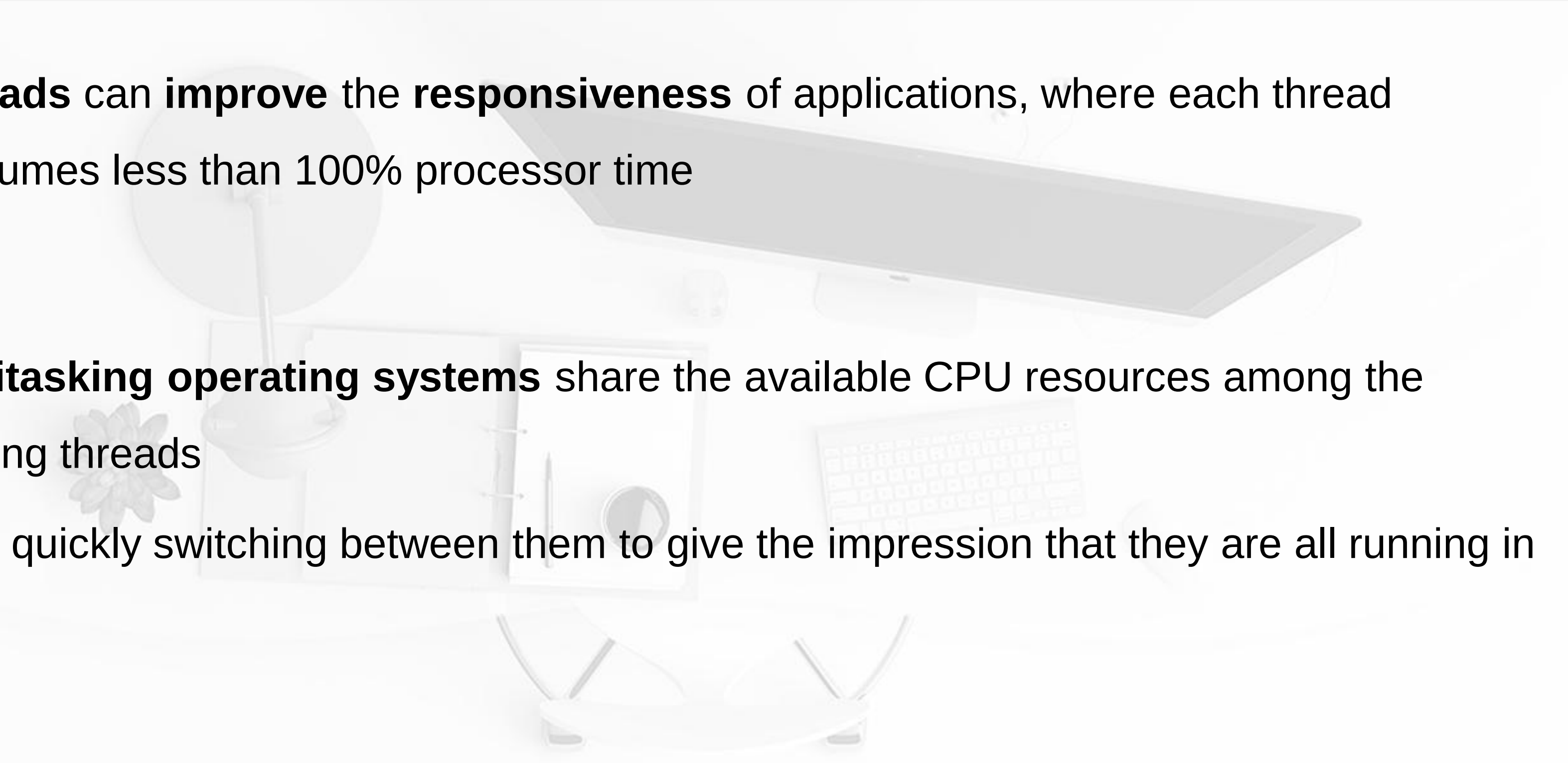
5. Thread pooling

Thread pooling

Every computer has a **limit to the number of threads** it can process at one time depending on the resources consumed by each thread (quite low number)



Thread pooling

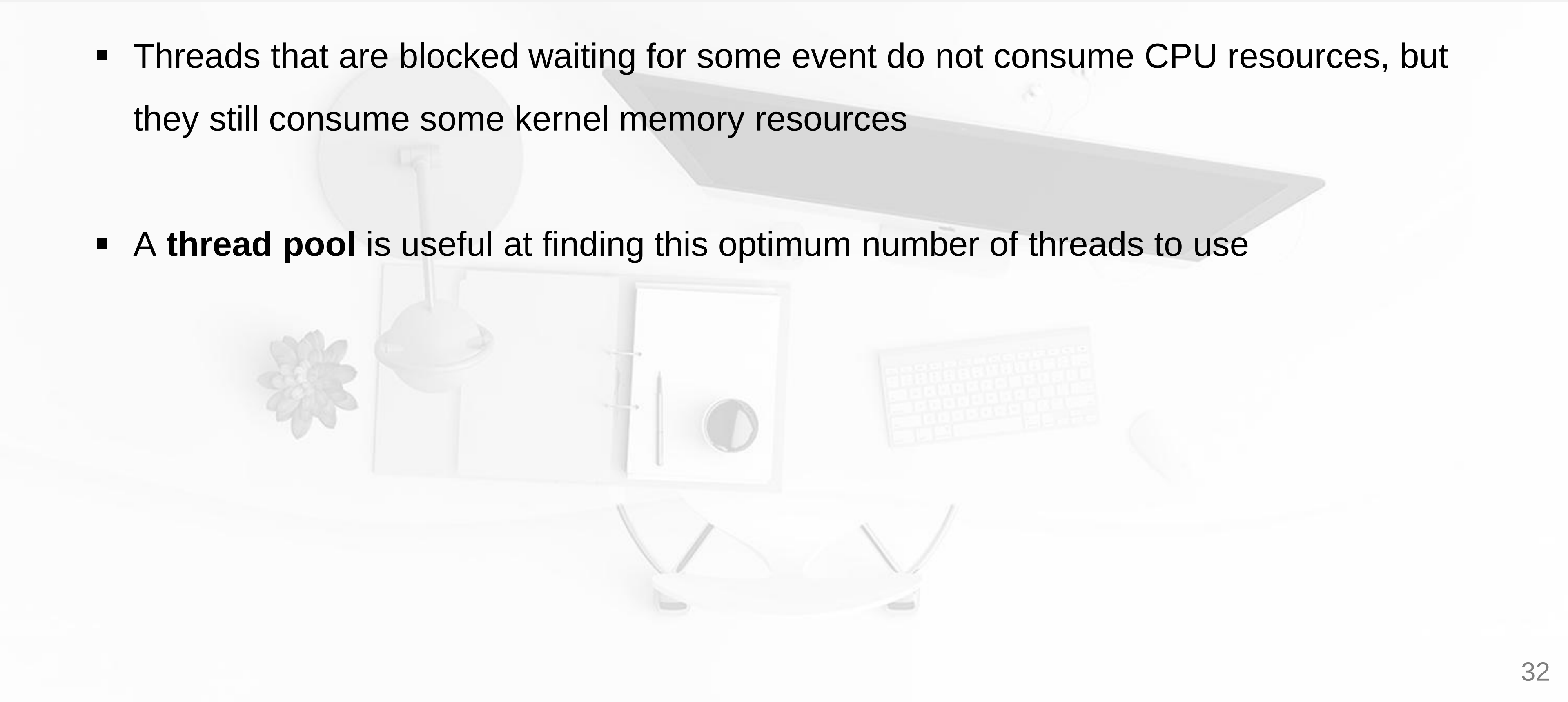


- **Threads** can **improve** the **responsiveness** of applications, where each thread consumes less than 100% processor time

- **Multitasking operating systems** share the available CPU resources among the running threads

    by quickly switching between them to give the impression that they are all running in parallel

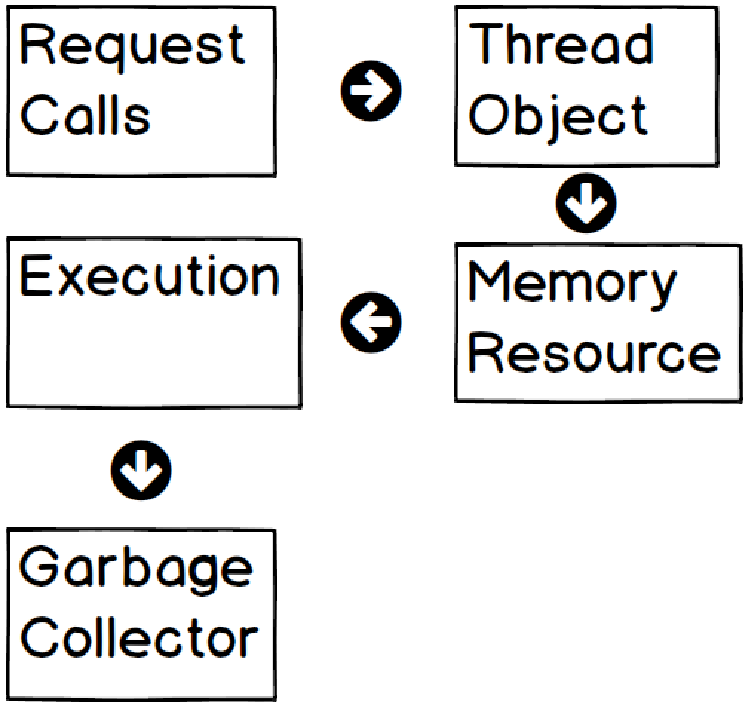
Thread pooling



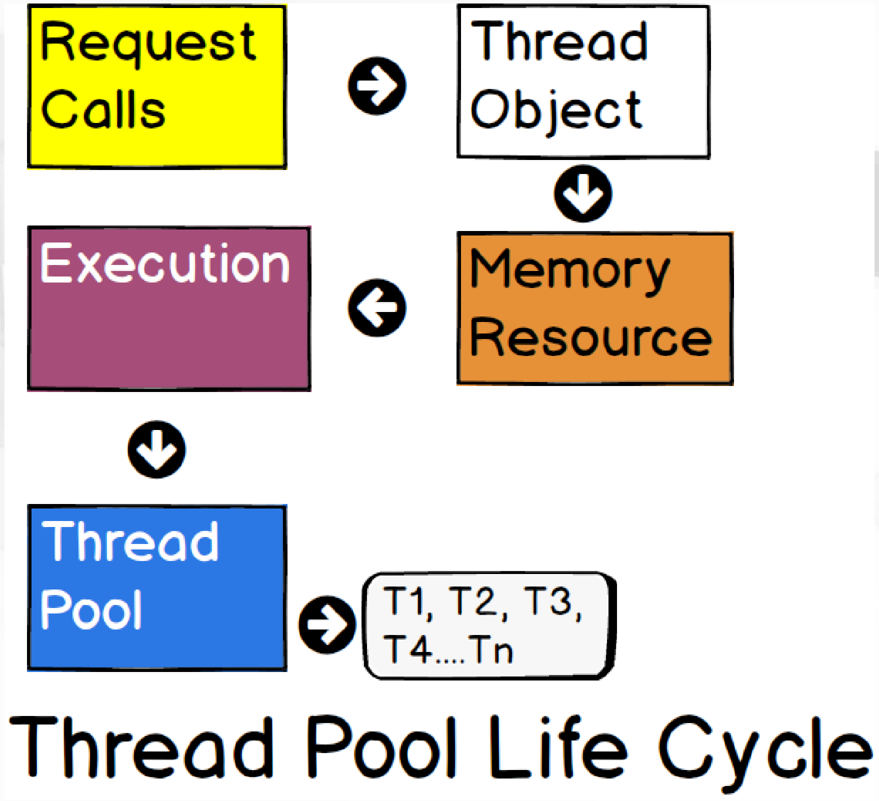
- Threads that are blocked waiting for some event do not consume CPU resources, but they still consume some kernel memory resources

- A **thread pool** is useful at finding this optimum number of threads to use

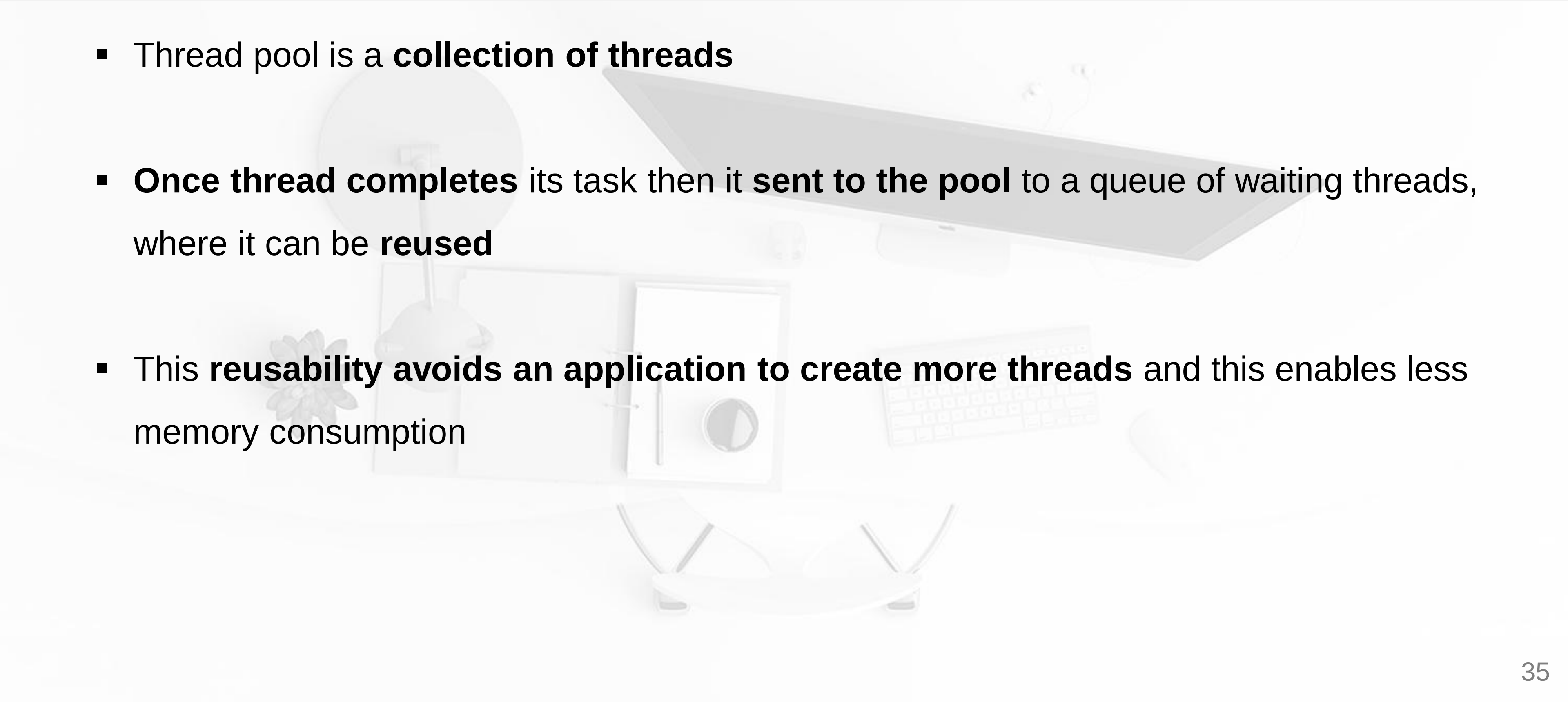
Thread Lifecycle



Thread pooling

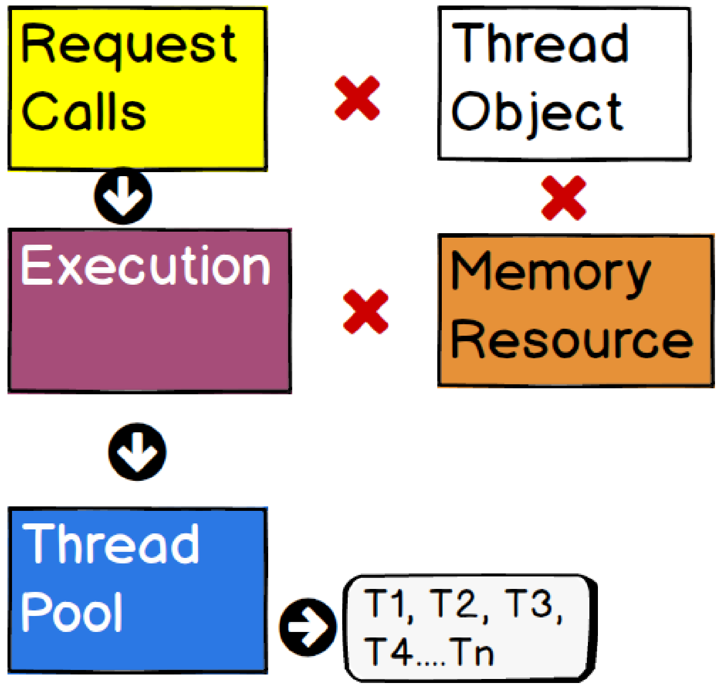


Thread pooling



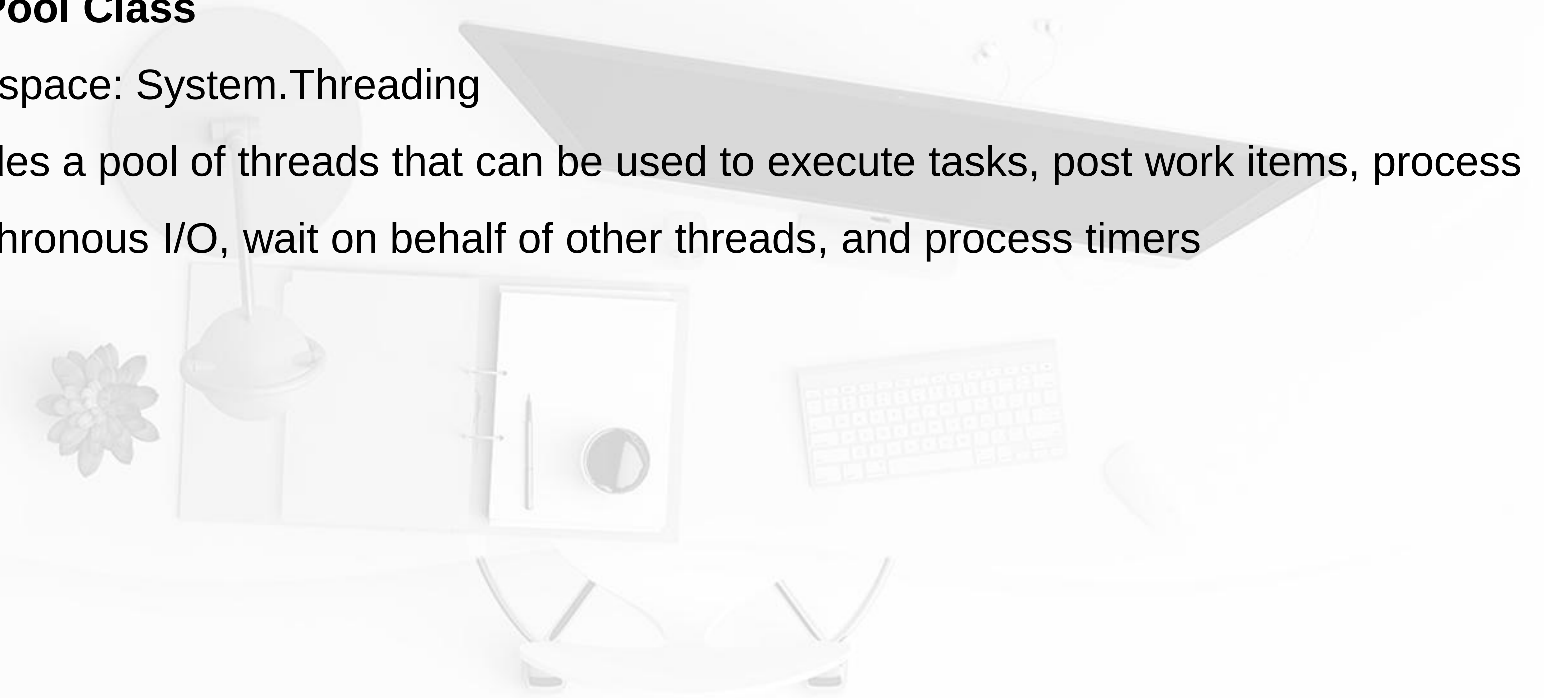
- Thread pool is a **collection of threads**
- **Once thread completes** its task then it **sent to the pool** to a queue of waiting threads, where it can be **reused**
- This **reusability avoids an application to create more threads** and this enables less memory consumption

Thread pooling



Thread Pool Life Cycle

Library

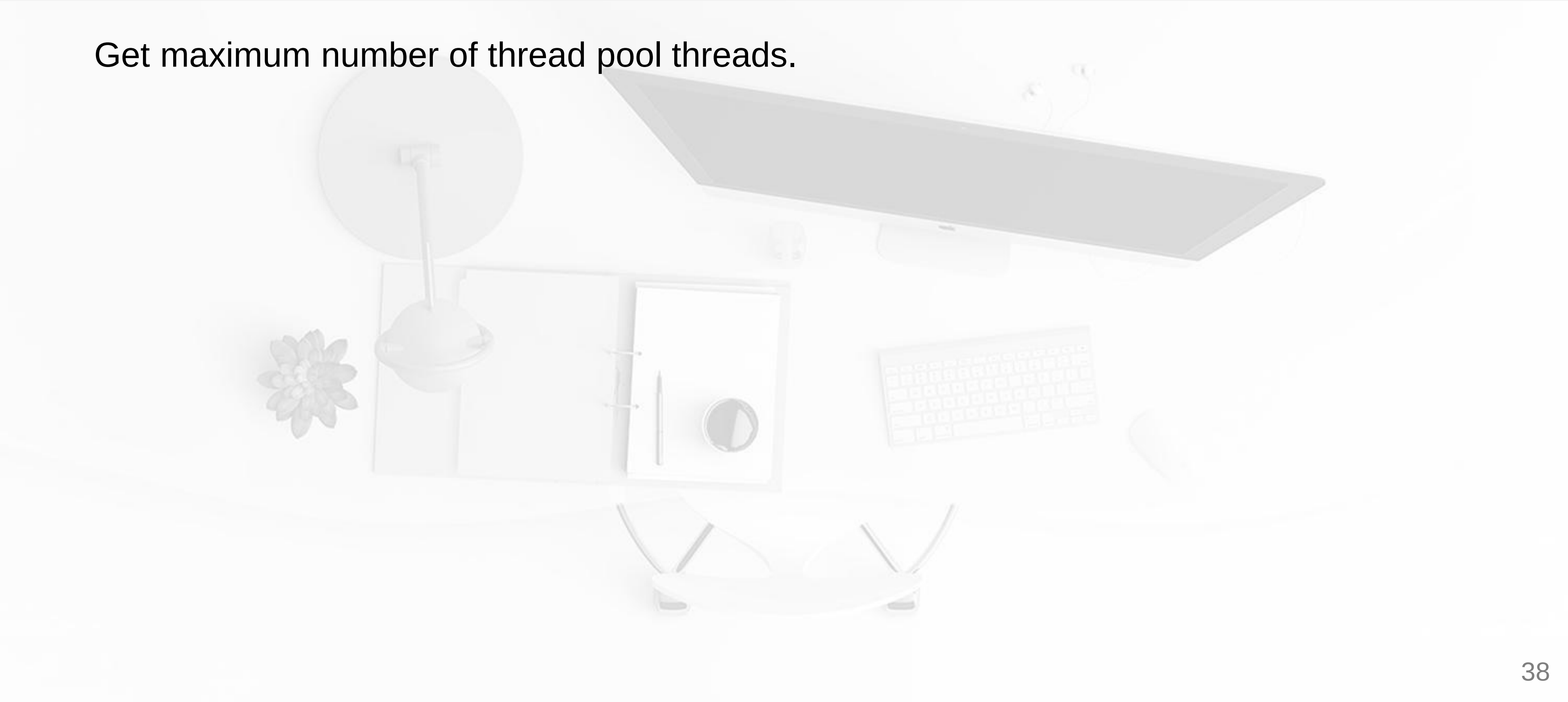


ThreadPool Class

- Namespace: System.Threading
- Provides a pool of threads that can be used to execute tasks, post work items, process asynchronous I/O, wait on behalf of other threads, and process timers

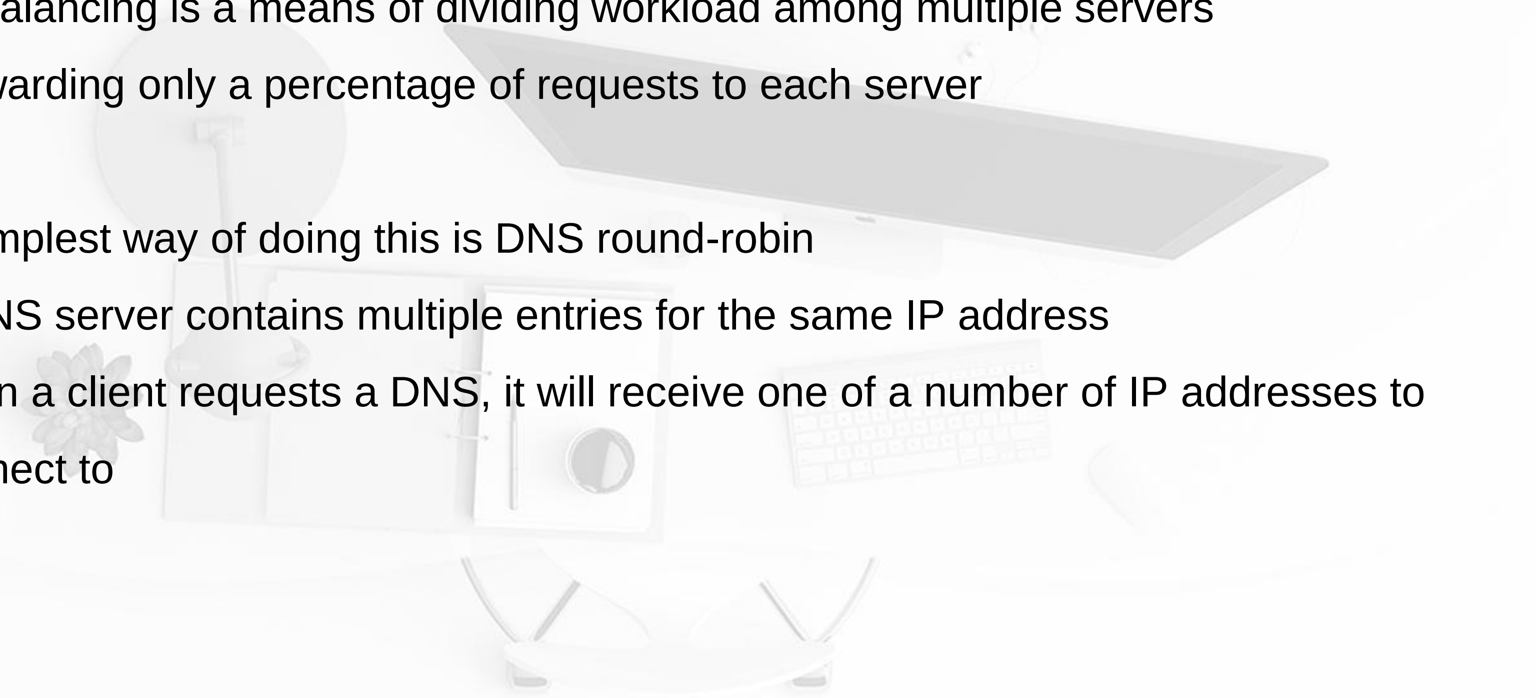
Exercise

Get maximum number of thread pool threads.



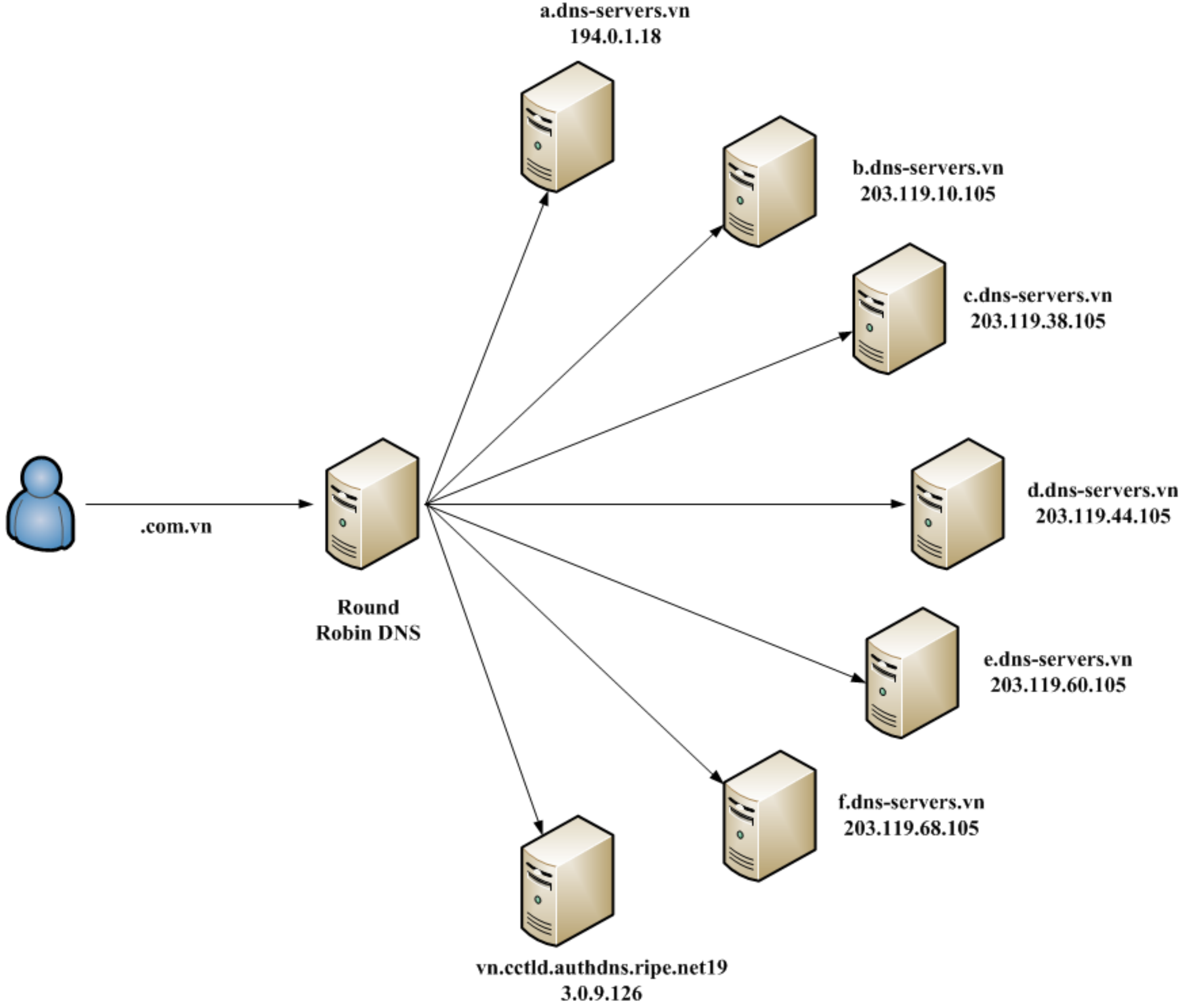
6. Load balancing

Load balancing

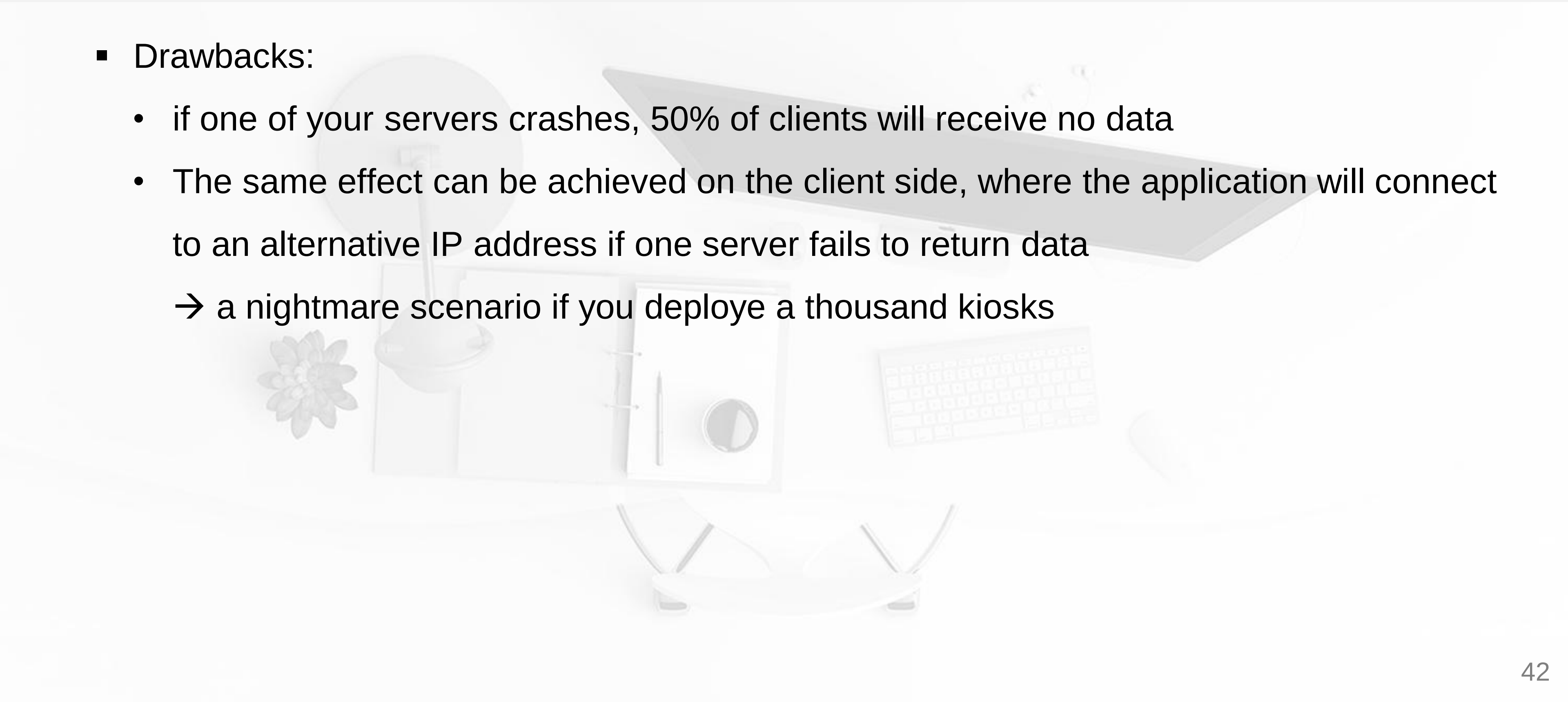


- Load balancing is a means of dividing workload among multiple servers by forwarding only a percentage of requests to each server

- The simplest way of doing this is DNS round-robin
  - a DNS server contains multiple entries for the same IP address
  - when a client requests a DNS, it will receive one of a number of IP addresses to connect to

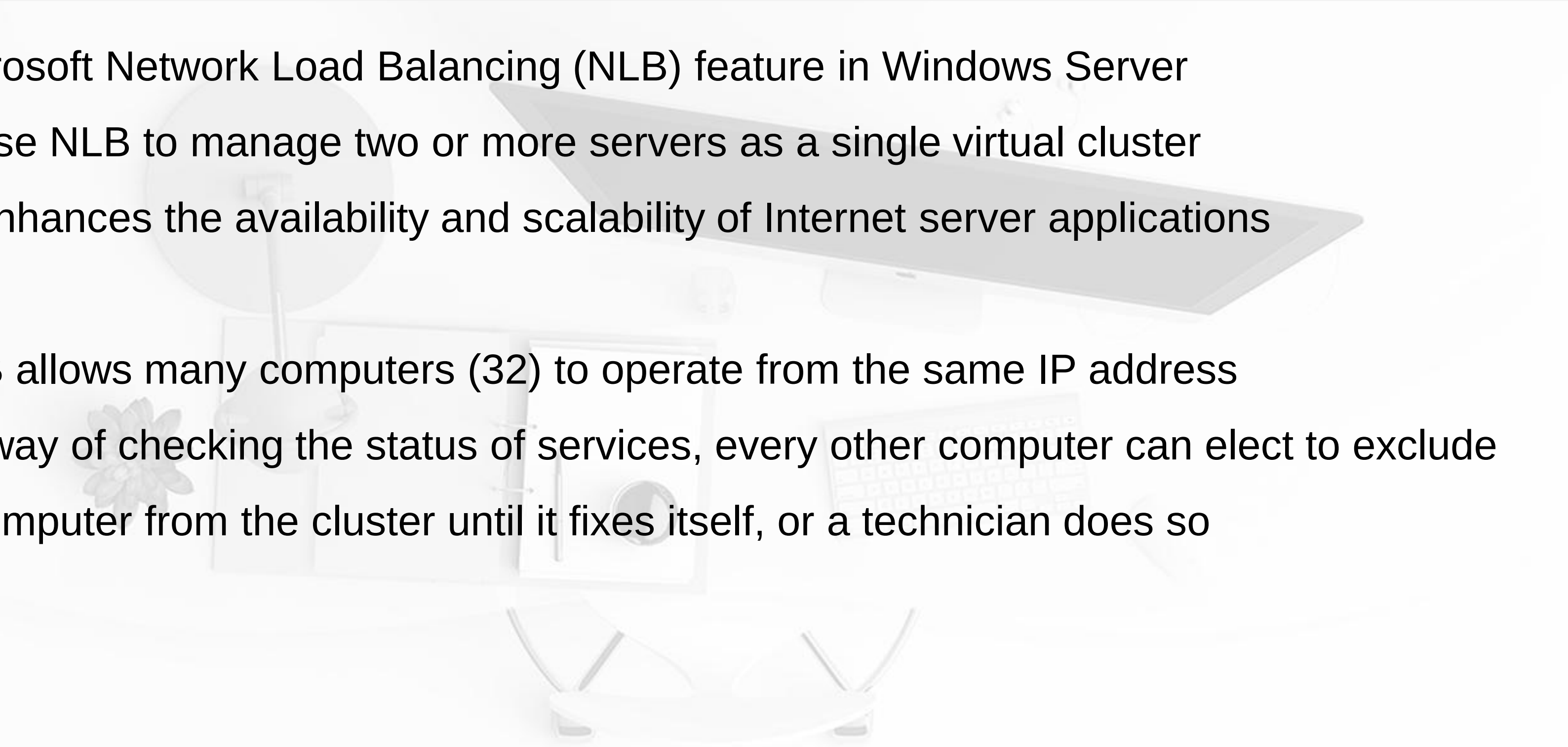

a.dns-servers.vn
194.0.1.18
b.dns-servers.vn
203.119.10.105
c.dns-servers.vn
203.119.38.105
.com.vn
Round
Robin DNS
d.dns-servers.vn
203.119.44.105
e.dns-servers.vn
203.119.60.105
f.dns-servers.vn
203.119.68.105
vn.cctld.authdns.ripe.net19
3.0.9.126

Load balancing



- Drawbacks:
  - if one of your servers crashes, 50% of clients will receive no data
  - The same effect can be achieved on the client side, where the application will connect to an alternative IP address if one server fails to return data
    → a nightmare scenario if you deploye a thousand kiosks

Load balancing



- Microsoft Network Load Balancing (NLB) feature in Windows Server
  - use NLB to manage two or more servers as a single virtual cluster
  - enhances the availability and scalability of Internet server applications

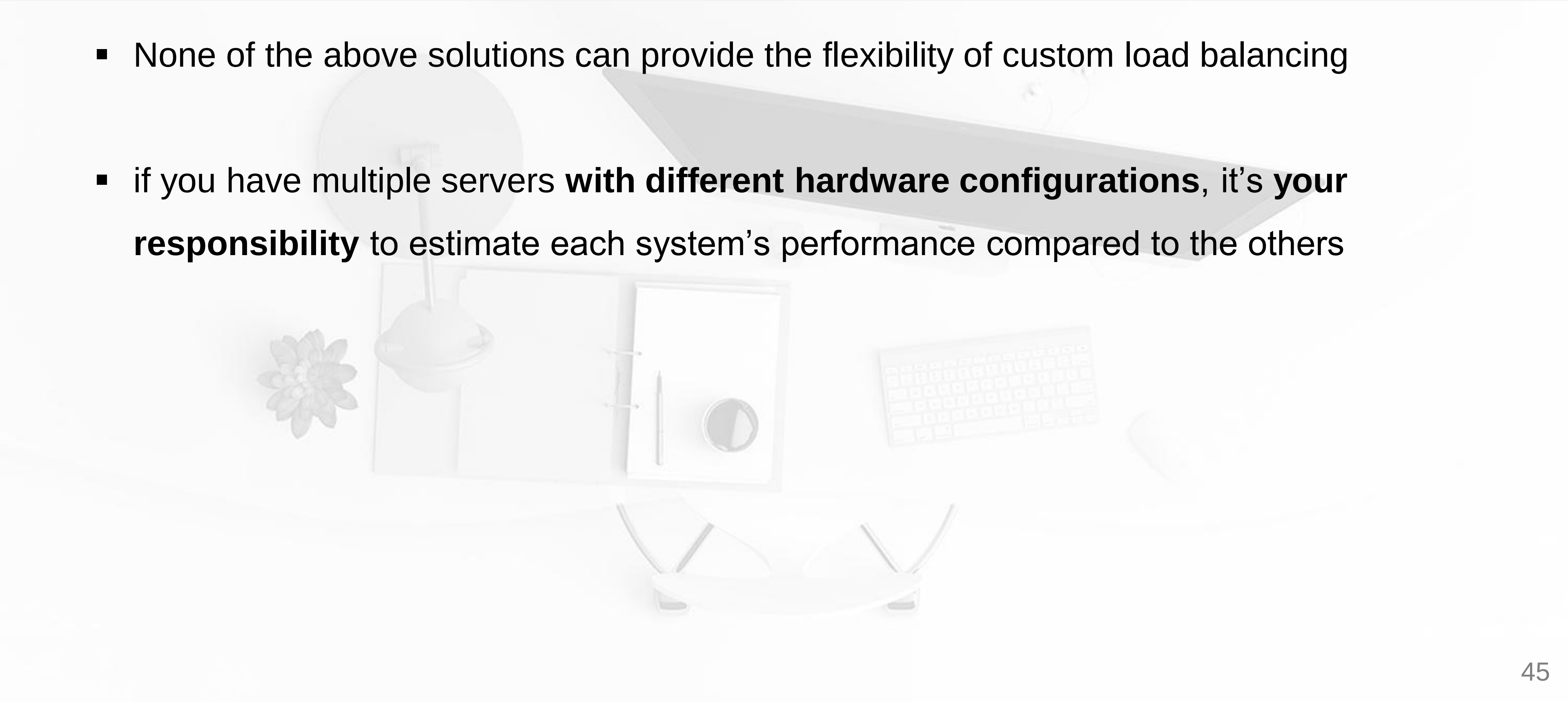
- NLB allows many computers (32) to operate from the same IP address

By way of checking the status of services, every other computer can elect to exclude that computer from the cluster until it fixes itself, or a technician does so

Load balancing

- NLBS is suitable for small clusters, but for highend server farms from between 10 and 8,000 computers, the ideal solution is a **hardware virtual server**, such as Cisco Local Director

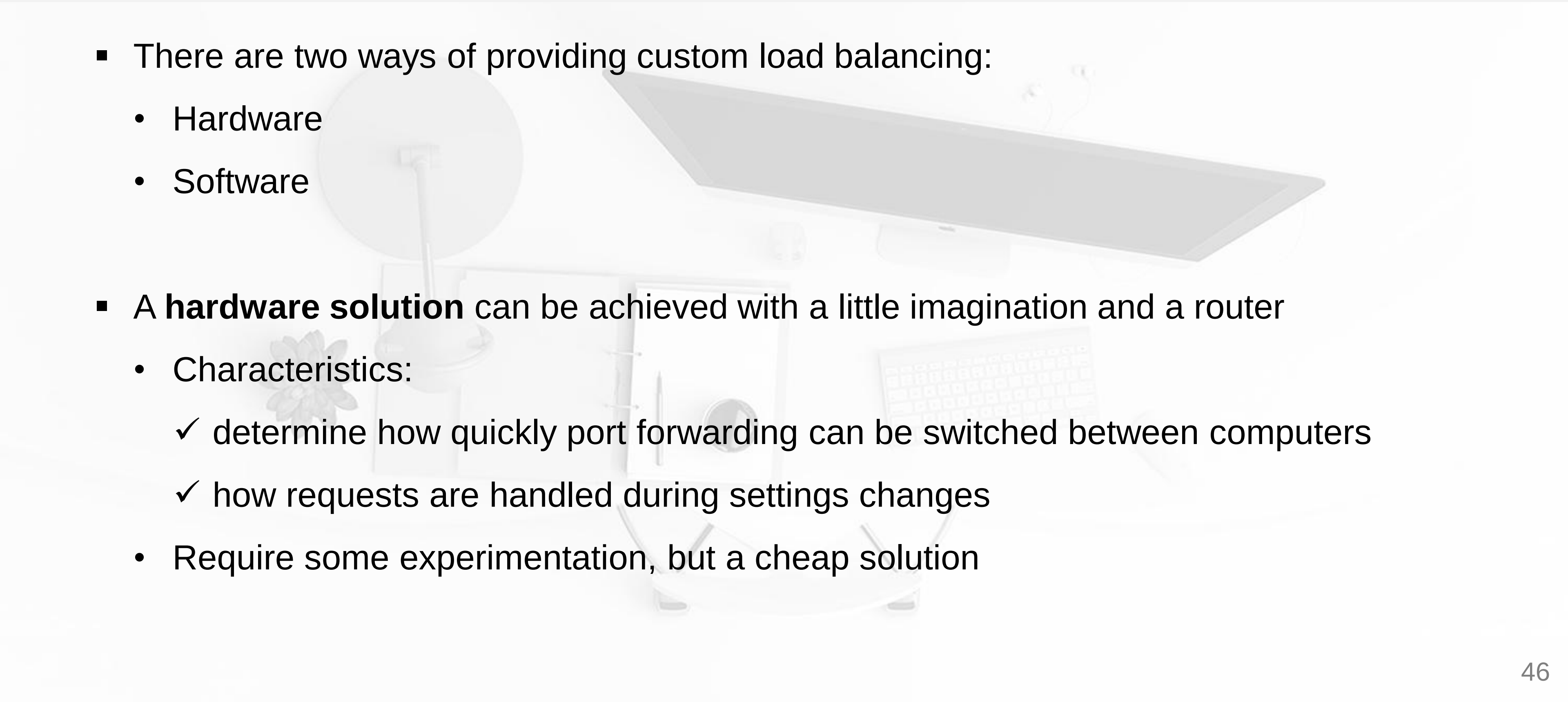
Load balancing



- None of the above solutions can provide the flexibility of custom load balancing

- if you have multiple servers **with different hardware configurations**, it’s **your responsibility** to estimate each system’s performance compared to the others

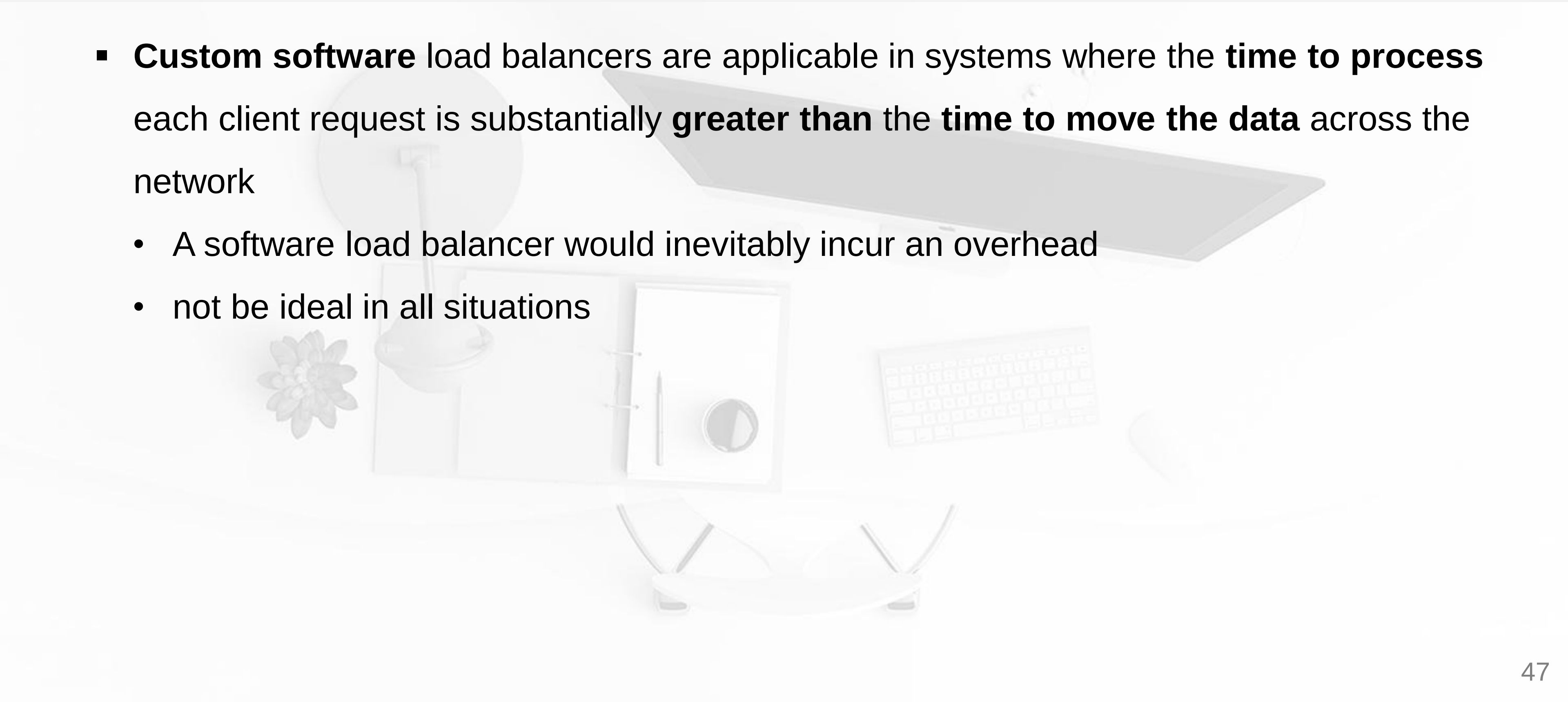
Load balancing



- There are two ways of providing custom load balancing:
  - Hardware
  - Software

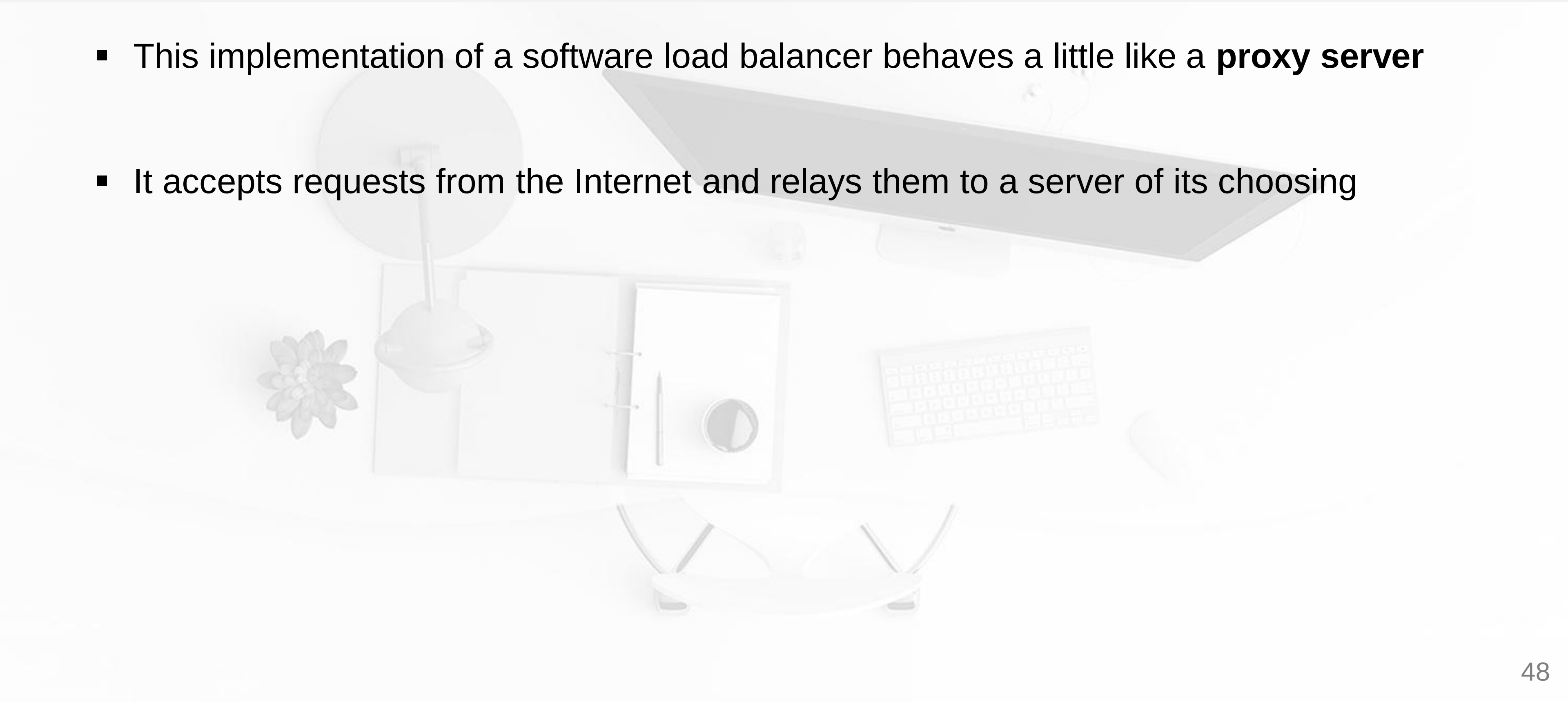
- A **hardware solution** can be achieved with a little imagination and a router
  - Characteristics:
    - ✓ determine how quickly port forwarding can be switched between computers
    - ✓ how requests are handled during settings changes
  - Require some experimentation, but a cheap solution

Load balancing



- **Custom software** load balancers are applicable in systems where the **time to process** each client request is substantially **greater than** the **time to move the data** across the network
  - A software load balancer would inevitably incur an overhead
  - not be ideal in all situations

Load balancing



- This implementation of a software load balancer behaves a little like a **proxy server**

- It accepts requests from the Internet and relays them to a server of its choosing

Q&A