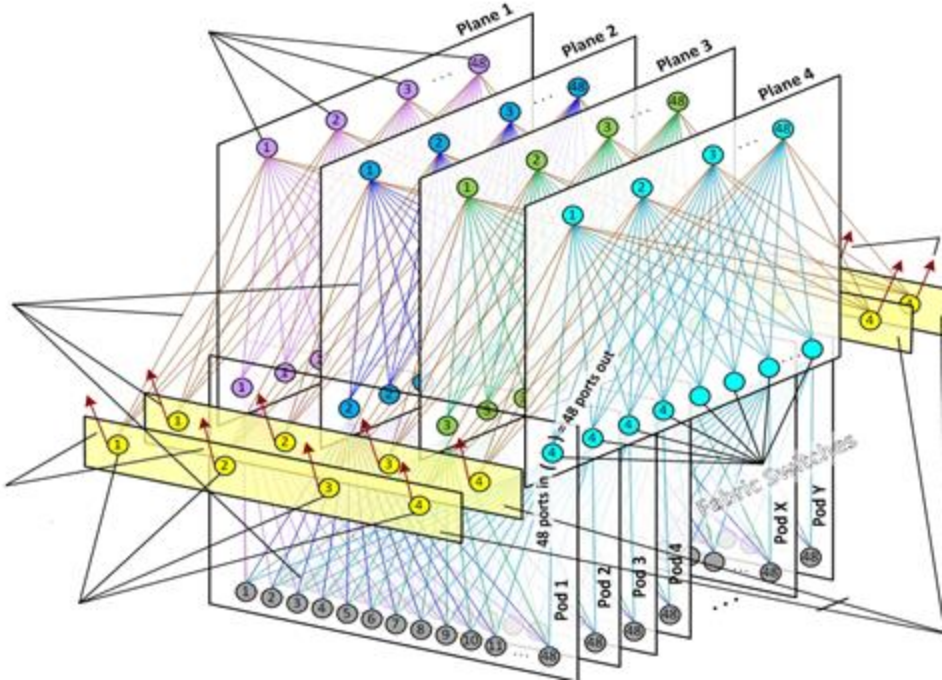
UNIVERSITEIT VAN AMSTERDAM

Large Systems:

Design + Implementation + Administration

2024-2025

➤ Large Systems Introduction



Shashikant Ilager

28 oktober 2024

Large Systems

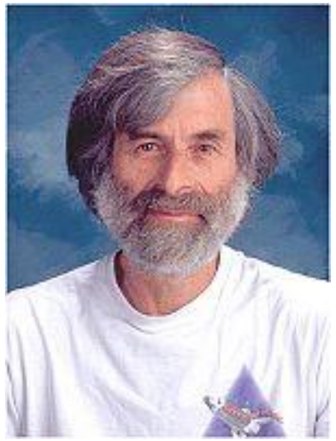
- Obviously not running on a single server
- Large systems today are distributed systems*:
  - *A system in which hardware or software components located at networked computers communicate and coordinate their actions only by message passing.”* [Coulouris]
  - *“A distributed system is a collection of independent computers that appear to the users of the system as a single computer.”* [Steen, Tanenbaum]
- Example Distributed Systems
  - Cluster Computing
  - Cloud Computing
  - Any Computing System that is loosely connected over communication networks

* Current large systems are inherently distributed in nature. So, we will use the words “Large Systems” and “Distributed Systems” interchangeably

Leslie Lamport's Definition

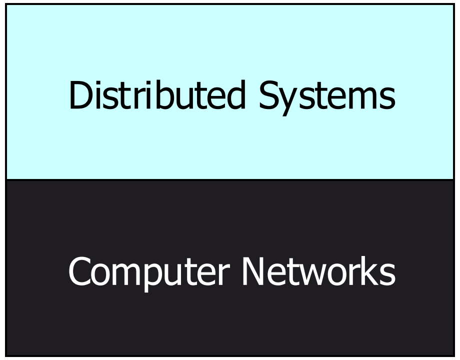
- *"A distributed system is one on which I cannot get any work done because some machine I have never heard of has crashed."*

- Leslie Lamport – a famous researcher on timing, message ordering, and clock synchronization in distributed systems. Turing Award, 2013.



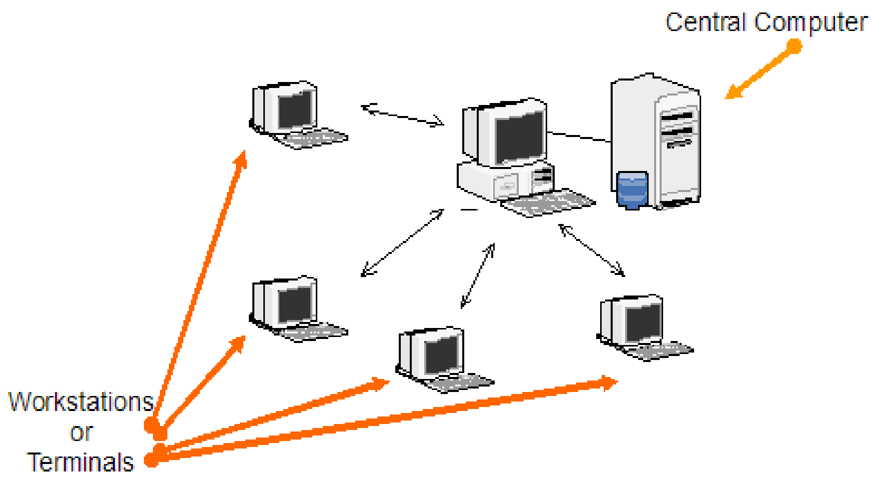
Networks vs. Distributed Systems

- Networks: A media for interconnecting local and wide area computers and exchanging messages based on protocols. Network entities are visible and they are explicitly addressed (IP address).
- Distributed System: existence of multiple autonomous computers is transparent
- However,
  - many problems (e.g., openness, reliability) in common, but at different levels.
    - Networks focuses on packets, routing, etc., whereas distributed systems focus on applications.
    - Every distributed system relies on services provided by a computer network.



Reasons for Distributed Systems

- Functional Separation:
  - Existence of computers with different capabilities and purposes:
- Clients and Servers
  - Data collection and data processing
- Inherent distribution
  - Information:
    - Different information is created and maintained by different people (e.g., Web pages)
  - People
    - Computer-supported collaborative work (virtual teams, engineering, virtual surgery)
    - Retail store and inventory systems for supermarket chains (e.g., Albert Heijn,)

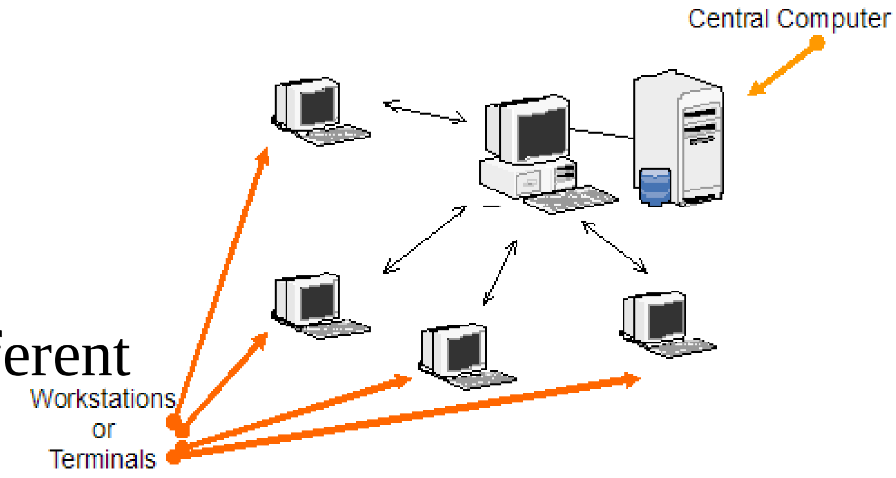


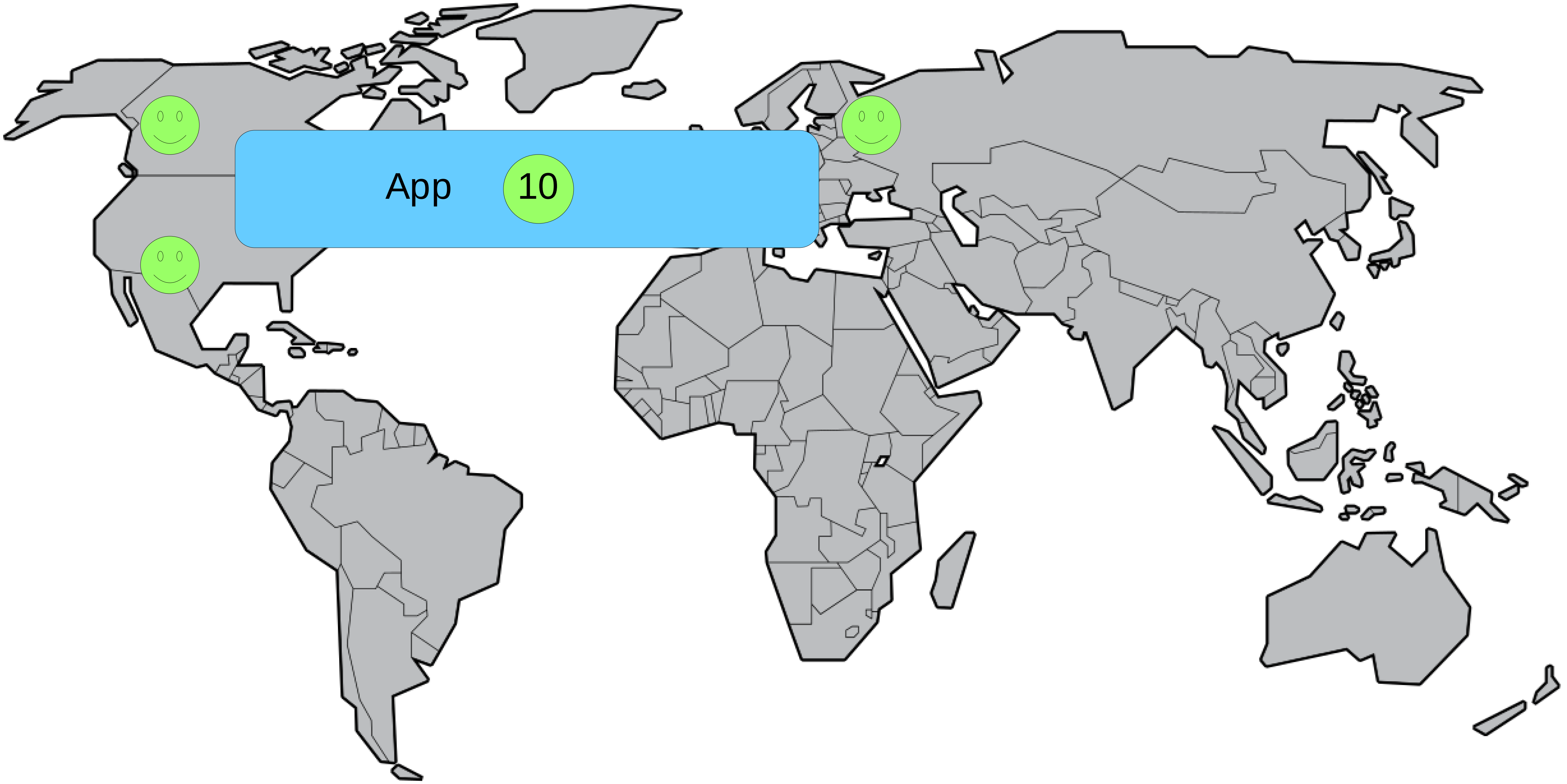
Reasons for Distributed Systems

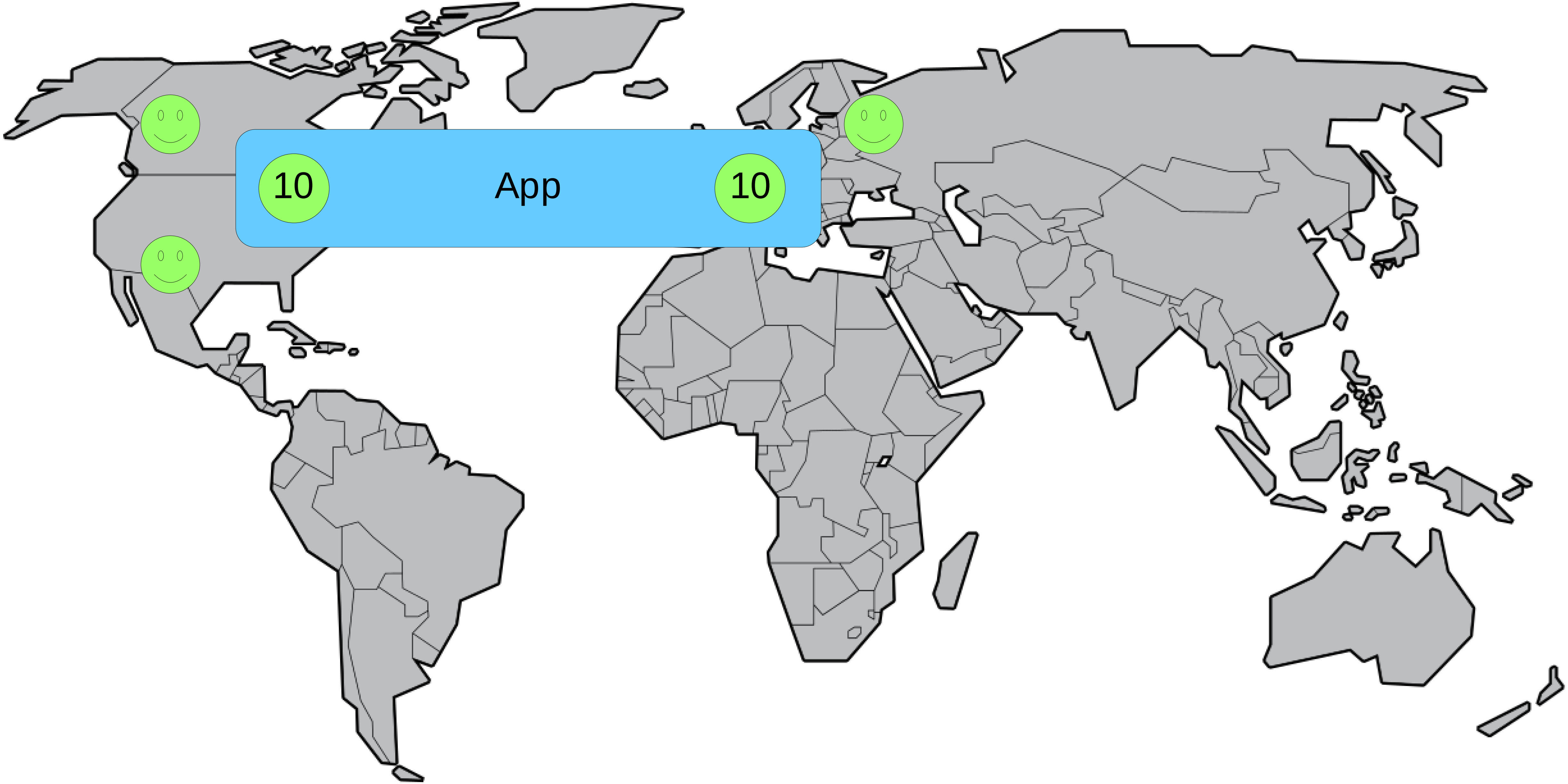
- Power imbalance and load variation:
  - Distribute computational load among different computers
- Reliability:
  - Long term preservation and data backup (replication) at different locations.
- Economies:

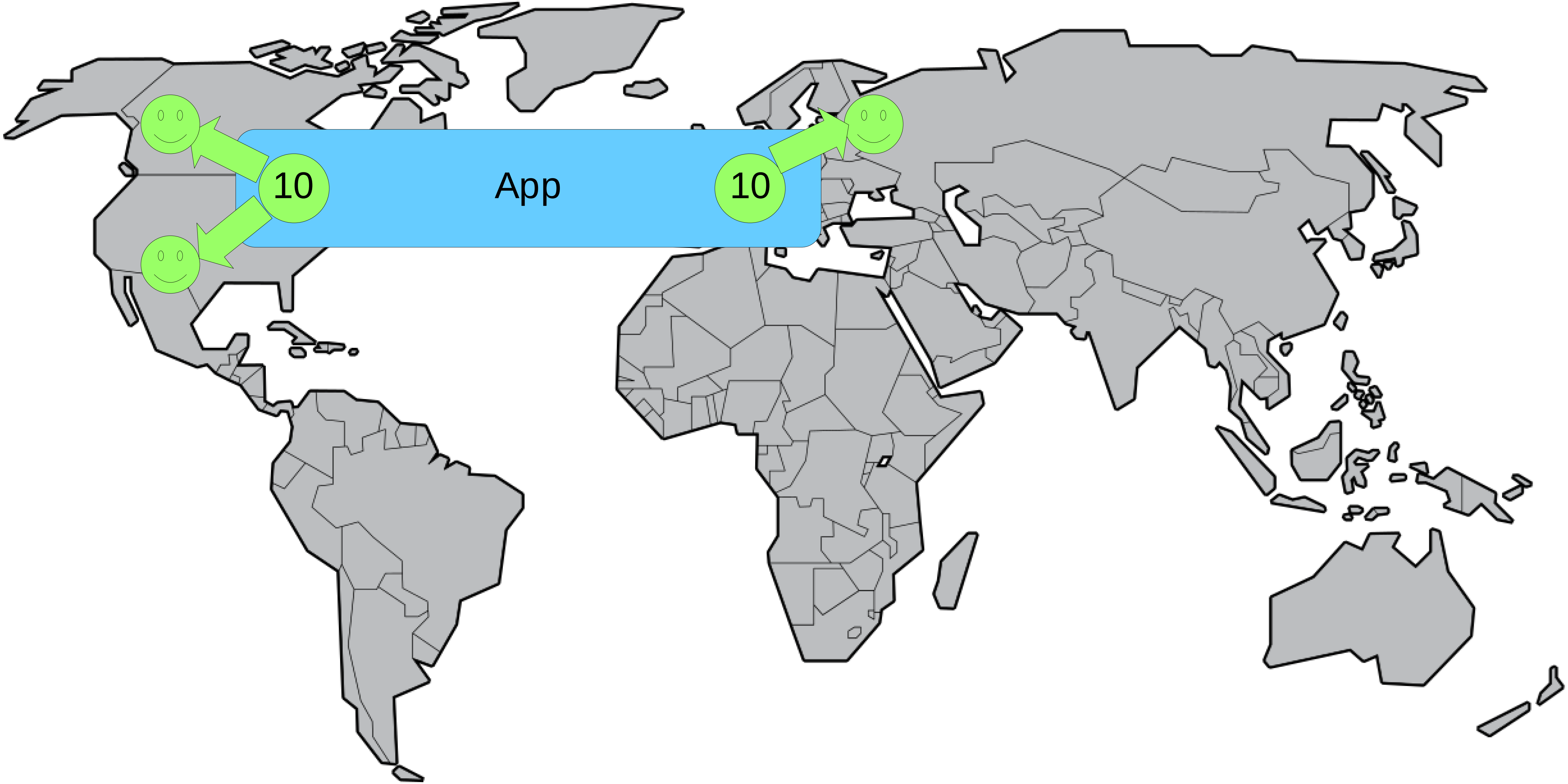
  Sharing a printer by many users and reduce the cost of ownership.

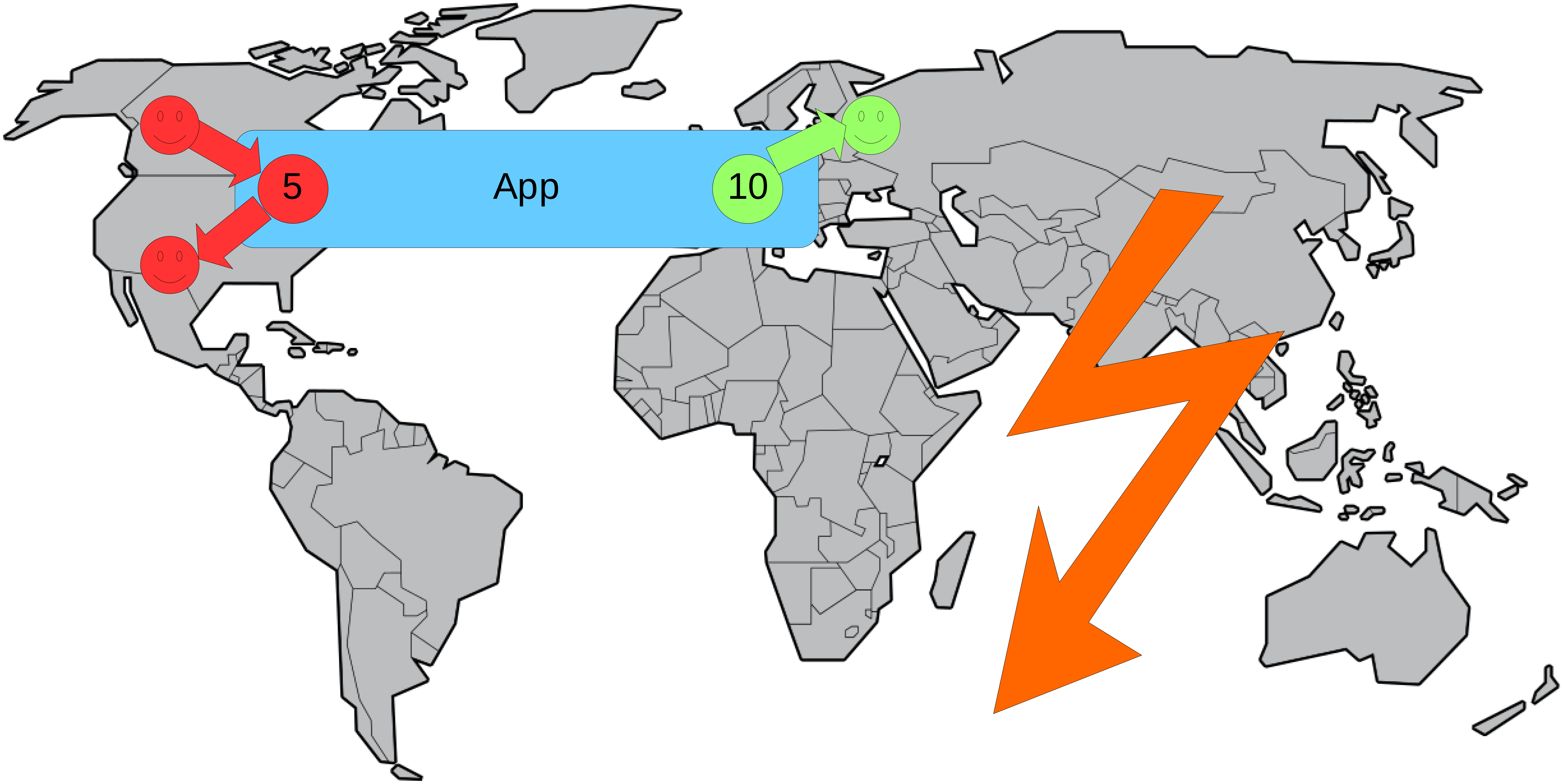
  Building a supercomputer out of a network of computers.




App
10


10
App
10


10
App
10


5
App
10

Transparency

- To appear as a single system, things must be hidden:
  - **Distribution transparency:** The location should be unknown to users
  - **Failure transparency:** A failure in hardware/software components shouldn’t be observable by end users.
  - **Replication transparency:** Transparent Data or Service replication should be designed to handle dynamic workloads and provide reliability
  - **Relocation transparency:** Any service migrations shouldn’t hinder performance or visibility

But

- Aiming at full distribution transparency may be too much:
  - There are communication latencies that cannot be hidden
  - Completely hiding failures of networks and nodes is (theoretically and practically) impossible. A very hard problem. Imagine the recent Windows blue screen problem
  - You cannot distinguish a slow computer from a failing one
  - You can never be sure that a server actually performed an operation before a crash

  …

Source: Maarten van Steen, “Distributed Systems”, 2017

But

- Full transparency will cost performance, exposing the distribution of the system
  - Keeping replicas exactly up-to-date with the master takes time
  - Immediately flushing write operations to disk for fault tolerance has costs

Hiding Failures

- Theoretically impossible to tell failed machine from slow machine
  - “No consensus protocol is totally correct in spite of one fault.” Mathematical proof [Fischer *et al*, 1985]*
  - If packets can take infinite time to arrive
  - Cannot tell whether packet late or the machine dead
- Cannot tell whether remote operation succeeded
  - Due to packet loss and server crashes

*https://groups.csail.mit.edu/tds/papers/Lynch/jacm85.pdf

Failures in Practice

The Joys of Real Hardware

Typical first year for a new cluster:

~1 network rewiring (rolling ~5% of machines down over 2-day span)

~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)

~5 racks go wonky (40-80 machines see 50% packetloss)

~8 network maintenances (4 might cause ~30-minute random connectivity losses)

~12 router reloads (takes out DNS and external vips for a couple minutes)

~3 router failures (have to immediately pull traffic for an hour)

~dozens of minor 30-second blips for dns

~1000 individual machine failures

~thousands of hard drive failures

slow disks, bad memory, misconfigured machines, flaky machines, etc.

Long distance links: wild dogs, sharks, dead horses, drunken hunters, etc.

Source: Jeff Dean, Google

Transparency Costs Performance

Numbers Everyone Should Know

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	25 ns
Main memory reference	100 ns
Compress 1K w/cheap compression algorithm	3,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from disk	20,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

Source: Jeff Dean, Google

Google

We’re Doomed!

Well, Partially

- Often can tell whether the machine is dead
  - Internet is partially synchronous
  - Most applications do not need up-to-date replicas in both California and The Netherlands
- General solutions are a few
  - Need application-specific solutions! (Mostly requiring middleware systems)

Dev+Ops Challenge

- So building distributed systems is a challenge
- So is administrating them
  - 24x7 users, no “scheduled downtime”
  - If many many components, failures are frequent
  - No more manual solutions (patch servers with CD)
  - Size of backups
  - Complexity!
- Additional complication: fast pace of business innovation

Dimensions of Scale

- Numerical Size

  #users

  #machines

  #data

- Geographical

  LAN / Region / Country / Continent / Global

- Administrative

  One owner / Many owners / Anarchy

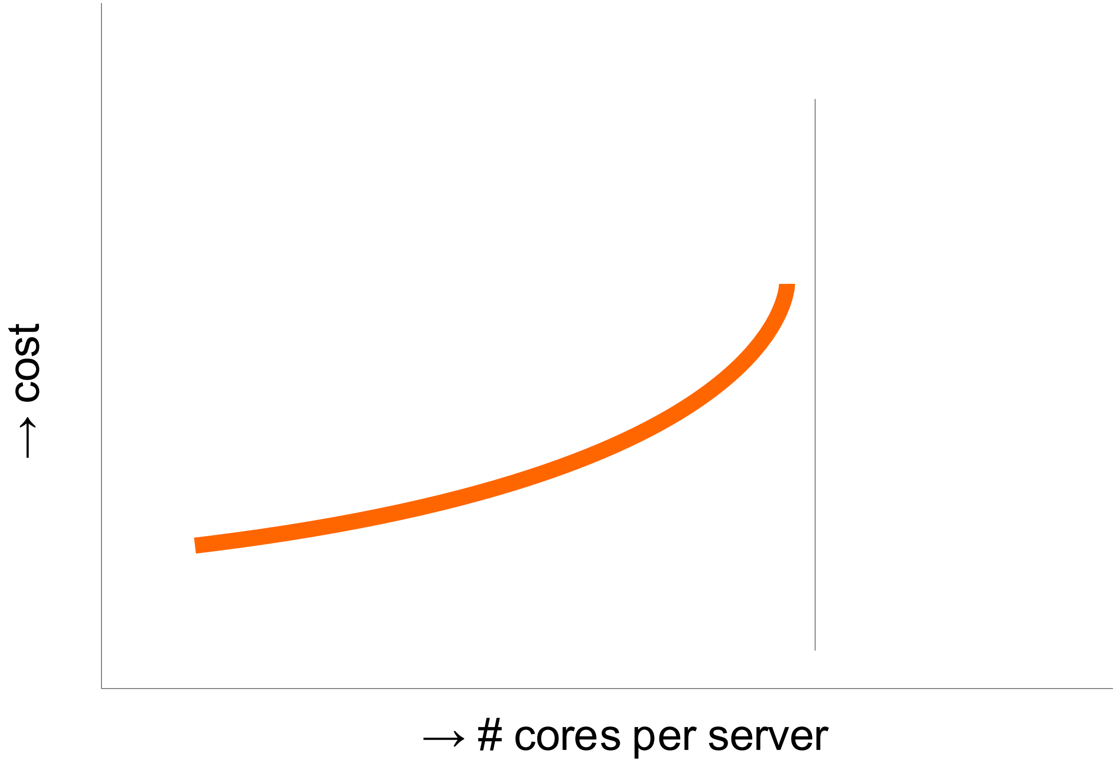
Source: [Neumann, 1994]

Scaling Techniques

- Bigger machines ("Scale up")
- Replication:
  - Caching: temporary replication
- Partitioning:
  - AKA Sharding
- Asynchronous communication
- Virtualization
  - Aggregate physical parts into a larger logical whole
  - Decouple interface from implementation, e.g. NFS

Scaling Up

- Buying bigger hardware
- Costs do not increase linearly!
- At some point, there is no bigger model...
- Complex hardware has complex behaviour
- Software must be able to use it!



From: “Scalability Rules”, 2nd Ed. Abbott and Fischer, 2017

Case Study: Google Evolution

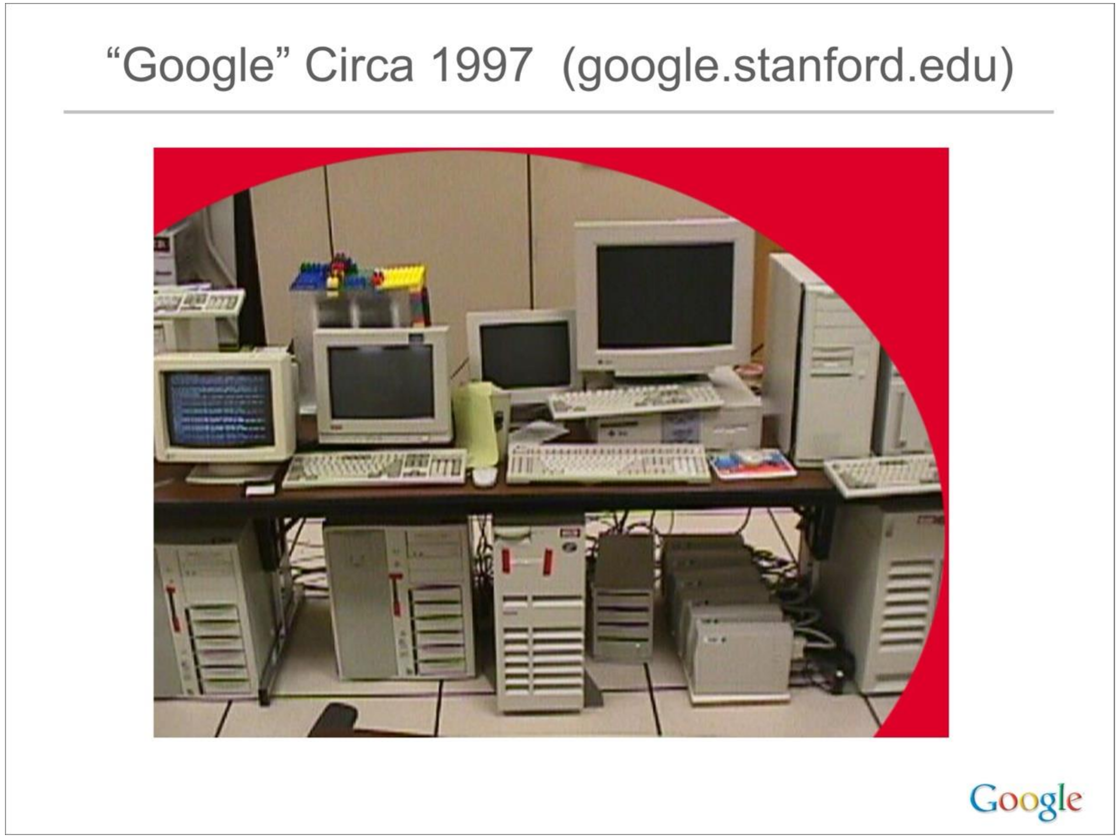
- Jeff Dean, "Building Software Systems at Google and Lessons Learned", Stanford Computer Science Department, Distinguished Computer Scientist Lecture, November, 2010

- https://research.google.com/pubs/jeff.html
- https://www.youtube.com/watch?v=modXC5IWTJI

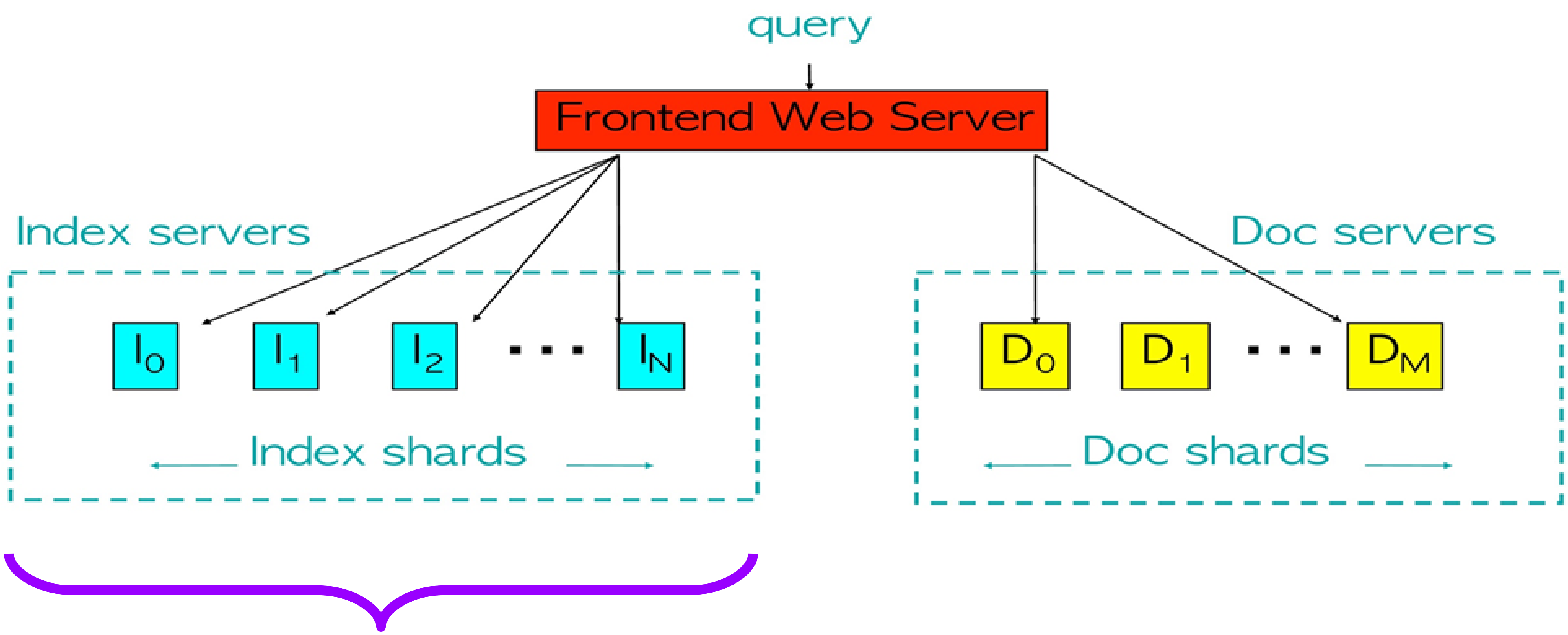
Google Search

- User types query: “uva-sne”
- Google needs to find documents on Web that match
- Step 1: Find out which documents match
  - From <query> get list of <docid,score> pairs
  - <” uva-sne”> → <1234,10.0>,<5678,8.0>
- Step 2: Get document summary
  - From <docid,query> generate <title,snipplet>
  - <1234,” uva-sne”> → <”uva/masters”,”SNE homepage”>
- Step 3: Generate HTML

Google Infra in 97….

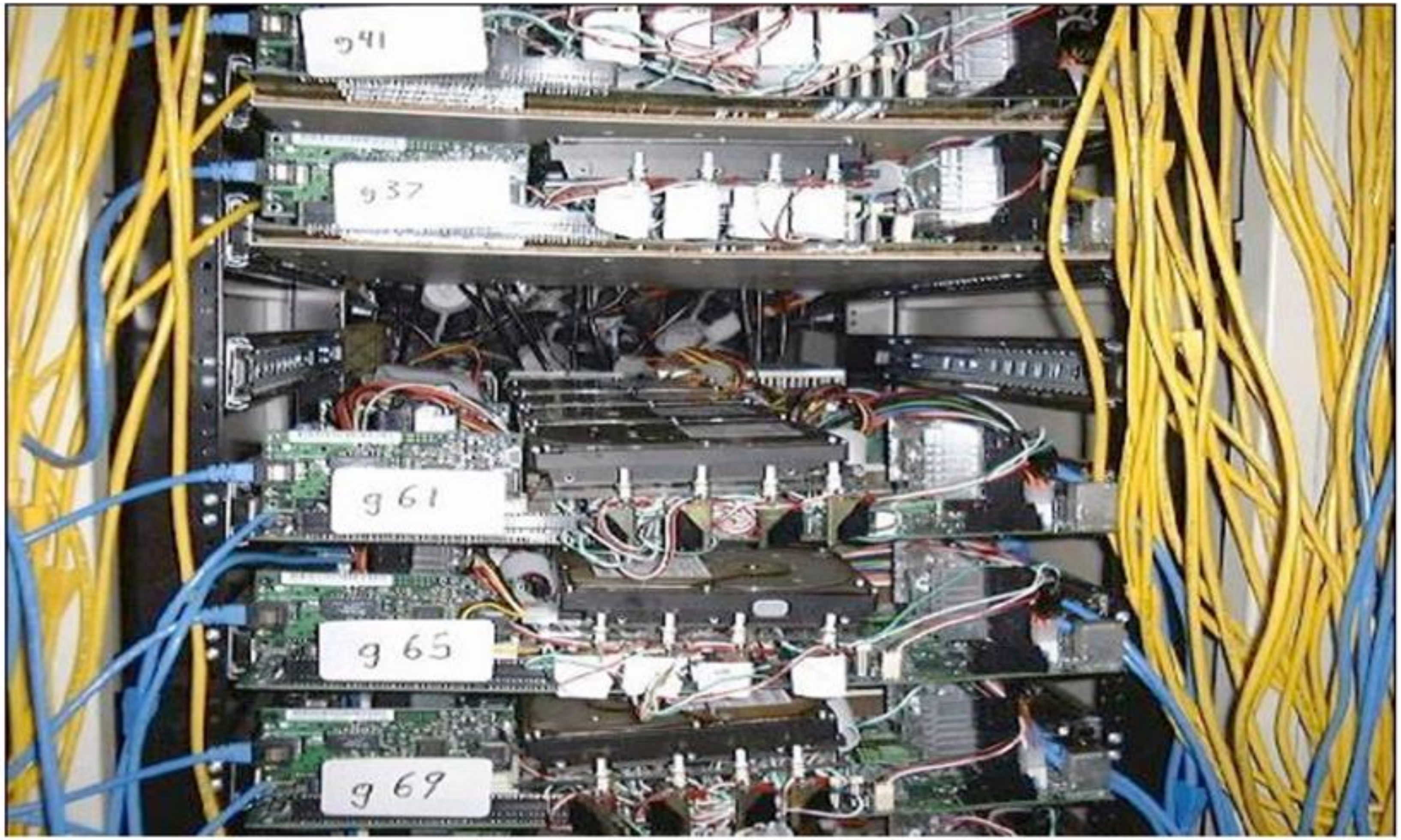


Research Project, circa 1997

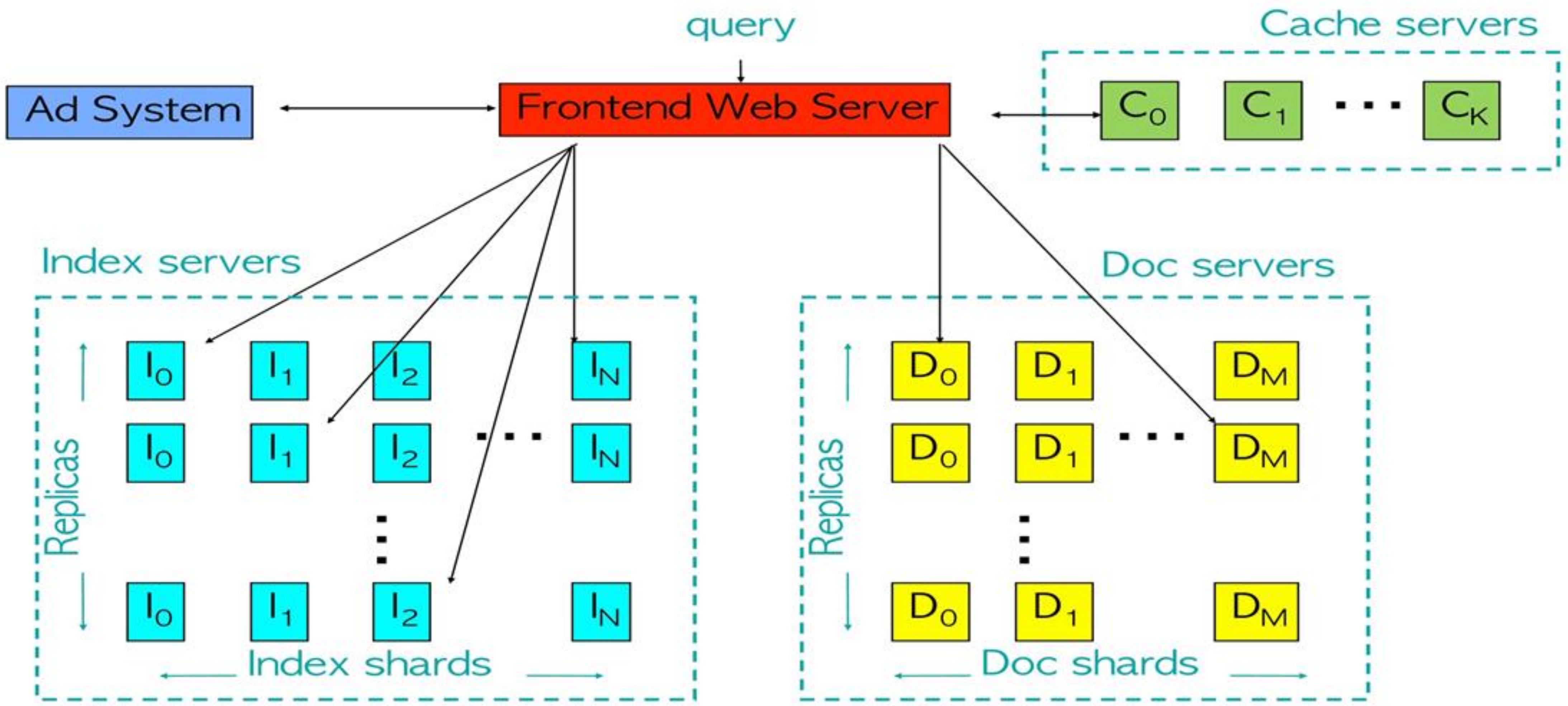


Index of all documents on the Internet

"Corkboards" (1999)



Serving System, circa 1999



Google

Caching in Web Search Systems

- Cache servers:
  - cache both index results and doc snippets
  - hit rates typically 30-60%
    - depends on frequency of index updates, mix of query traffic, level of personalization, etc
- Main benefits:
  - performance! a few machines do work of 100s or 1000s
  - much lower query latency on hits
    - queries that hit in cache tend to be both popular and expensive (common words, lots of documents to score, etc.)
- Beware: big latency spike/capacity drop when index updated or cache flushed

Google

Indexing (circa 1998-1999)

- Simple batch indexing system
  - No real checkpointing, so machine failures painful
  - No checksumming of raw data, so hardware bit errors caused problems
    - Exacerbated by early machines having no ECC, no parity
    - Sort 1 TB of data without parity: ends up "mostly sorted"
    - Sort it again: "mostly sorted" another way

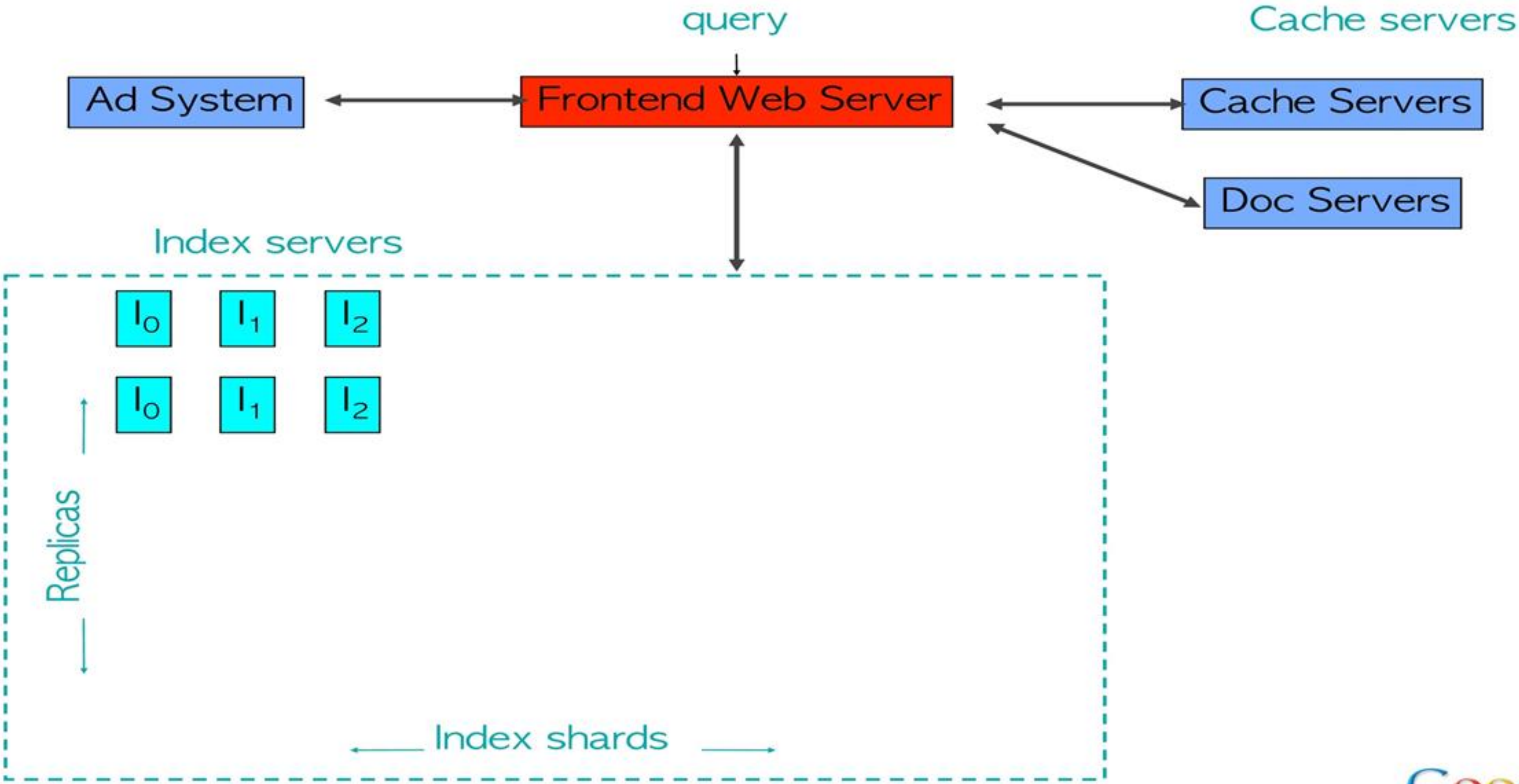
- "Programming with adversarial memory"
  - Developed file abstraction that stores checksums of small records and can skip and resynchronize after corrupted records

Google Data Center (2000)

Increasing Index Size and Query Capacity

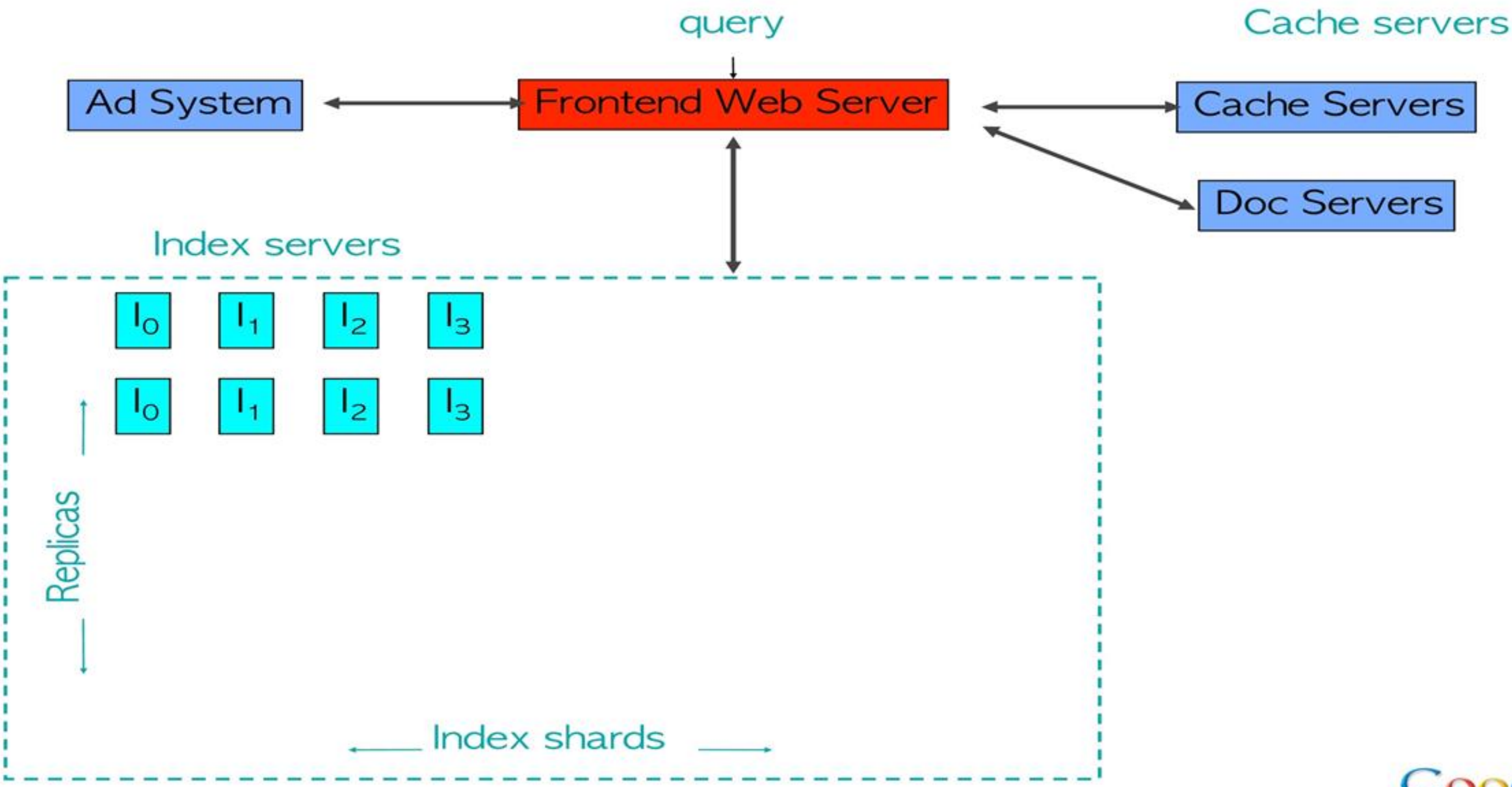
- Huge increases in index size in ’99, ’00, ’01, ...
  - From ~50M pages to more than 1000M pages
- At same time as huge traffic increases
  - ~20% growth per month in 1999, 2000, ...
  - ... plus major new partners (e.g. Yahoo in July 2000 doubled traffic overnight)
- Performance of index servers was paramount
  - Deploying more machines continuously, but...
  - Needed ~10-30% software-based improvement every month

Dealing with Growth



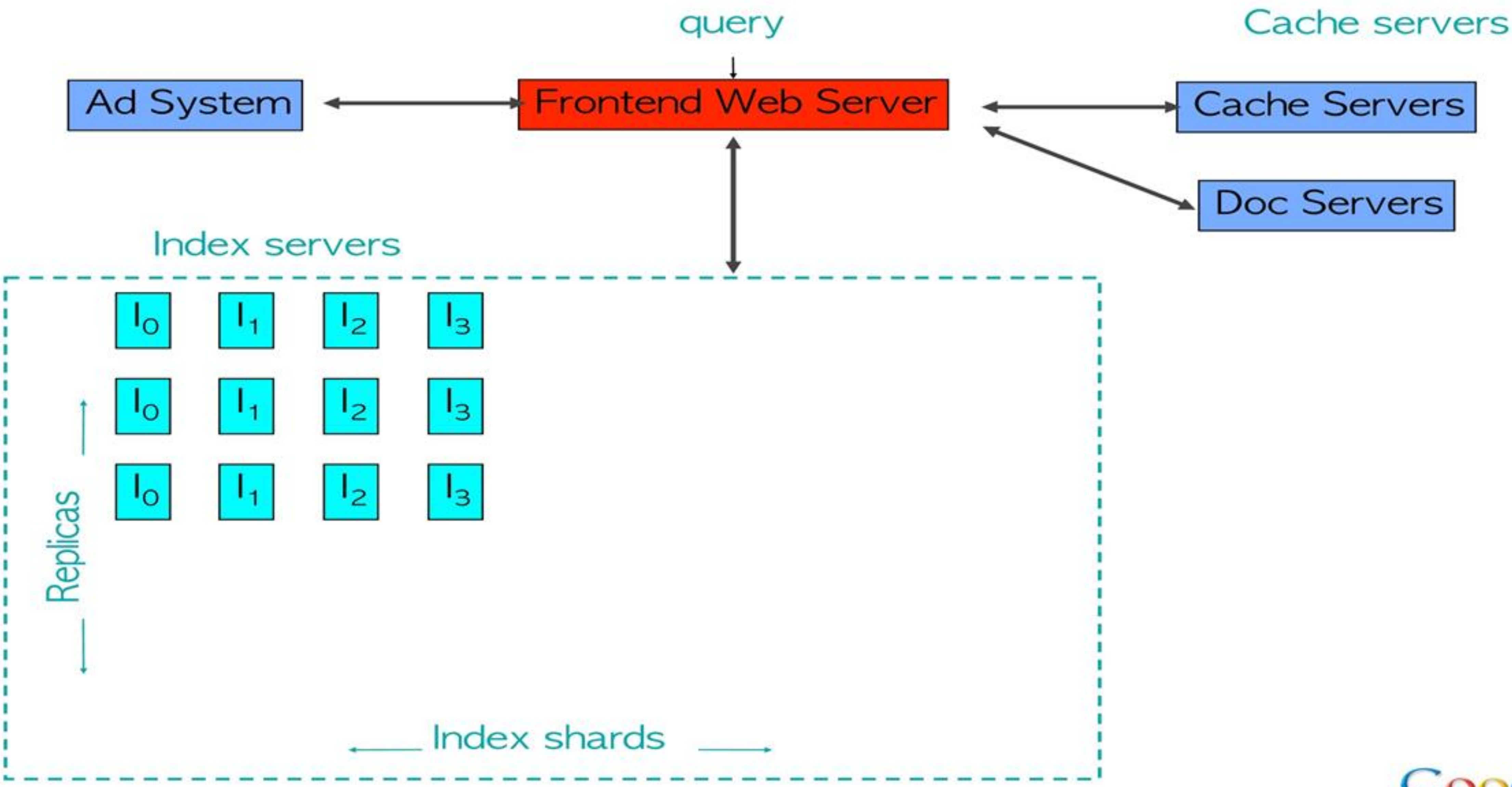
Google

Dealing with Growth

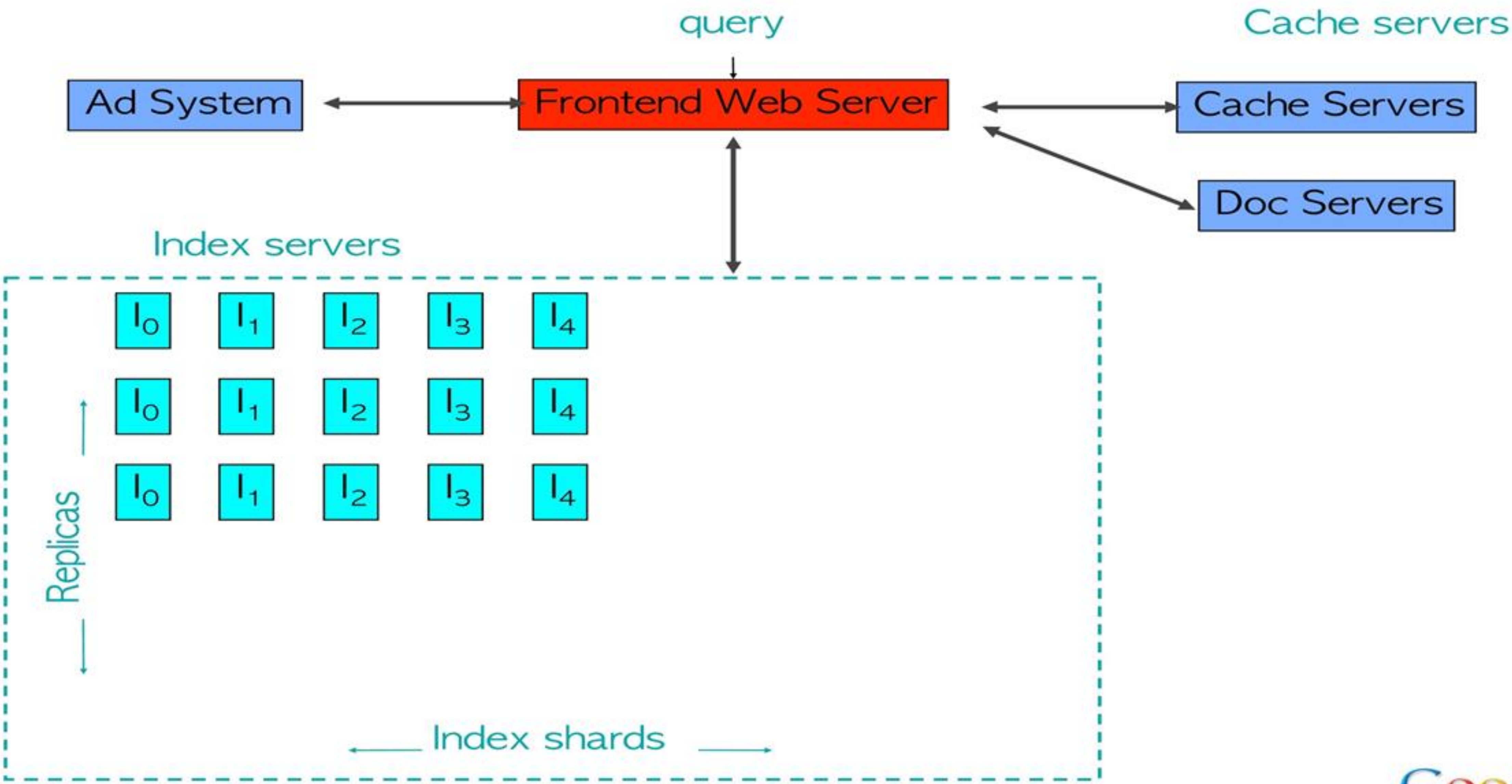


Google

Dealing with Growth

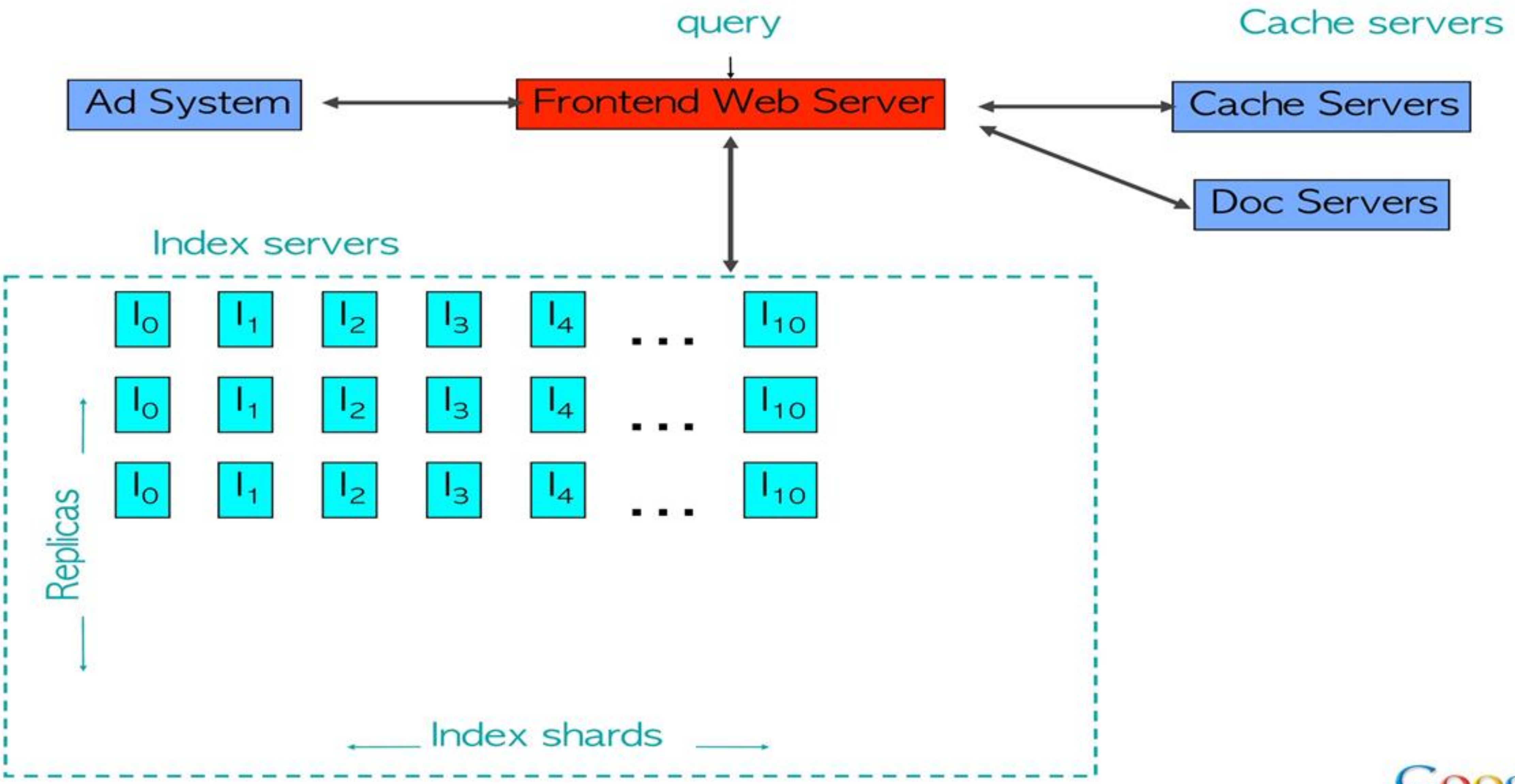


Dealing with Growth



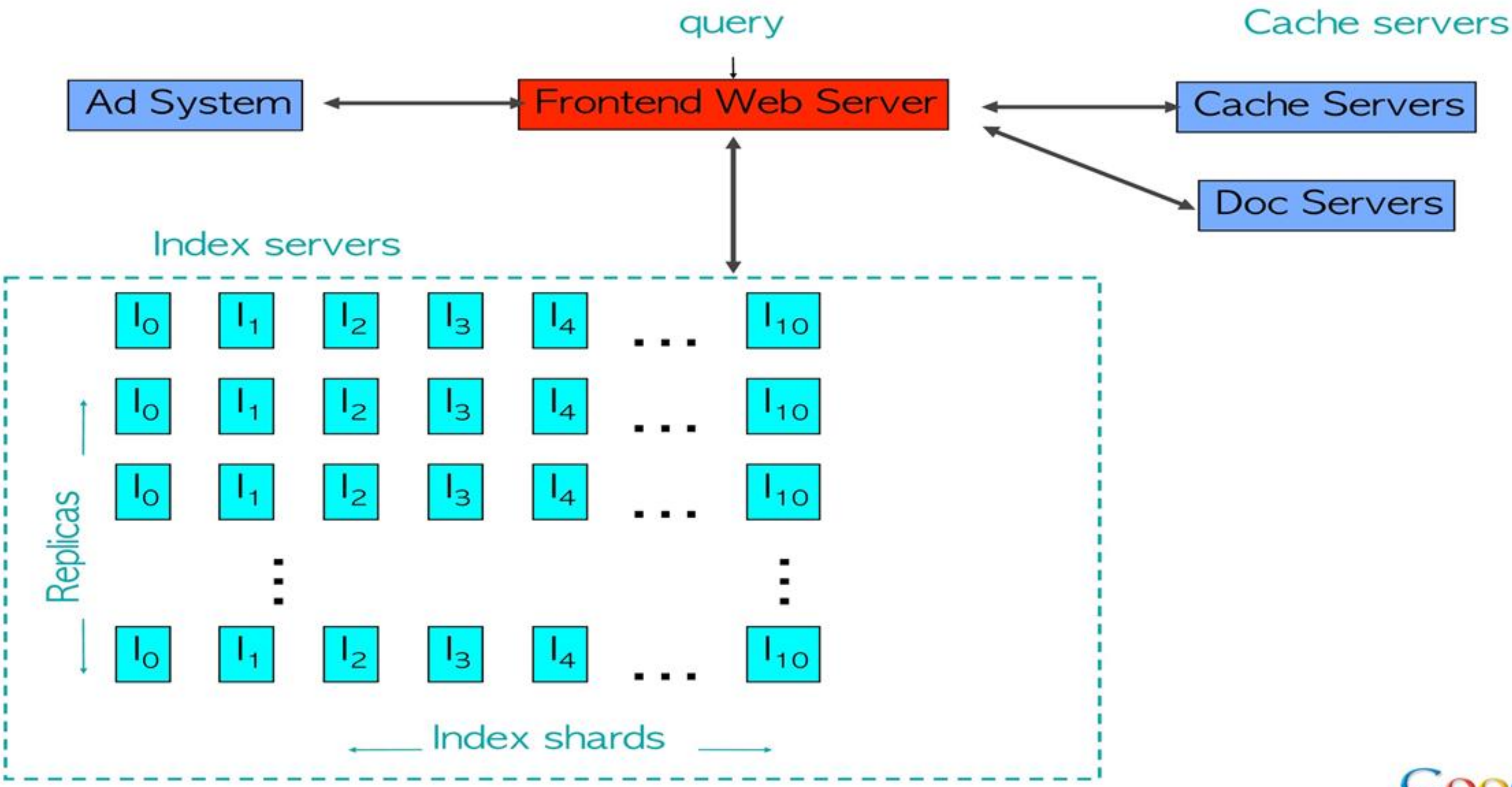
Google

Dealing with Growth



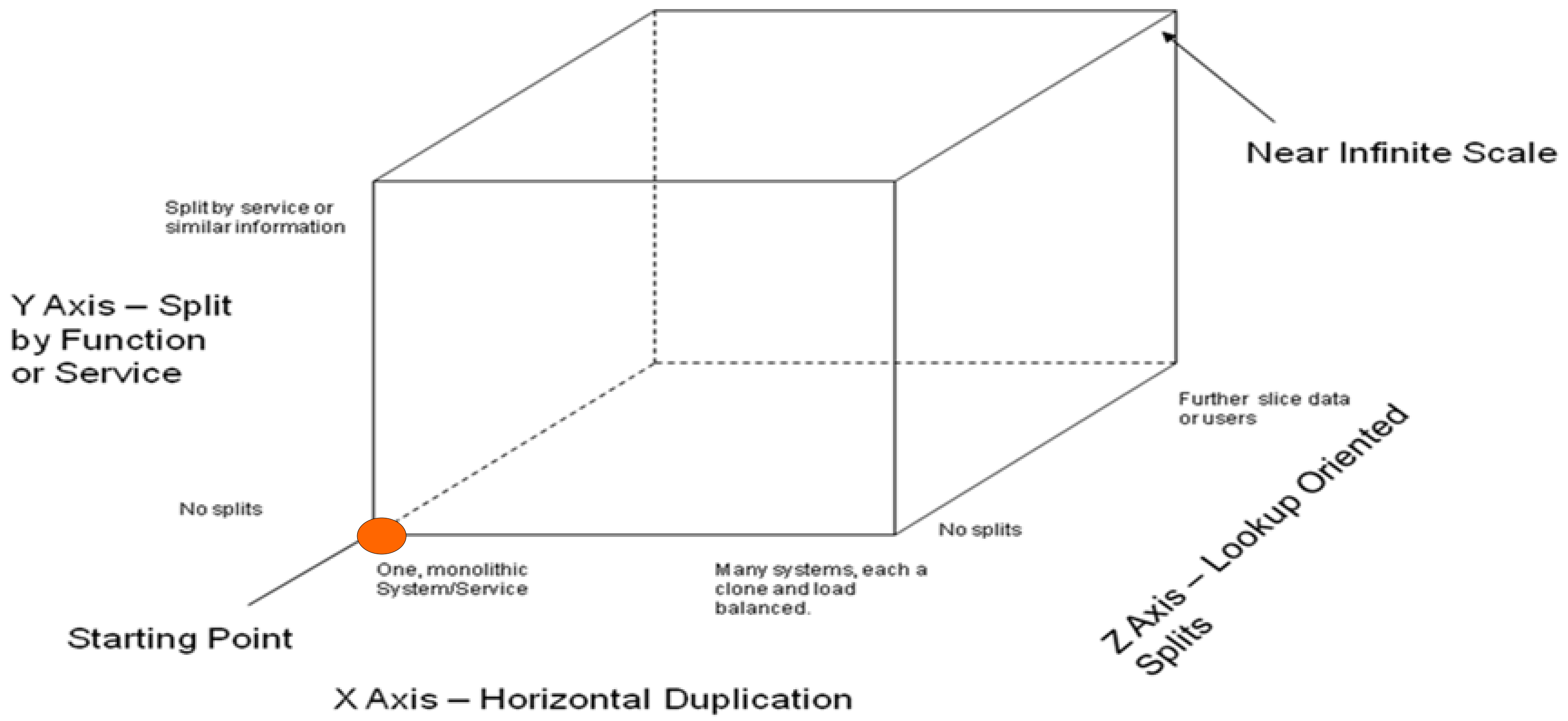
Google

Dealing with Growth



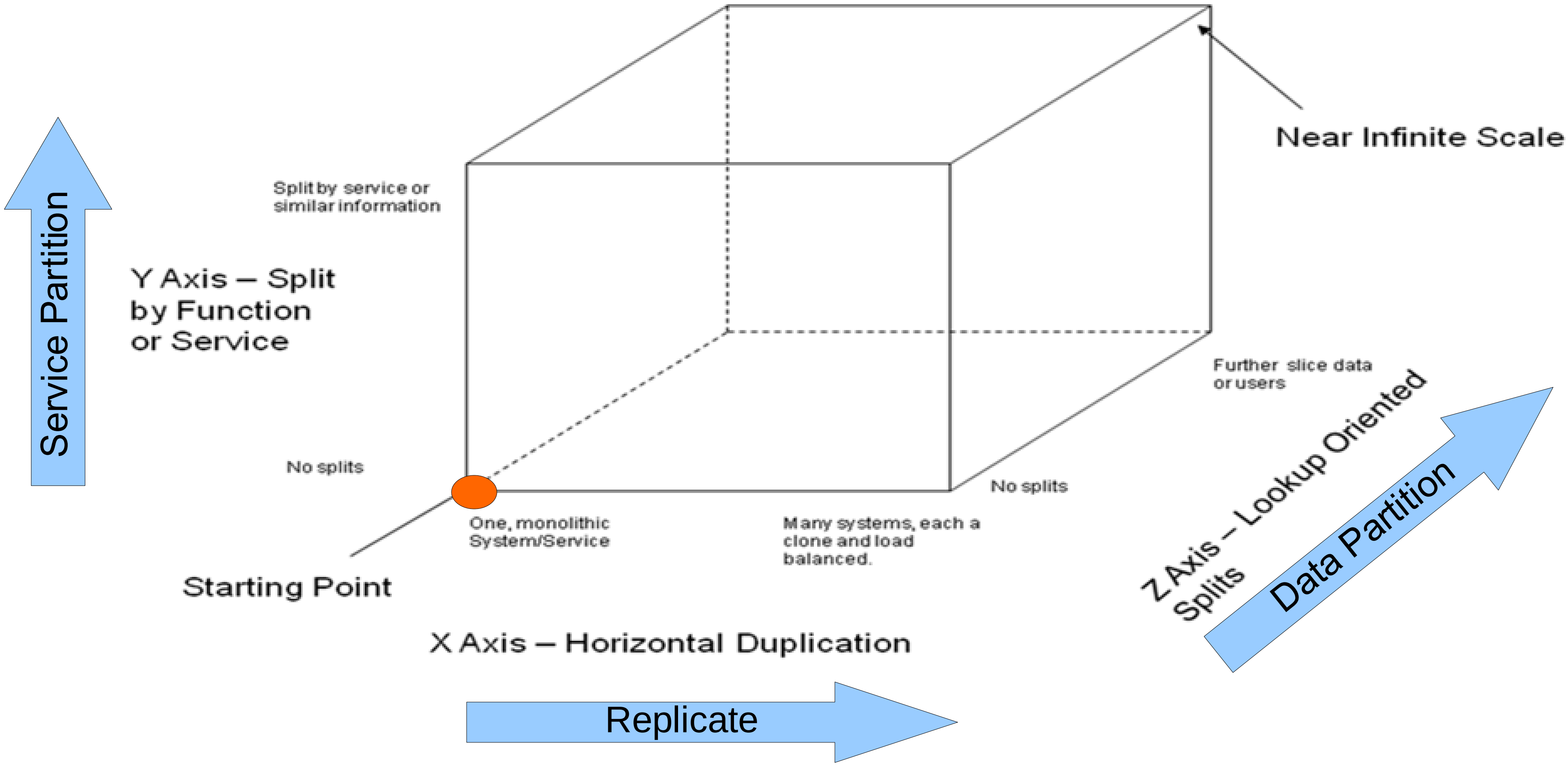
Google

AKF Scale Cube



Source: http://akfpartners.com/growth-blog/splitting-applications-or-services-for-scale

AKF Scale Cube



Source: http://akfpartners.com/growth-blog/splitting-applications-or-services-for-scale

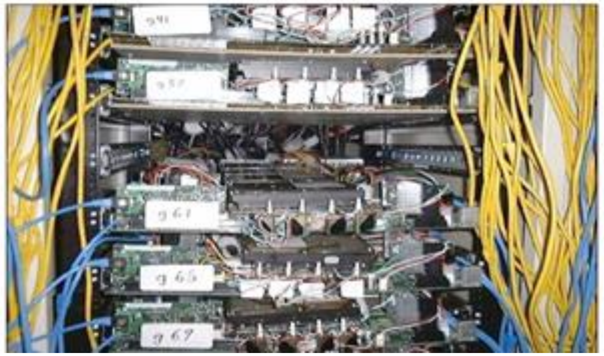
A Single Google Data Center now…

[CyrusOne Middenmeer, Netherlands]

*Each Data center is now easily equivalent to a 40-50 football stadium in size, consuming more than 100 megawatt s power

A Global View of Google Data Centers …

"Corkboards" (1999)



Google Data Center (2000)

Google Cloud regions. With around ~ 40 geographically distributed data centres [as of 2024]

Source: https://cloud.google.com/about/locations

Scaling Techniques

- Bigger machines
- Virtualization
- Asynchronous communication
- Replication & Caching
- Partitioning

➢Virtualization