
# Data Engineering: Introduction

# Computers

# Mark I

**Release date:** August 7, 1944; 77 years ago

**Power:** 5 horsepower (3.7 kW)

**Dimensions:** 816 cubic feet (23 $m^3$) – 51 feet (16 m) in length, 8 feet (2.4 m) in height, and 2 feet (0.61 m) deep

**Mass:** 9,445 pounds (4.7 short tons; 4.3 t)

- Store 72 numbers, each 23 decimal digits long
- Do 3 additions or subtractions in a second
- Multiplication took 6 seconds
- Division took 15.3 seconds

# Fagaku

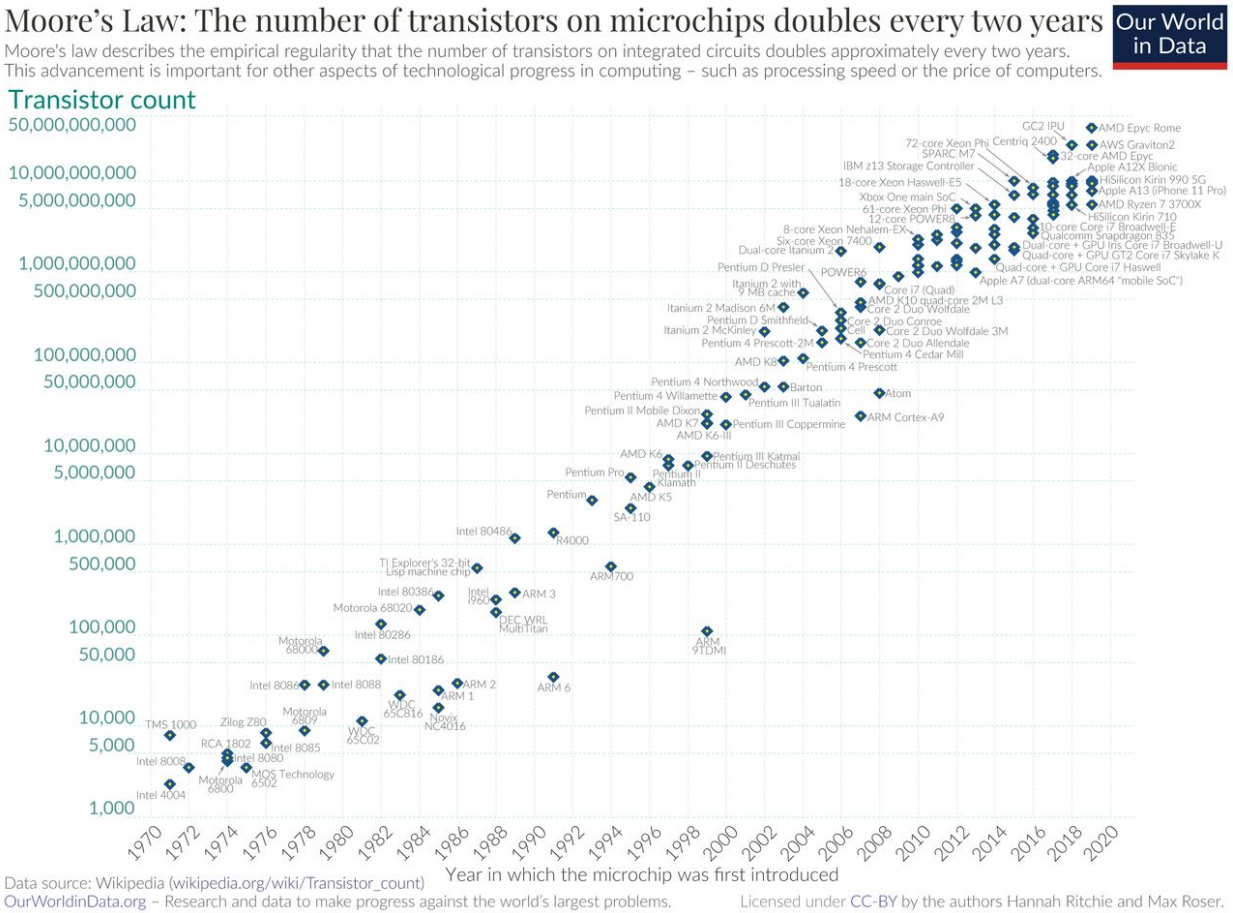
**Memory:** HBM2 32 GiB/node

**Storage:** 1.6 TB NVMe SSD/16 nodes (L1)

- 150 PB shared Lustre FS (L2)
- Cloud storage services (L3)

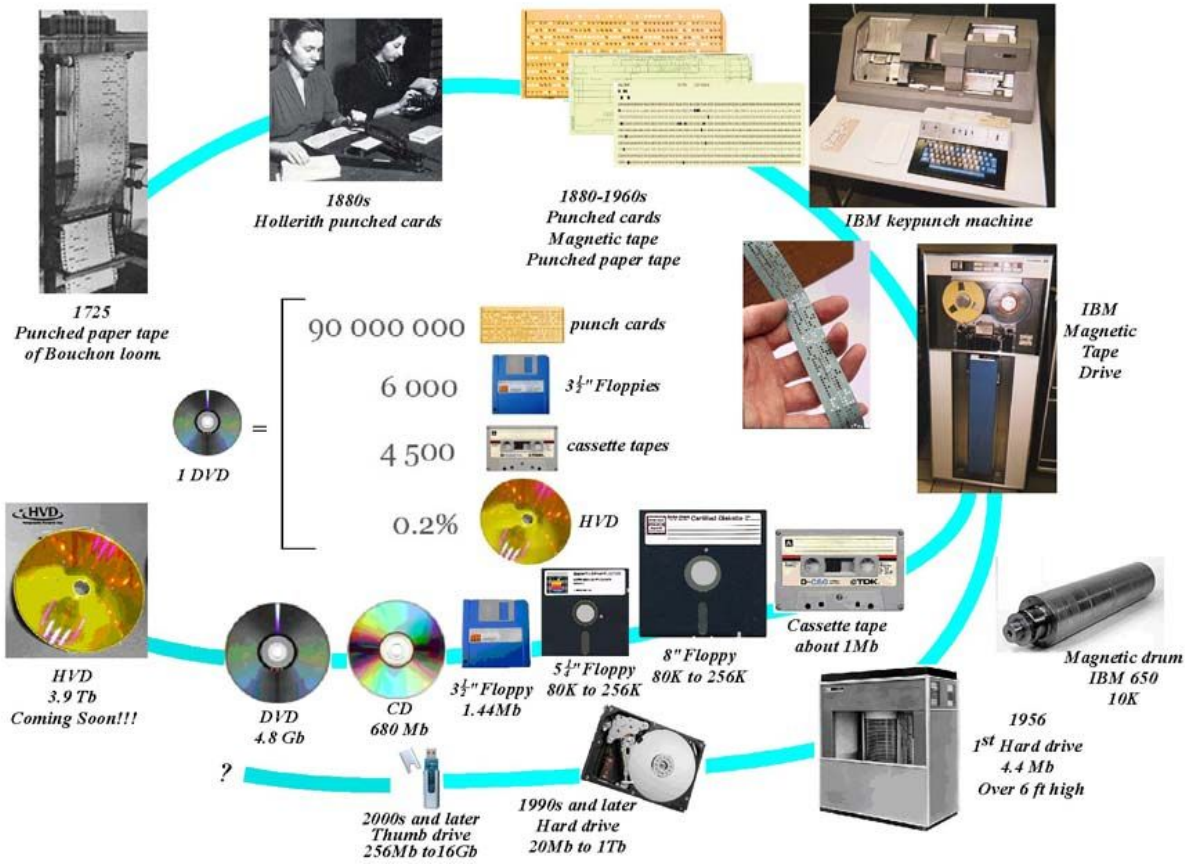
**Speed:** 442 PFLOPS

**Cost:** US$1 billion (total programme cost)


Moore's Law: The number of transistors on microchips doubles every two years
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years.
This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.
Our World in Data
Transistor count
50,000,000,000
10,000,000,000
5,000,000,000
1,000,000,000
500,000,000
100,000,000
50,000,000
10,000,000
5,000,000
1,000,000
500,000
100,000
50,000
10,000
5,000
1,000
1970
1972
1974
1976
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
Year in which the microchip was first introduced
Intel 4004
Intel 8008
Intel 8080
Intel 8085
MOS Technology 6502
Motorola 6800
Motorola 6809
Zilog Z80
RCA 1802
TMS 1000
Intel 8086
Intel 8088
Intel 80186
Intel 80286
Intel 80386
Motorola 68020
Motorola 68000
ARM 1
ARM 2
ARM 3
ARM 6
ARM 7TDMI
ARM700
WDC 65C816
WDC 65C02
Novix NC4016
Intel 80486
R4000
SA-110
AMD K5
Pentium
Pentium Pro
Pentium II Klamath
Pentium II Deschutes
AMD K6
AMD K6-III
Pentium III Katmai
Pentium 4 Willamette
Pentium 4 Northwood
Pentium II Mobile Dixon
AMD K7
Pentium III Coppermine
Pentium III Tualatin
Barton
Atom
ARM Cortex-A9
AMD K8
Pentium 4 Prescott
Pentium 4 Cedar Mill
Pentium 4 Prescott-2M
Itanium 2 McKinley
Cell
Core 2 Duo Wolfdale 3M
Core 2 Duo Allendale
Itanium 2 Madison 6M
Pentium D Smithfield
Core 2 Duo Conroe
Itanium 2 with 9 MB cache
Core 2 Duo Wolfdale
AMD K10 quad-core 2M L3
Core i7 (Quad)
Pentium D Presler
POWER6
Dual-core Itanium 2
Six-core Xeon 7400
8-core Xeon Nehalem-EX
12-core POWER8
61-core Xeon Phi
Xbox One main SoC
18-core Xeon Haswell-E5
IBM z13 Storage Controller
SPARC M7
72-core Xeon Phi
GC2 IPU
Centriq 2400
32-core AMD Epyc
AWS Graviton2
AMD Epyc Rome
Apple A12X Bionic
HiSilicon Kirin 990 5G
Apple A13 (iPhone 11 Pro)
AMD Ryzen 7 3700X
HiSilicon Kirin 710
10-core Core i7 Broadwell-E
Qualcomm Snapdragon 835
Dual-core + GPU Iris Core i7 Broadwell-U
Quad-core + GPU GT2 Core i7 Skylake K
Quad-core + GPU Core i7 Haswell
Apple A7 (dual-core ARM64 "mobile SoC")
TI Explorer's 32-bit Lisp machine chip
DEC WRL MultiTitan
Intel i960
Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)
OurWorldinData.org – Research and data to make progress against the world's largest problems.
Licensed under CC-BY by the authors Hannah Ritchie and Max Roser.



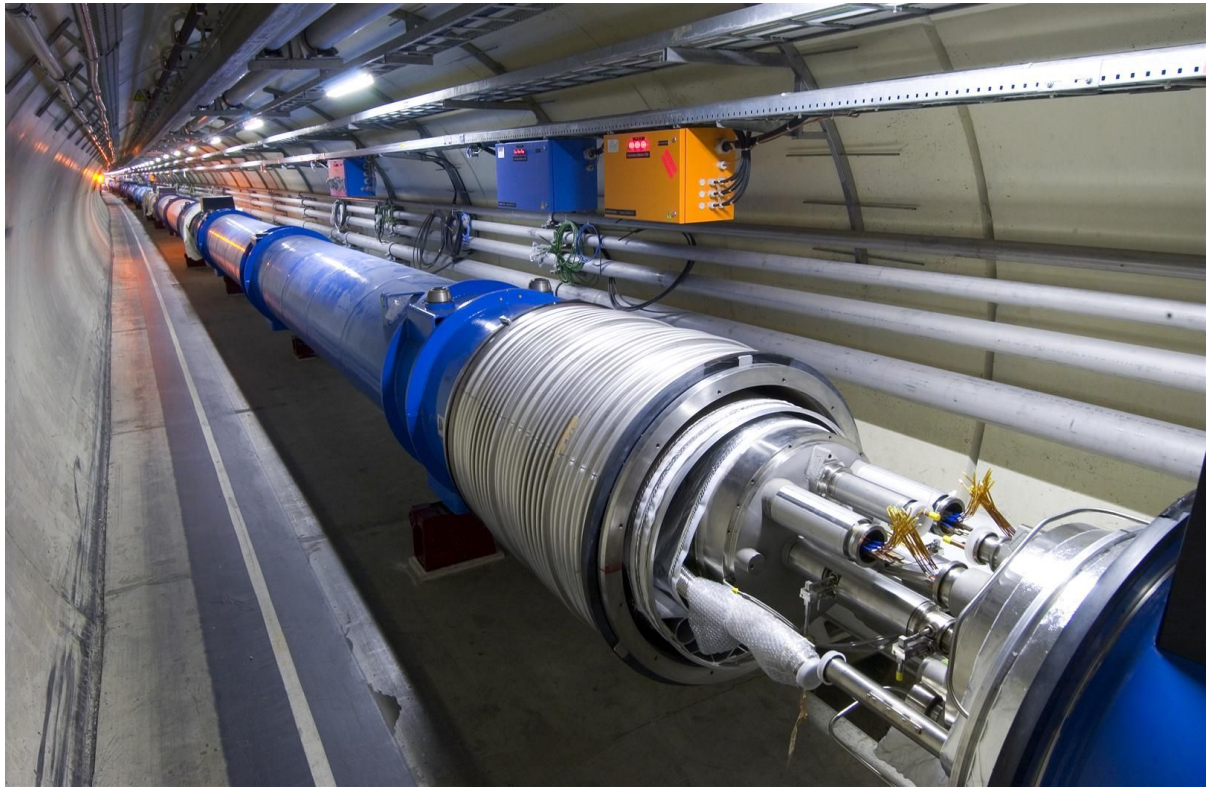
Wikipedia


1725
Punched paper tape
of Bouchon loom.
1880s
Hollerith punched cards
1880-1960s
Punched cards
Magnetic tape
Punched paper tape
IBM keypunch machine
IBM
Magnetic
Tape
Drive
1 DVD
=
90 000 000
punch cards
6 000
3½" Floppies
4 500
cassette tapes
0.2%
HVD
HVD
3.9 Tb
Coming Soon!!!
DVD
4.8 Gb
CD
680 Mb
3½" Floppy
1.44Mb
5¼" Floppy
80K to 256K
8" Floppy
80K to 256K
Cassette tape
about 1Mb
Magnetic drum
IBM 650
10K
1956
1st Hard drive
4.4 Mb
Over 6 ft high
?
2000s and later
Thumb drive
256Mb to16Gb
1990s and later
Hard drive
20Mb to 1Tb

# CERN

- 50 PB/Year
- Storage: 1 ZB
- 15000 servers

https://monit-grafana-open.cern.ch/d/000000884/it-overview?orgId=16

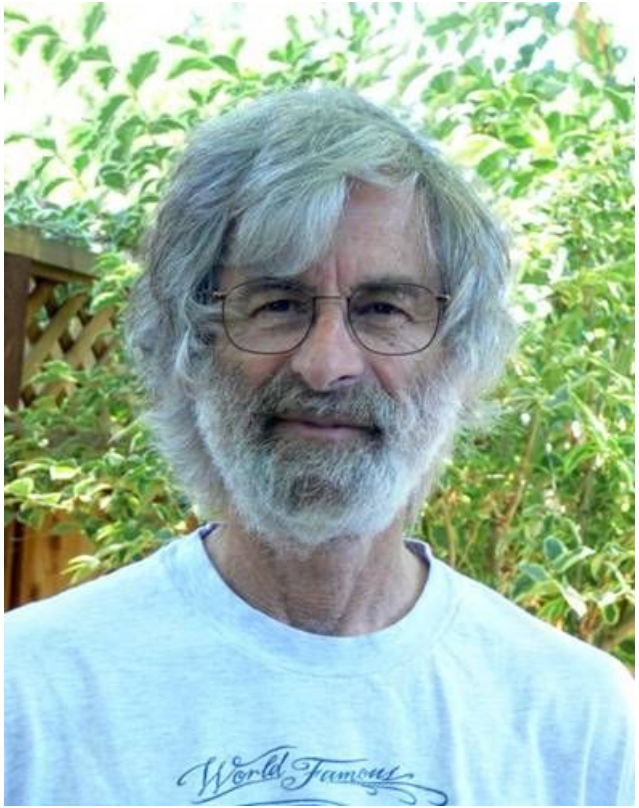


# Why make a system distributed?

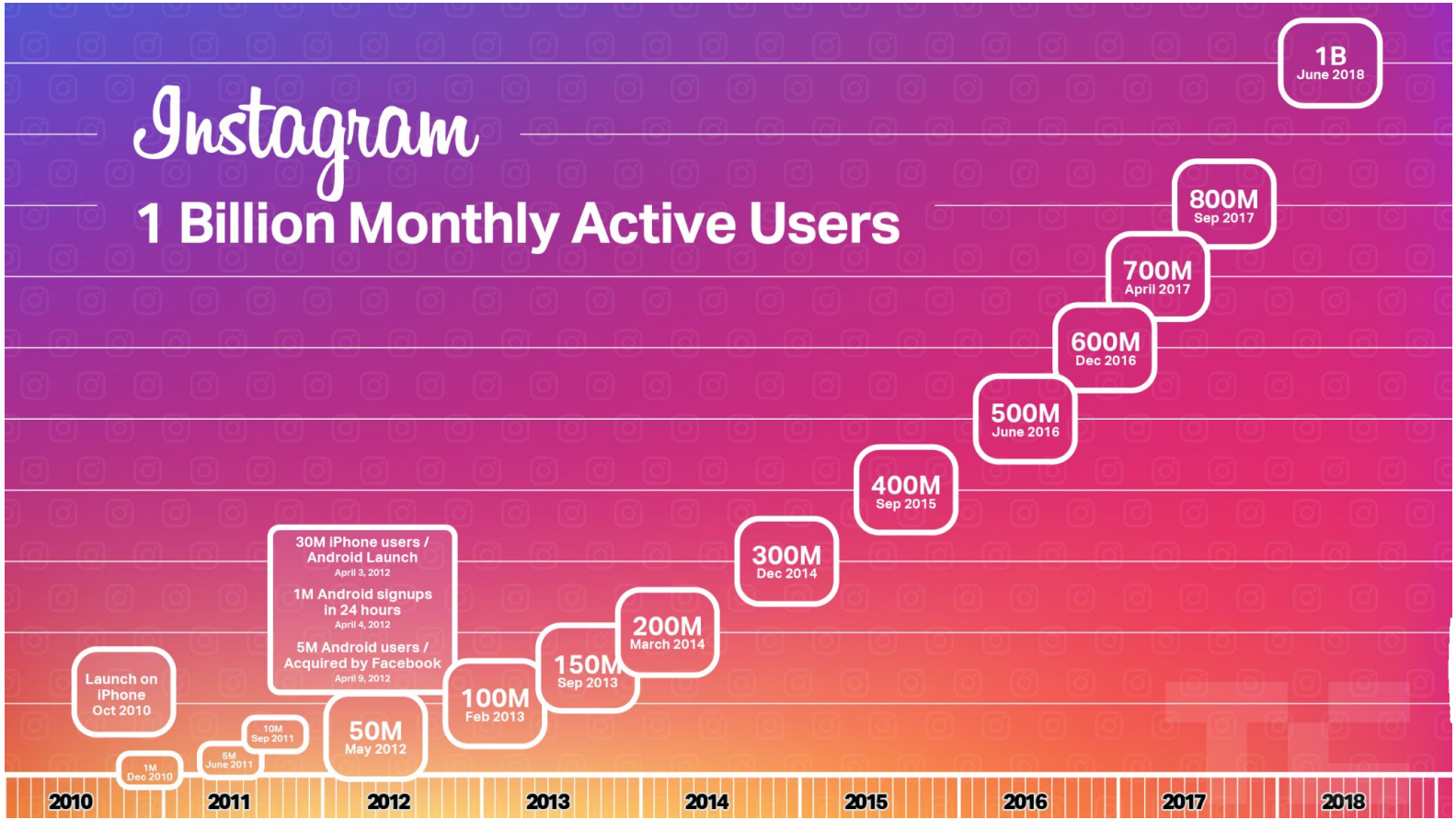
- It’s inherently distributed
- For better reliability
- For better performance
- To solve bigger problems

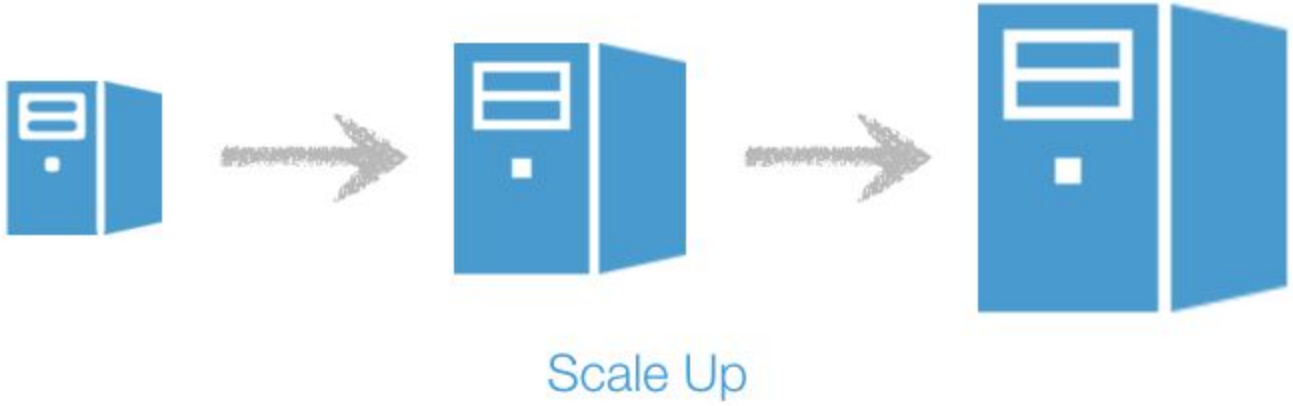
# Distributed System

“. . . a system in which the failure of a computer you didn’t even know existed can render your own computer unusable.” — Leslie Lamport




# Scalability

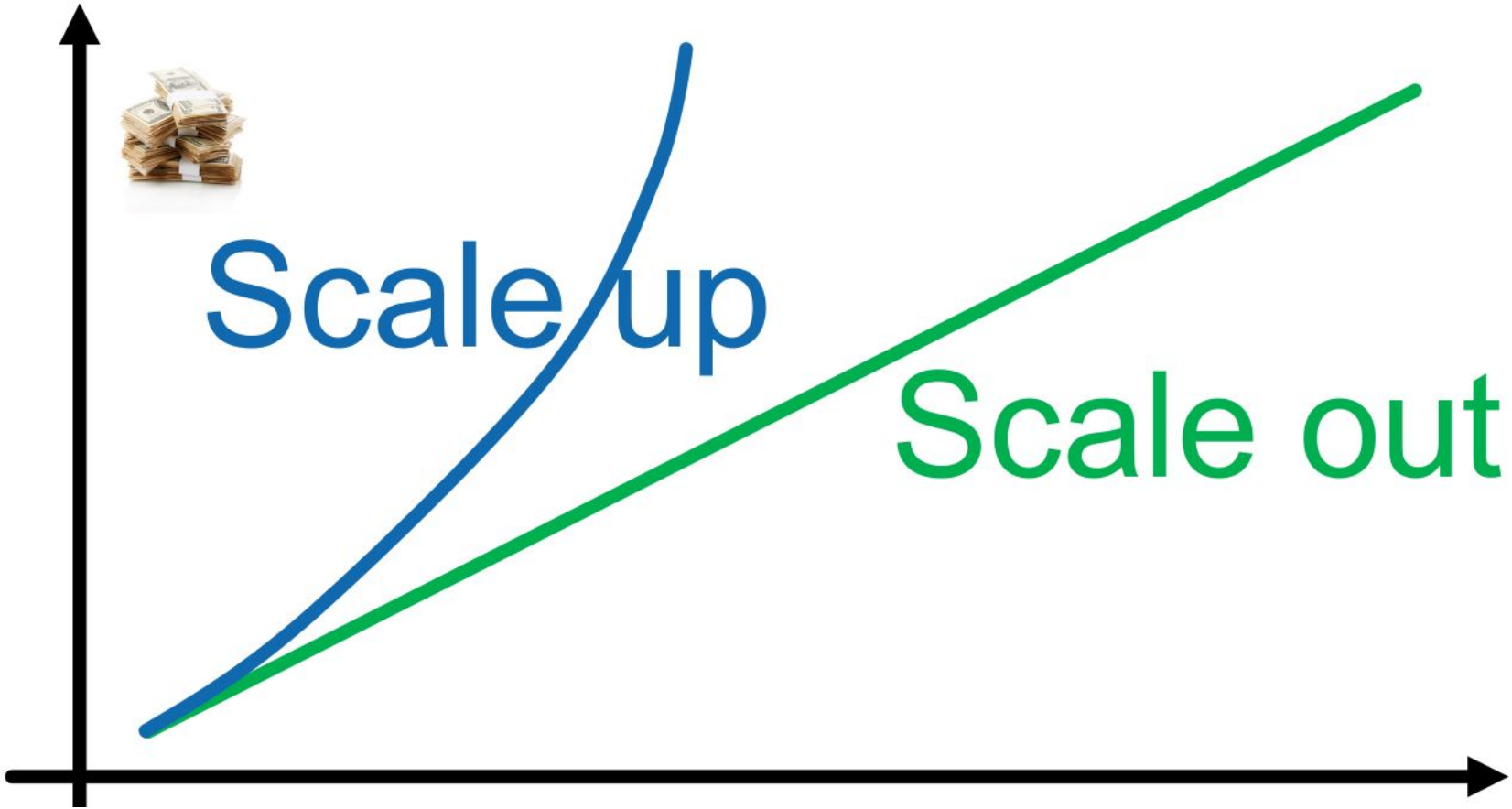

Instagram
1 Billion Monthly Active Users
1B
June 2018
800M
Sep 2017
700M
April 2017
600M
Dec 2016
500M
June 2016
400M
Sep 2015
300M
Dec 2014
200M
March 2014
150M
Sep 2013
100M
Feb 2013
50M
May 2012
30M iPhone users / Android Launch
April 3, 2012
1M Android signups in 24 hours
April 4, 2012
5M Android users / Acquired by Facebook
April 9, 2012
Launch on iPhone
Oct 2010
1M
Dec 2010
5M
June 2011
10M
Sep 2011
2010
2011
2012
2013
2014
2015
2016
2017
2018



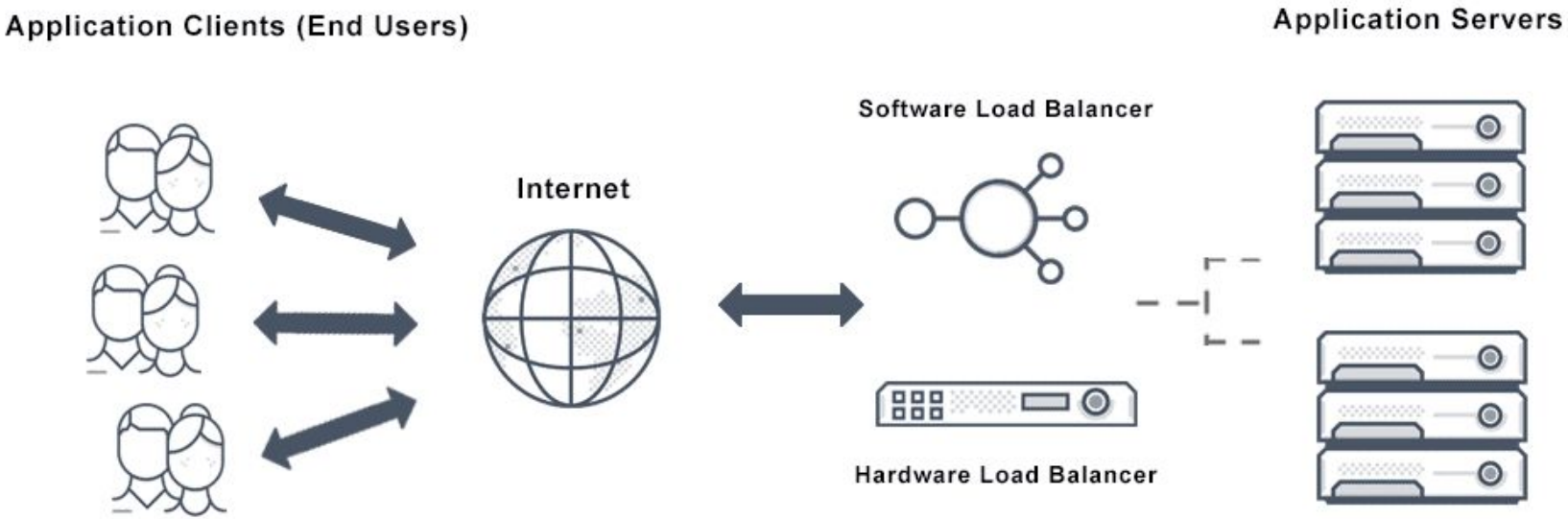


Source

# Hardware price comparison

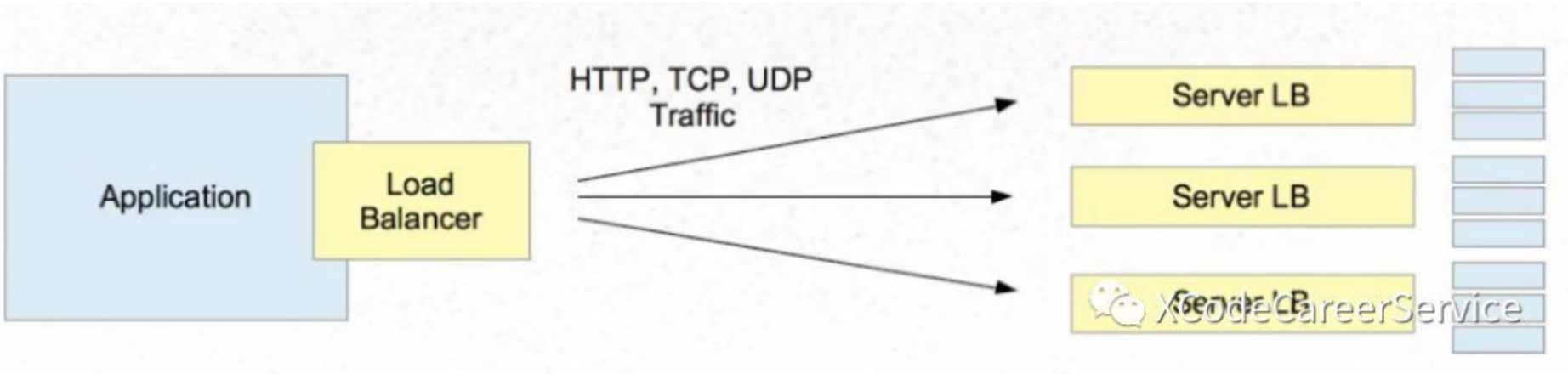


# What's the problems?



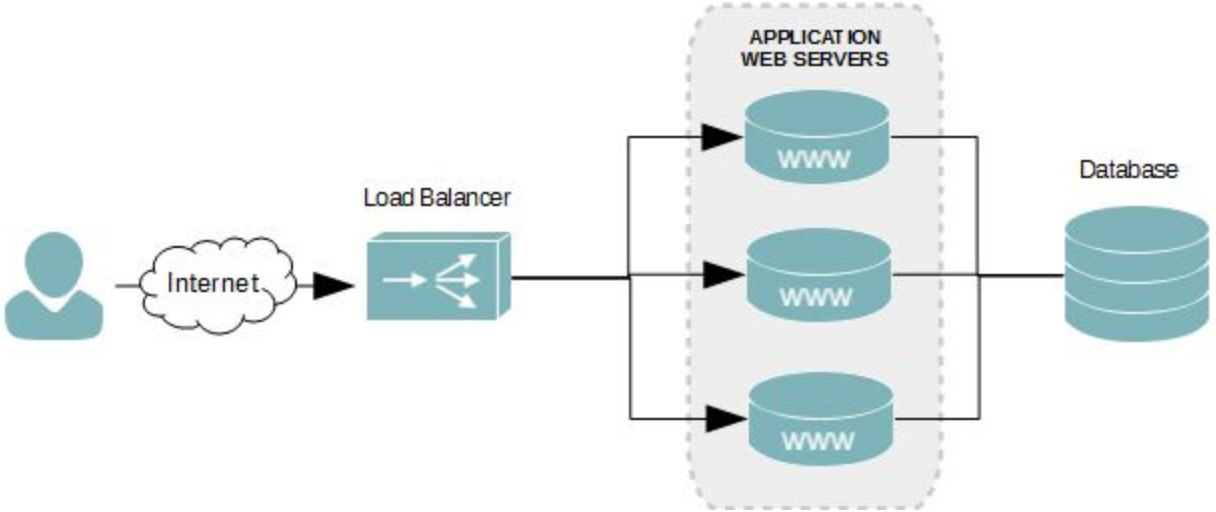

Source

# Smart Client-Side Load Balancing



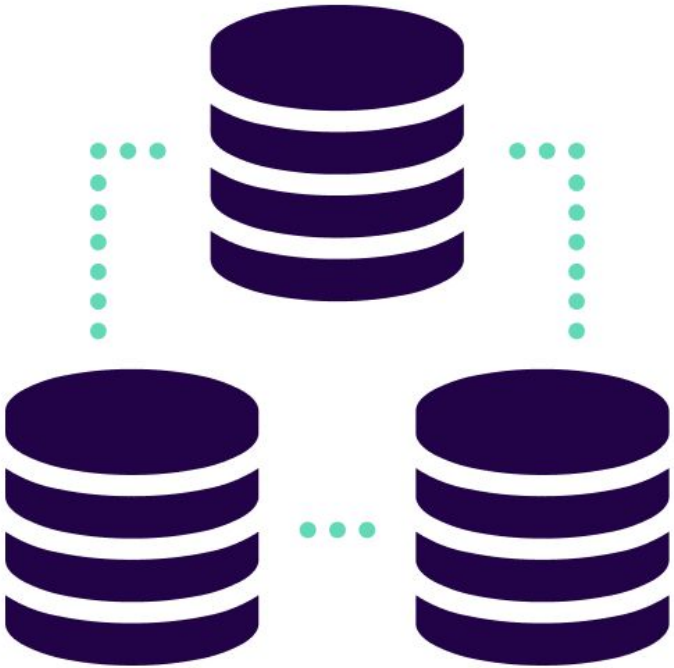

Source

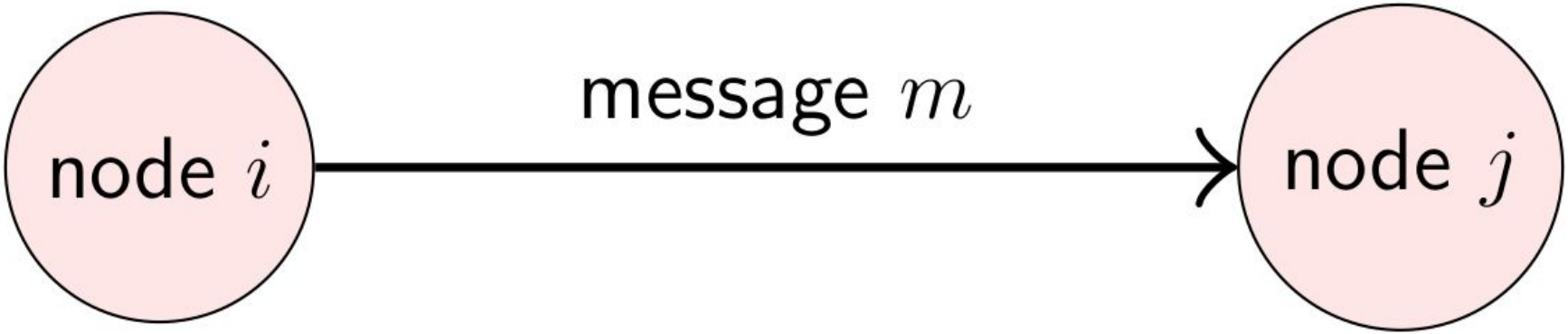
# Shared Database

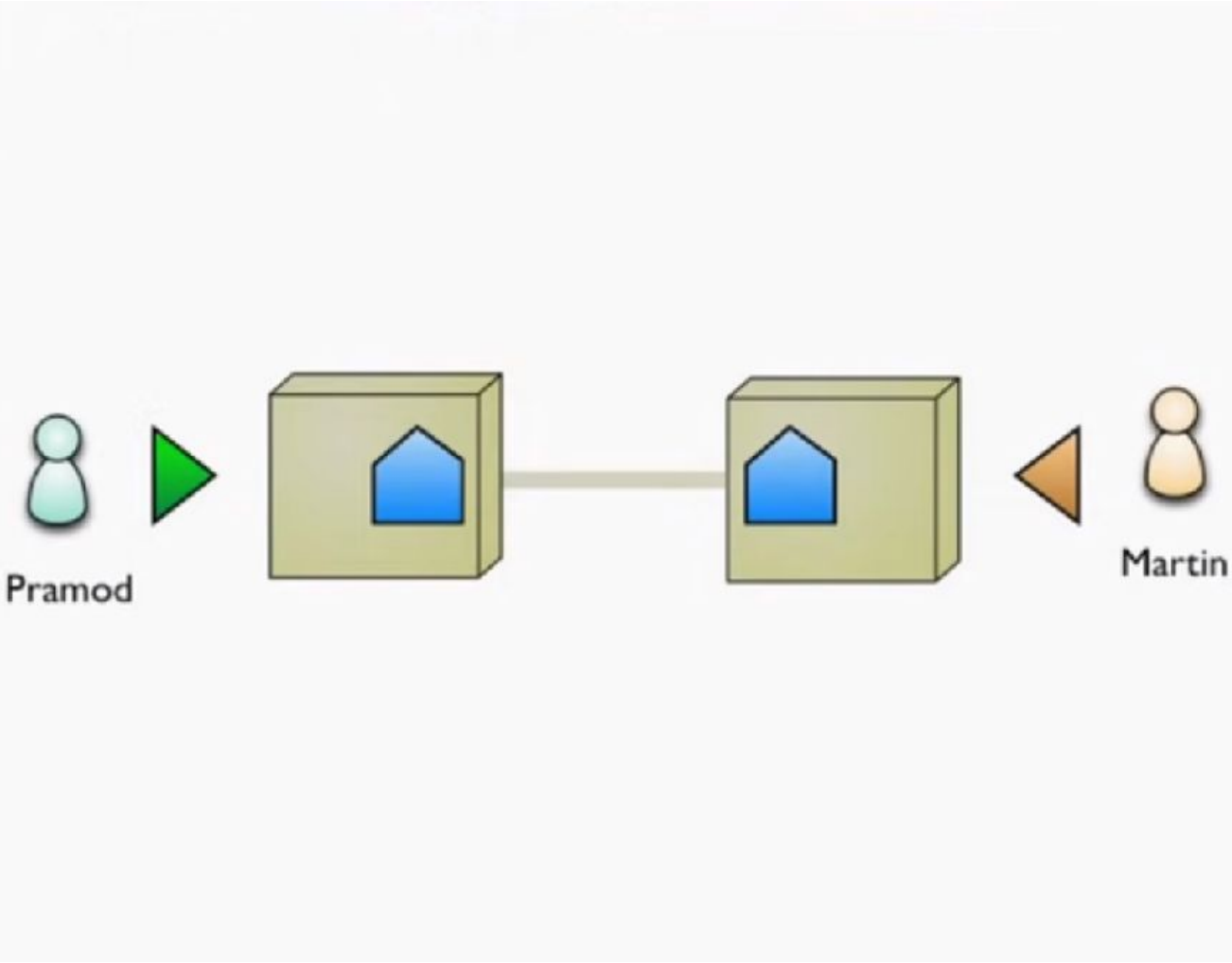


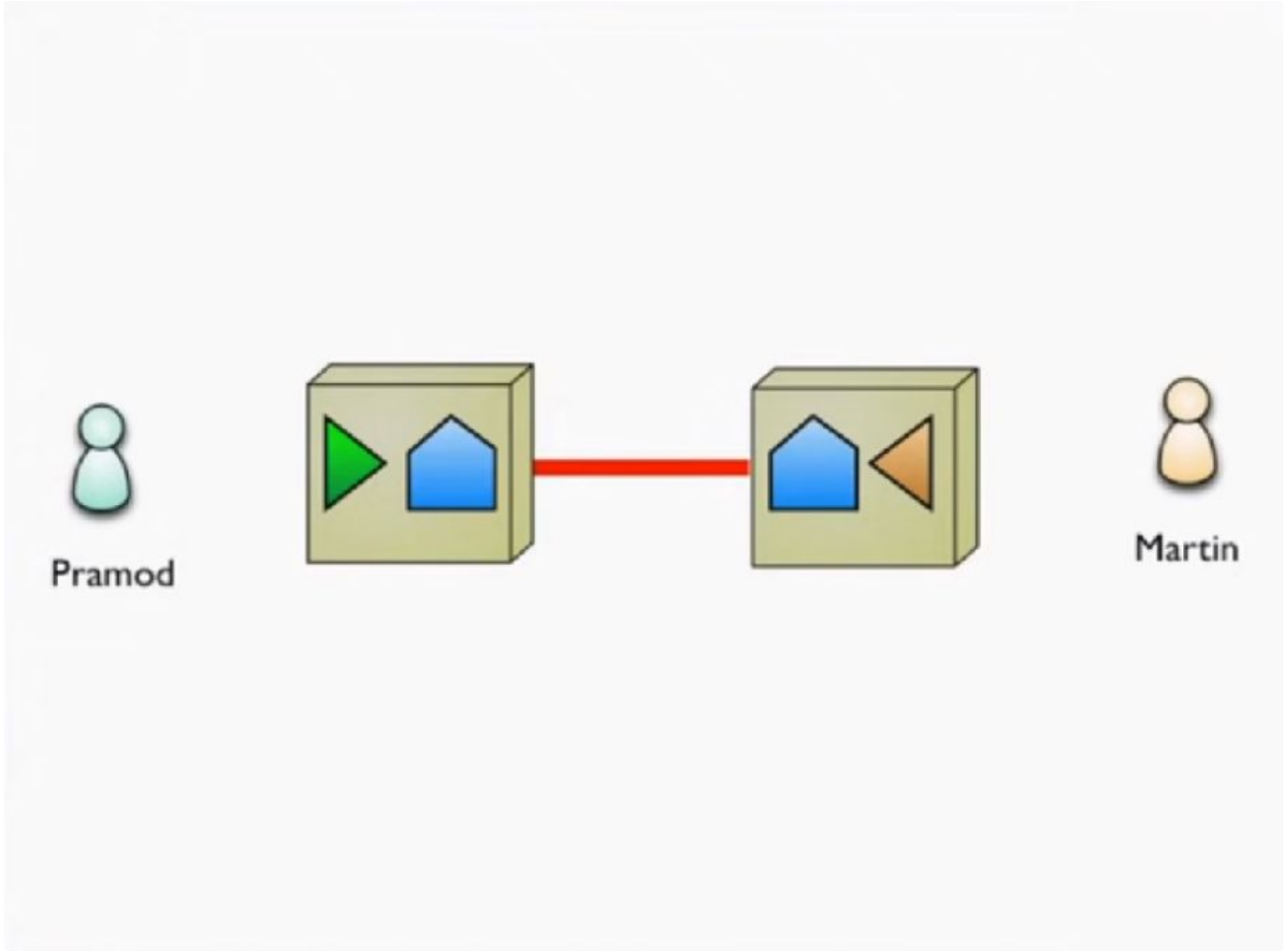

Source

- Sharding
- Replication

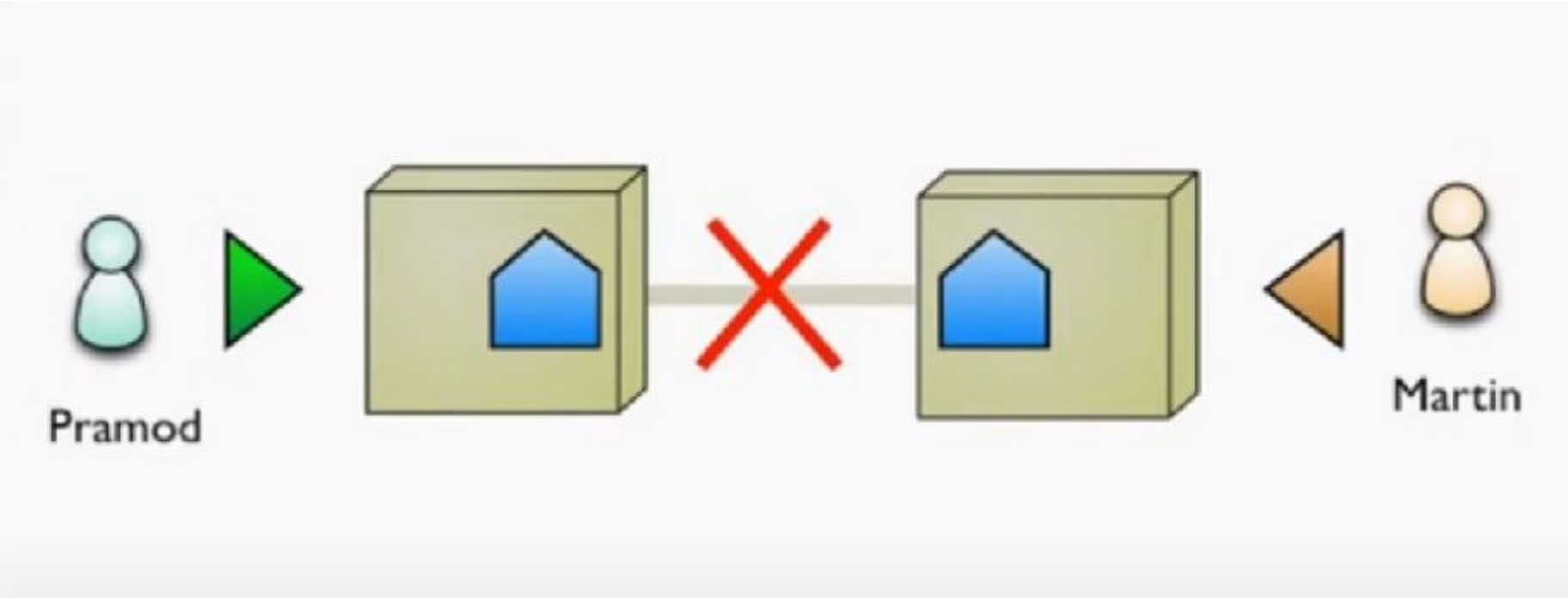


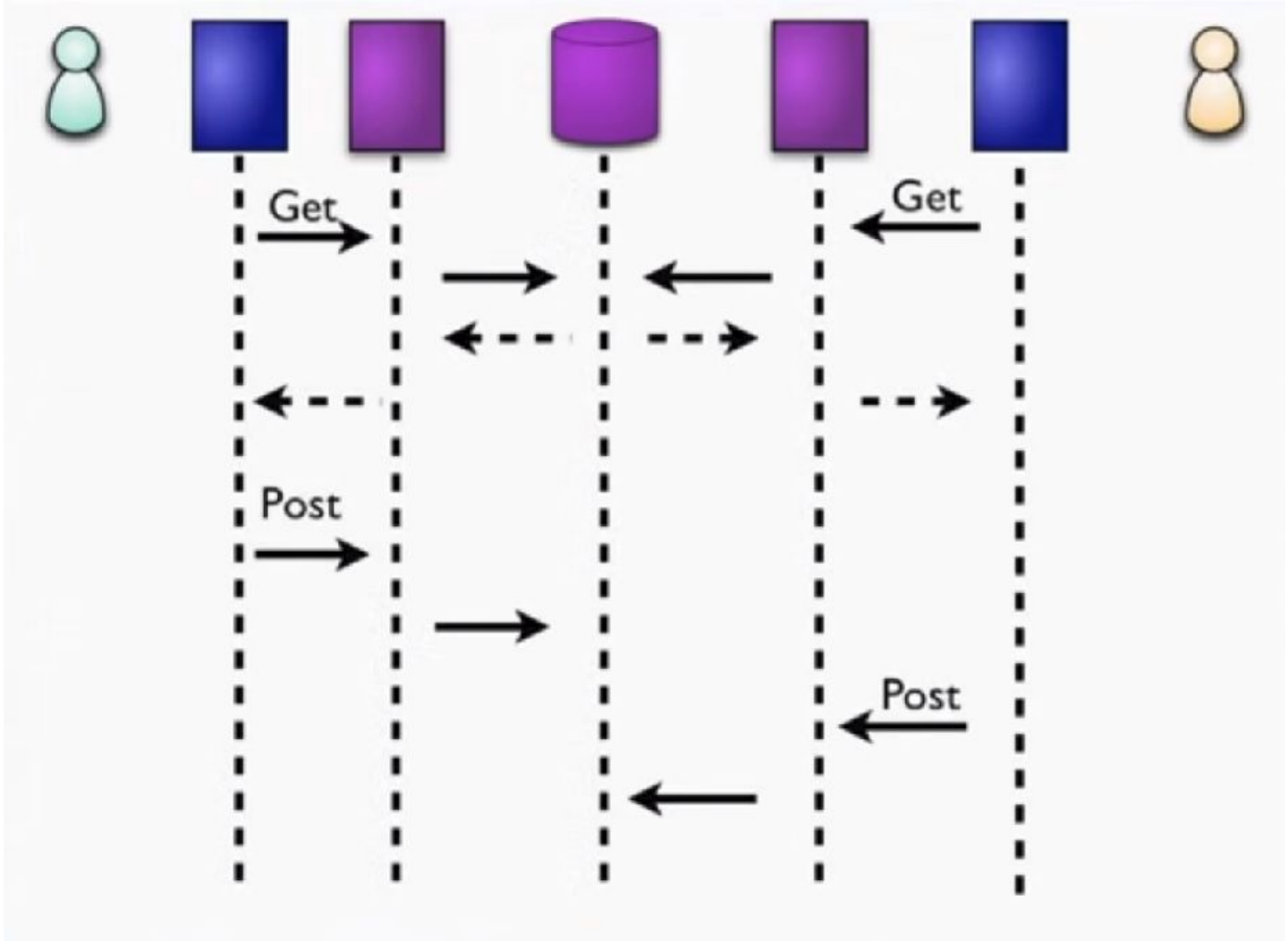

node $i$
message $m$
node $j$


Pramod
Martin

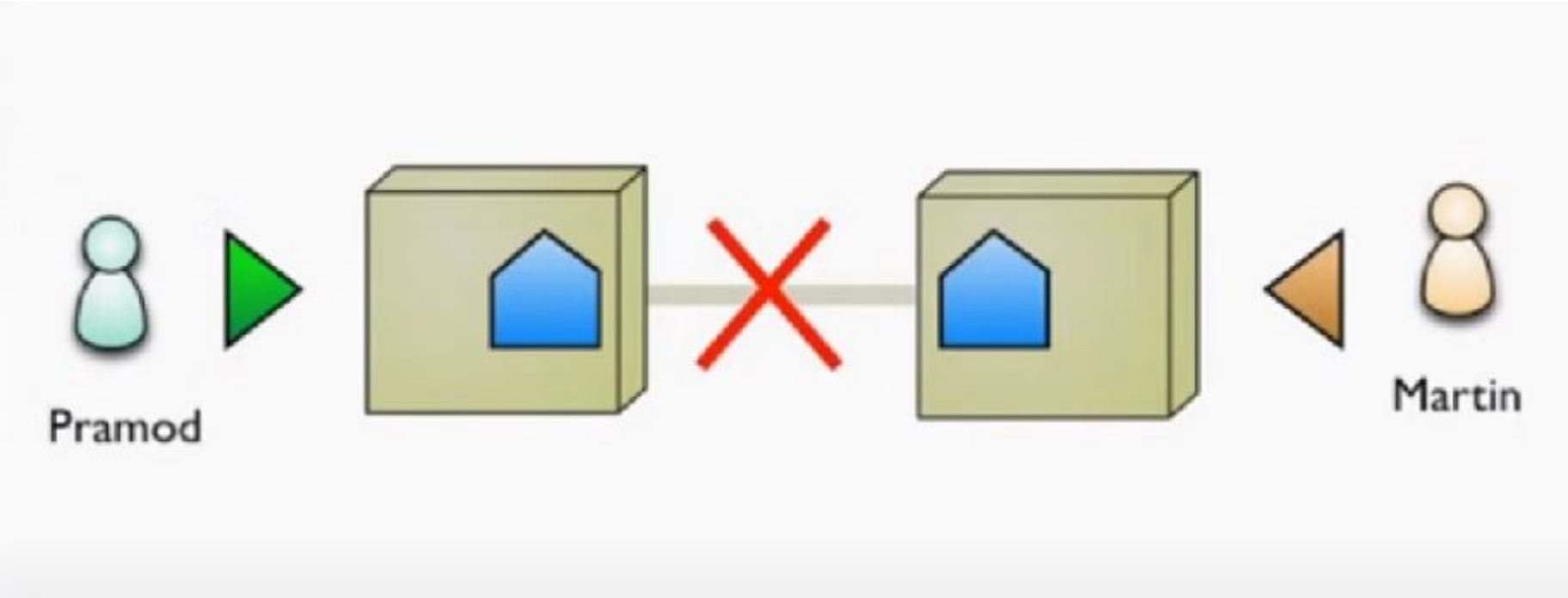

Pramod
Martin

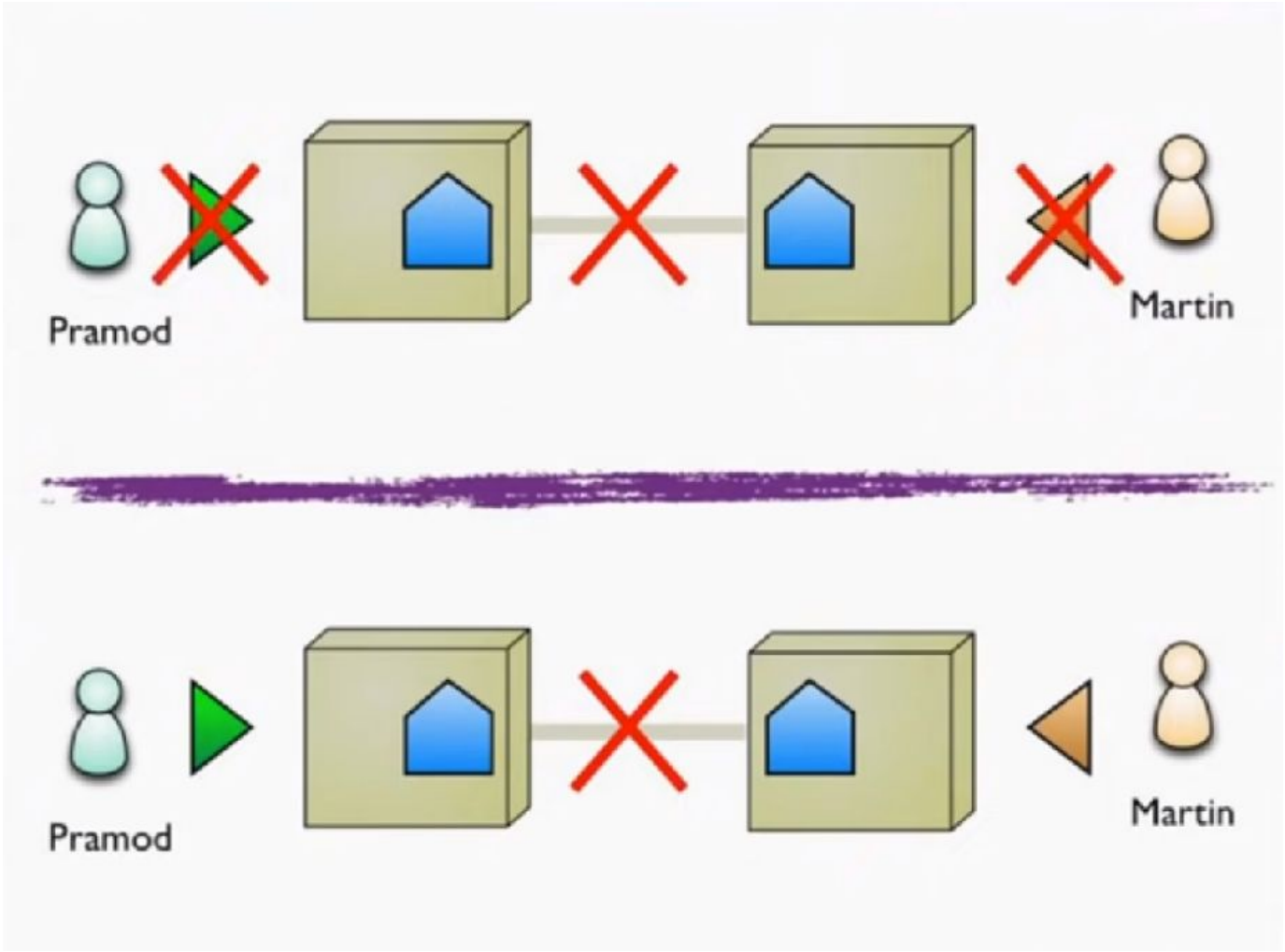
# Partition Tolerance

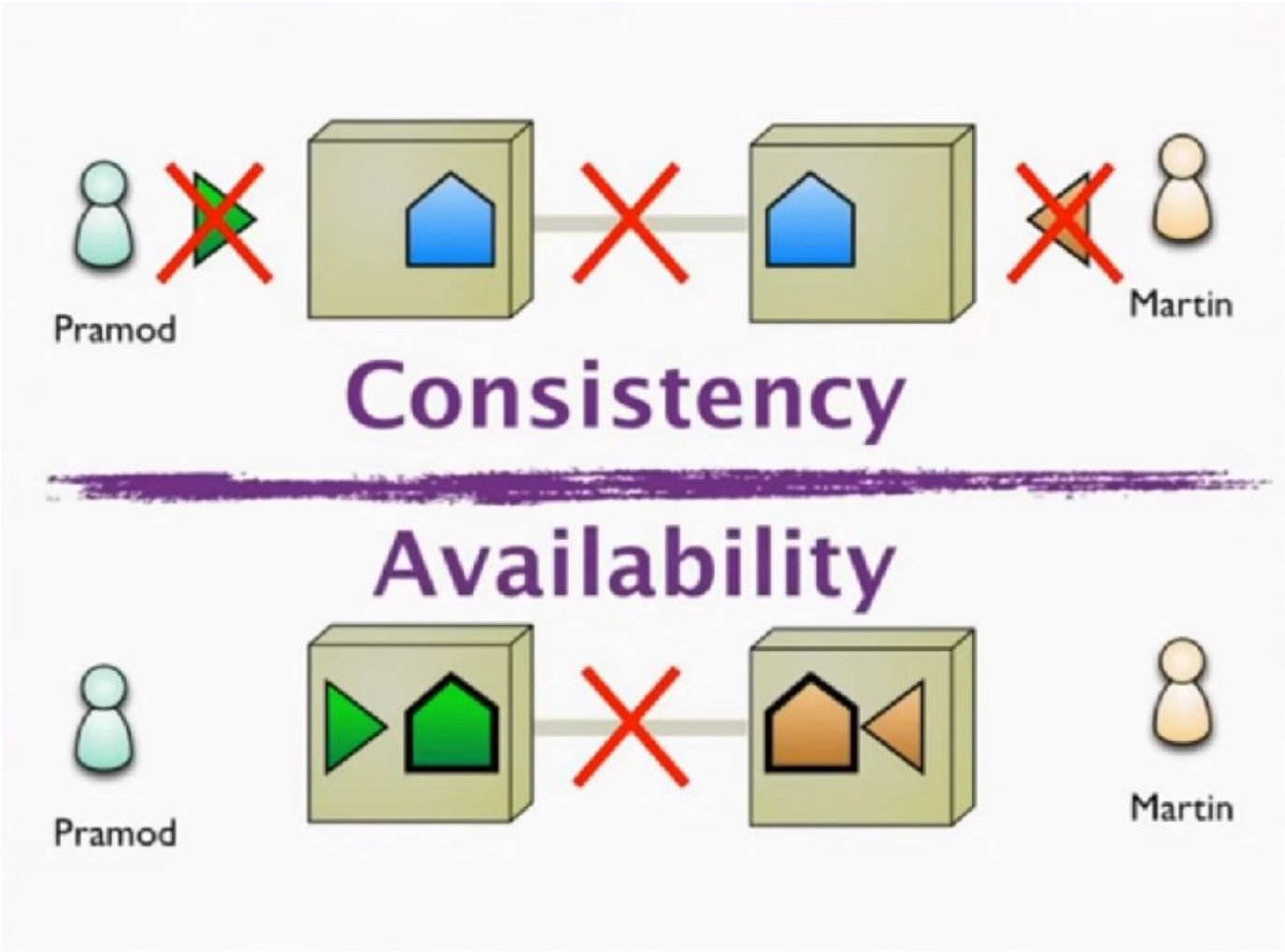



Get
Get
Post
Post

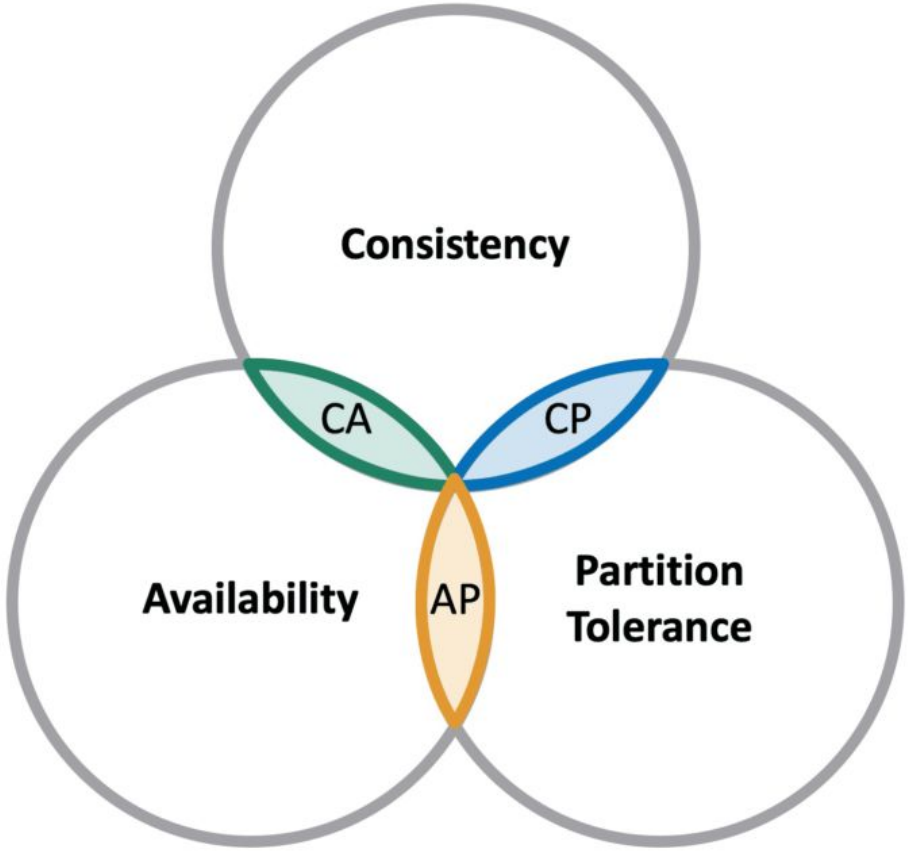
# Partition Tolerance




Pramod
Martin
Pramod
Martin


Pramod
Martin
Consistency
Availability
Pramod
Martin

# Cap Theorem



# Assumptions

- The network is reliable
- The network is secure
- The network is homogeneous
- The topology does not change
- Latency is zero
- Bandwidth is infinite
- Transport cost is zero
- There is one administrator

# An Example:
# Distributed Cache System

HDD


CrystalDiskMark 3.0.1 x64
File Edit Theme Help Language
All
5 1000MB C: 13% (88/699GB)
Read [MB/s] Write [MB/s]
Seq 112.3 109.3
512K 41.69 48.05
4K 0.543 0.693
4K QD32 1.004 0.698

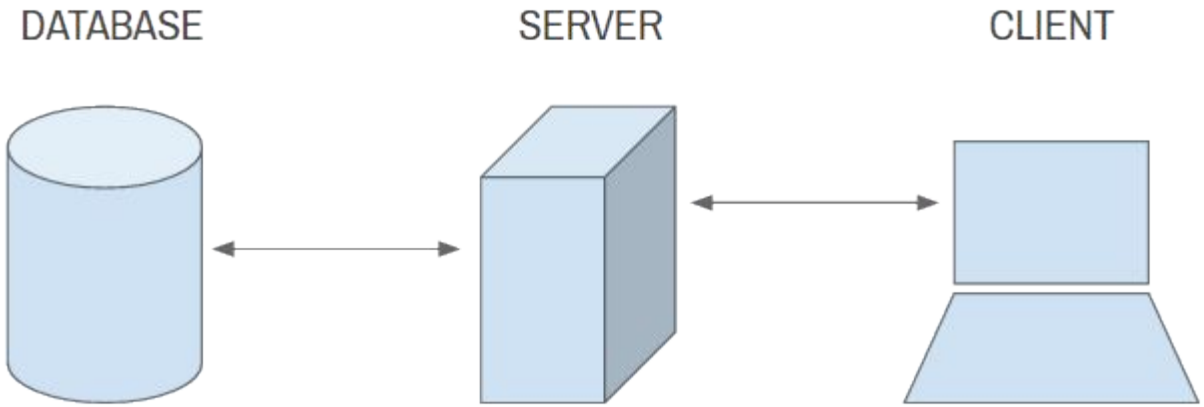
SSD


CrystalDiskMark 3.0.1 x64
File Edit Theme Help Language
All
5 1000MB C: 45% (101/224GB)
Read [MB/s] Write [MB/s]
Seq 477.9 235.7
512K 402.7 248.9
4K 30.49 64.67
4K QD32 200.3 233.2

RAM

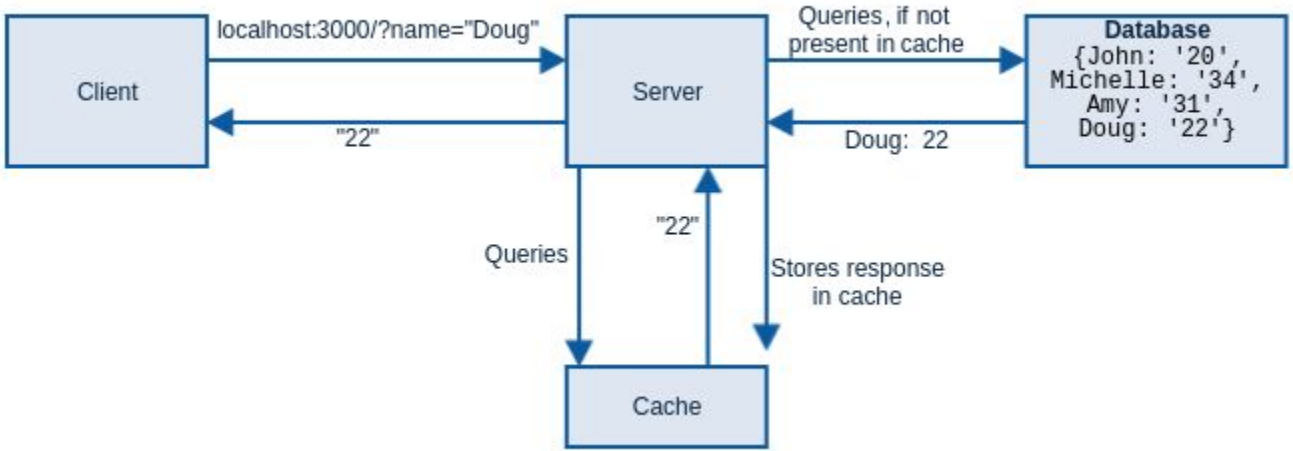

CrystalDiskMark 3.0.1 x64
File Edit Theme Help Language
All
5 1000MB R: 1% (48/4089MB)
Read [MB/s] Write [MB/s]
Seq 5766 7760
512K 5649 7172
4K 657.0 554.8
4K QD32 631.9 544.7

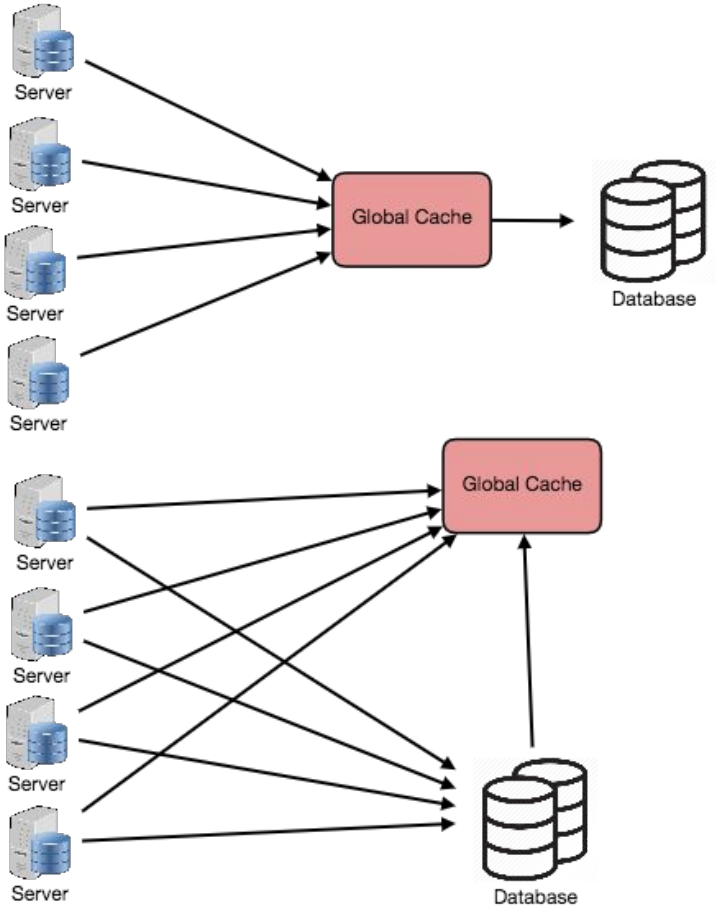
# Latency




# A Cache is like short-term memory.

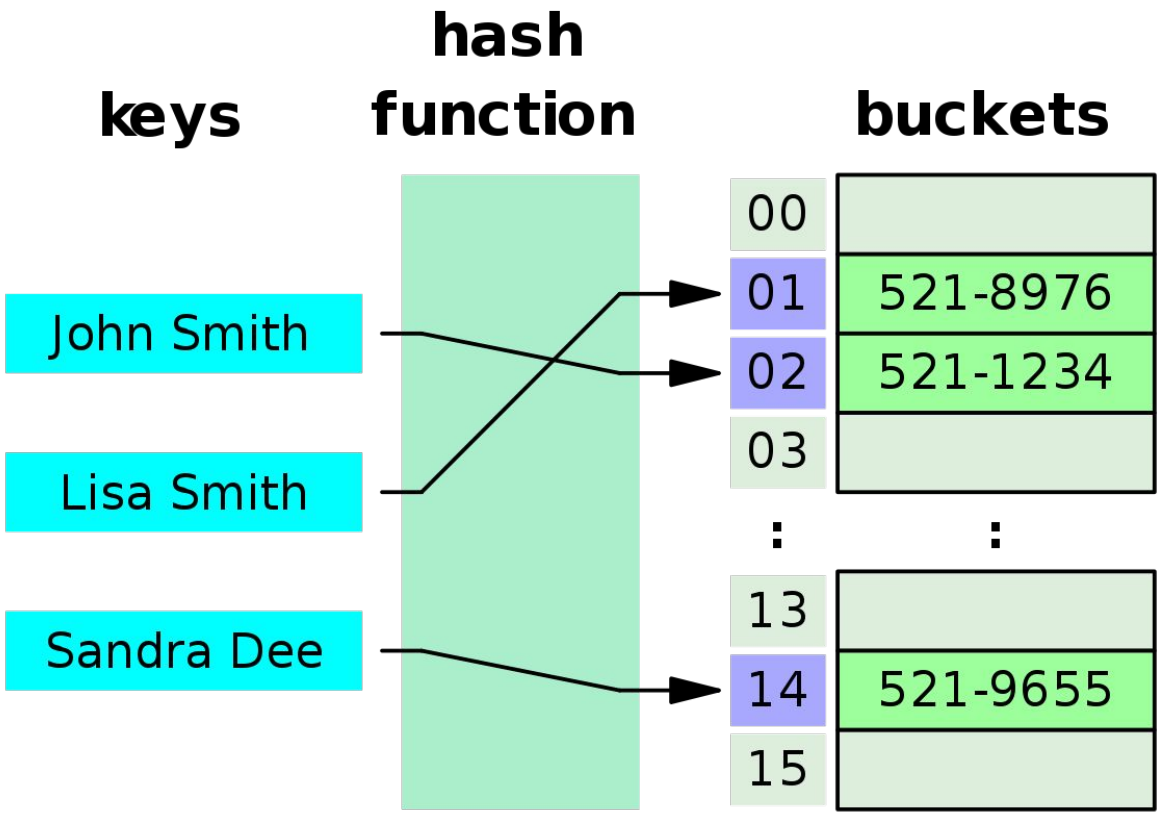

Client
Makes request to
Returns response to
Server
Queries
Returns results to
Database


Client
localhost:3000/?name="Doug"
"22"
Server
Queries, if not present in cache
Doug: 22
Database
{John: '20',
Michelle: '34',
Amy: '31',
Doug: '22'}
Queries
"22"
Stores response in cache
Cache


Server
Server
Server
Server
Global Cache
Database
Server
Server
Server
Server
Global Cache
Database

# Hash Table

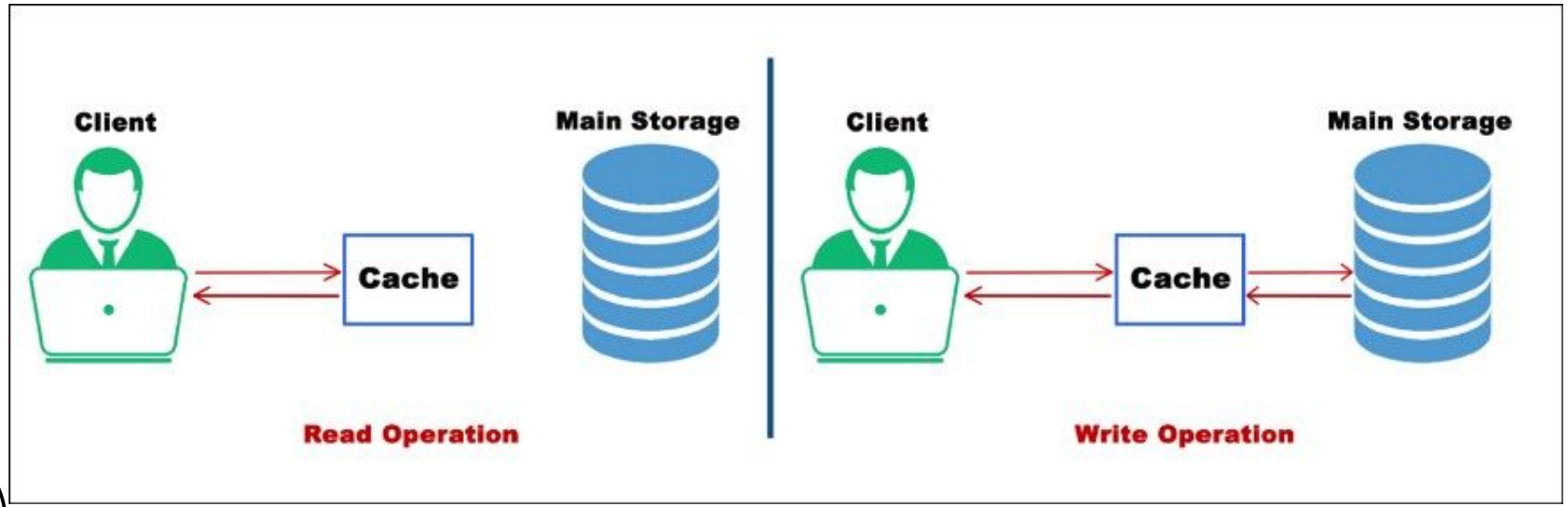
## Data structure used to implement cache


keys
hash function
buckets
John Smith
Lisa Smith
Sandra Dee
00
01
02
03
13
14
15
521-8976
521-1234
521-9655

# Cache Invalidation and Modification

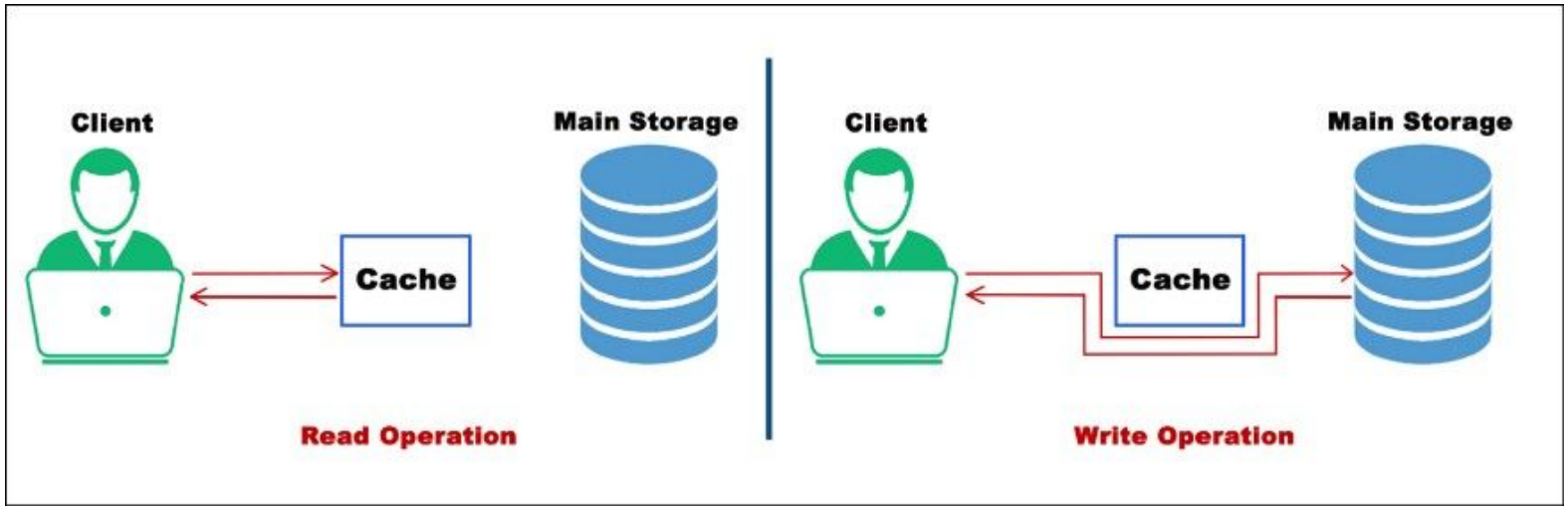
# Read-Through

The data is written into the cache and the corresponding database at the same time. This scheme maintains the complete data ***consistency*** between the cache and the main storage.



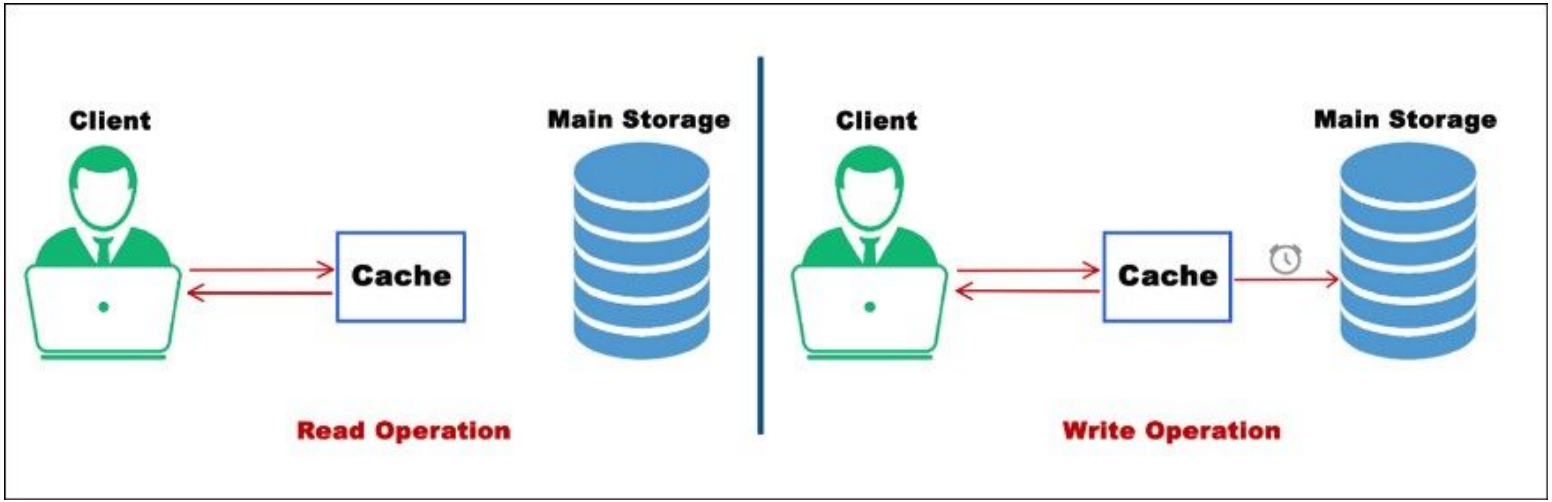

# Write-Through

In this strategy, every information directly written to the database just bypassing the cache.




# Write-Back

The data is written to cache only and completion is immediately confirmed to the client. The write to the permanent storage is done after specified intervals or under certain conditions. This results in low latency and high throughput for write-intensive applications.




# Fault Tolerance

As we know all of our hash table and data are stored in RAM, what happens if there is a power loss, all of our data goes for the toss. This means our cache system is not persistent so to make it persistent we have to do something.

- **Regular interval snapshot**
- **Log reconstruction**

# RAM is smaller than a hard disk!

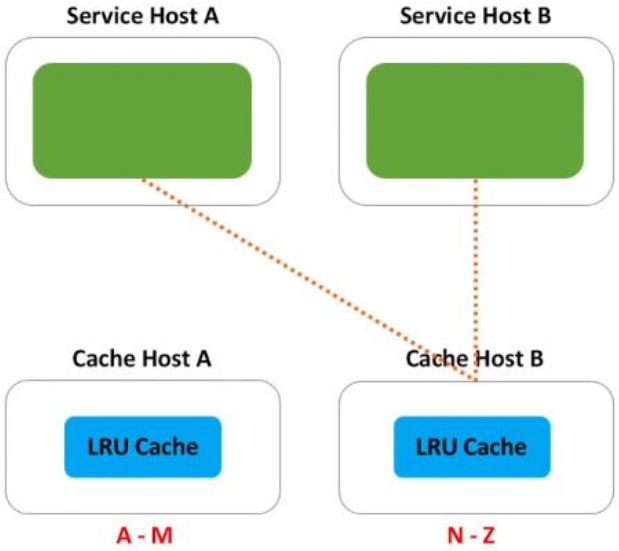
# Cache Eviction Policies

- **First In First Out (FIFO):** The cache evicts the first block accessed first without any regard to how often or how many times it was accessed before.
- **Last In First Out (LIFO):** The cache evicts the block accessed most recently first without any regard to how often or how many times it was accessed before.
- **Least Recently Used (LRU):** Discards the least recently used items first.
- **Most Recently Used (MRU):** Discards, in contrast to LRU, the most recently used items first.
- **Least Frequently Used (LFU):** Counts how often an item is needed. Those that are used least often are discarded first.

# Make it Distributed!

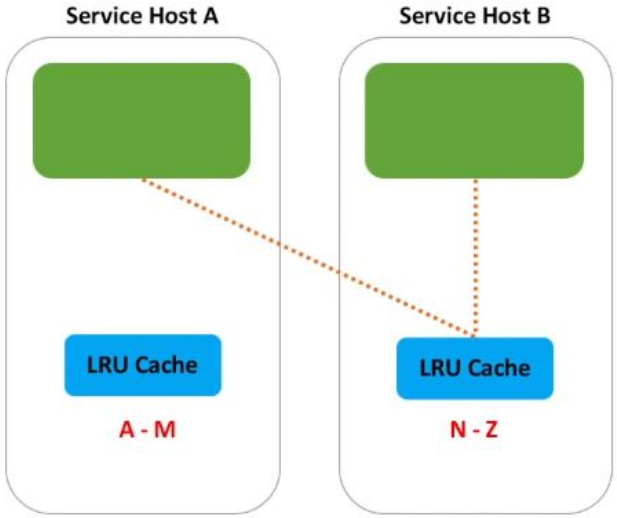
# Distributed Cache

- Scalability
- High Availability
- Fault Tolerance
- Consistency



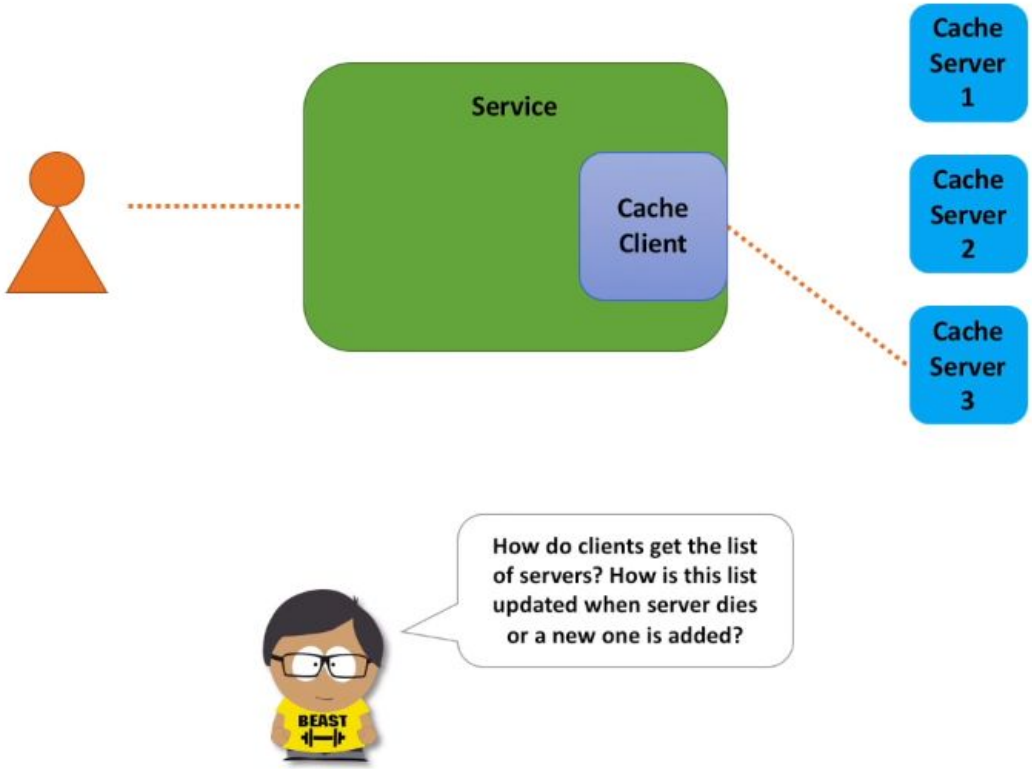

**Dedicated cache cluster**

- Isolation of resources between service and cache
- Can be used by multiple services
- Flexibility in choosing hardware

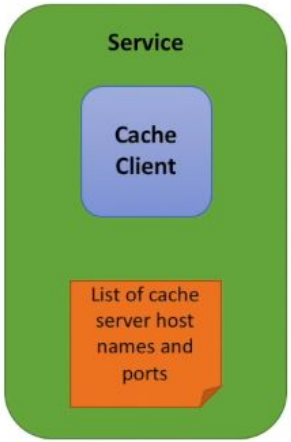



**Co-located cache**

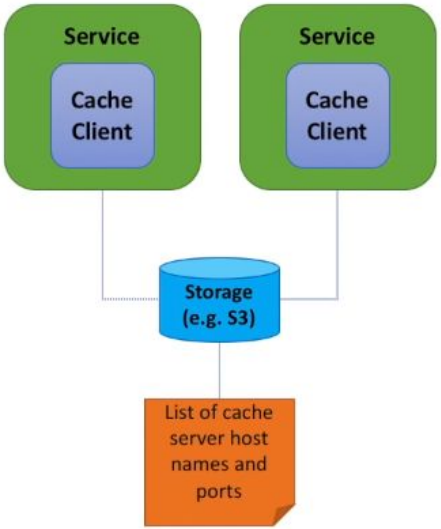
- No extra hardware and operational cost
- Scales together with the service

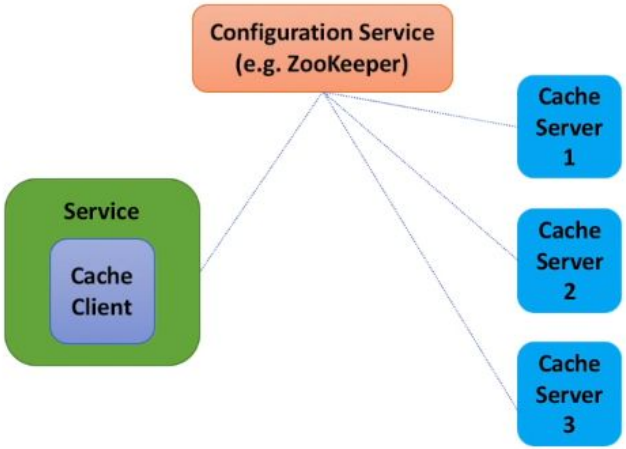



- Cache client knows about all cache servers
- All cache clients should have the same list of servers
- Client stores list of servers in sorted order (by hash value, e.g. TreeMap in Java)
- Binary search is used to identify the server O (log n)
- Cache client uses TCP or UDP protocol to talk to servers
- If server is unavailable, client proceeds as though it was a cache miss




Use configuration management tools (e.g. Chef, Puppet) to deploy modified file to every service host

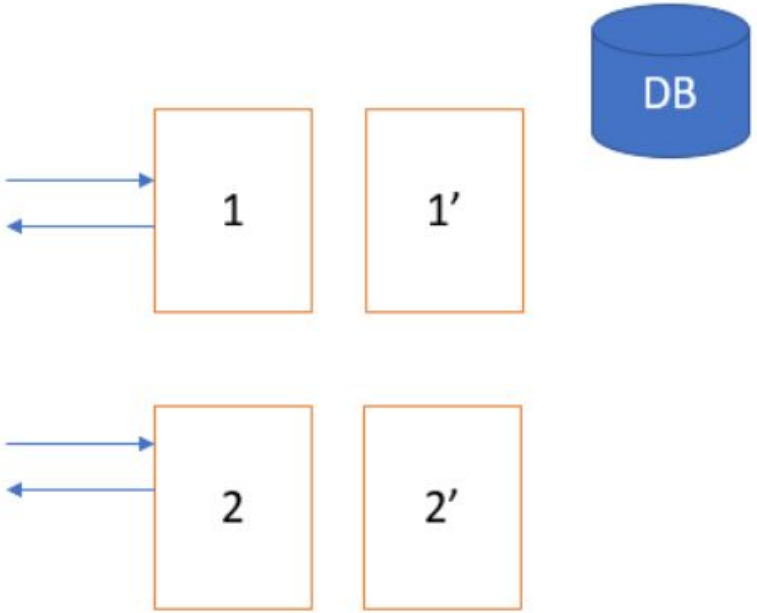


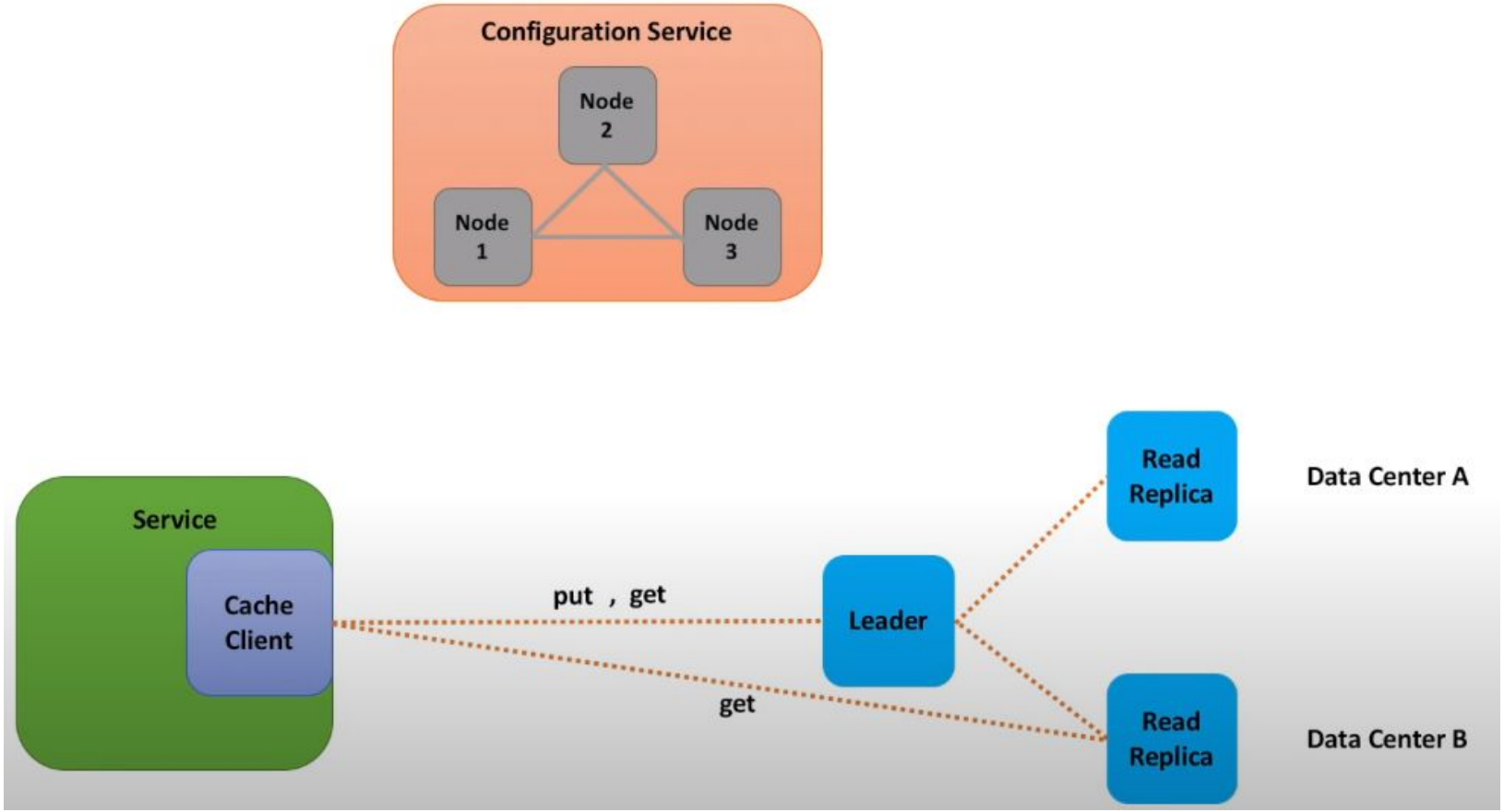
# Distributed Cache

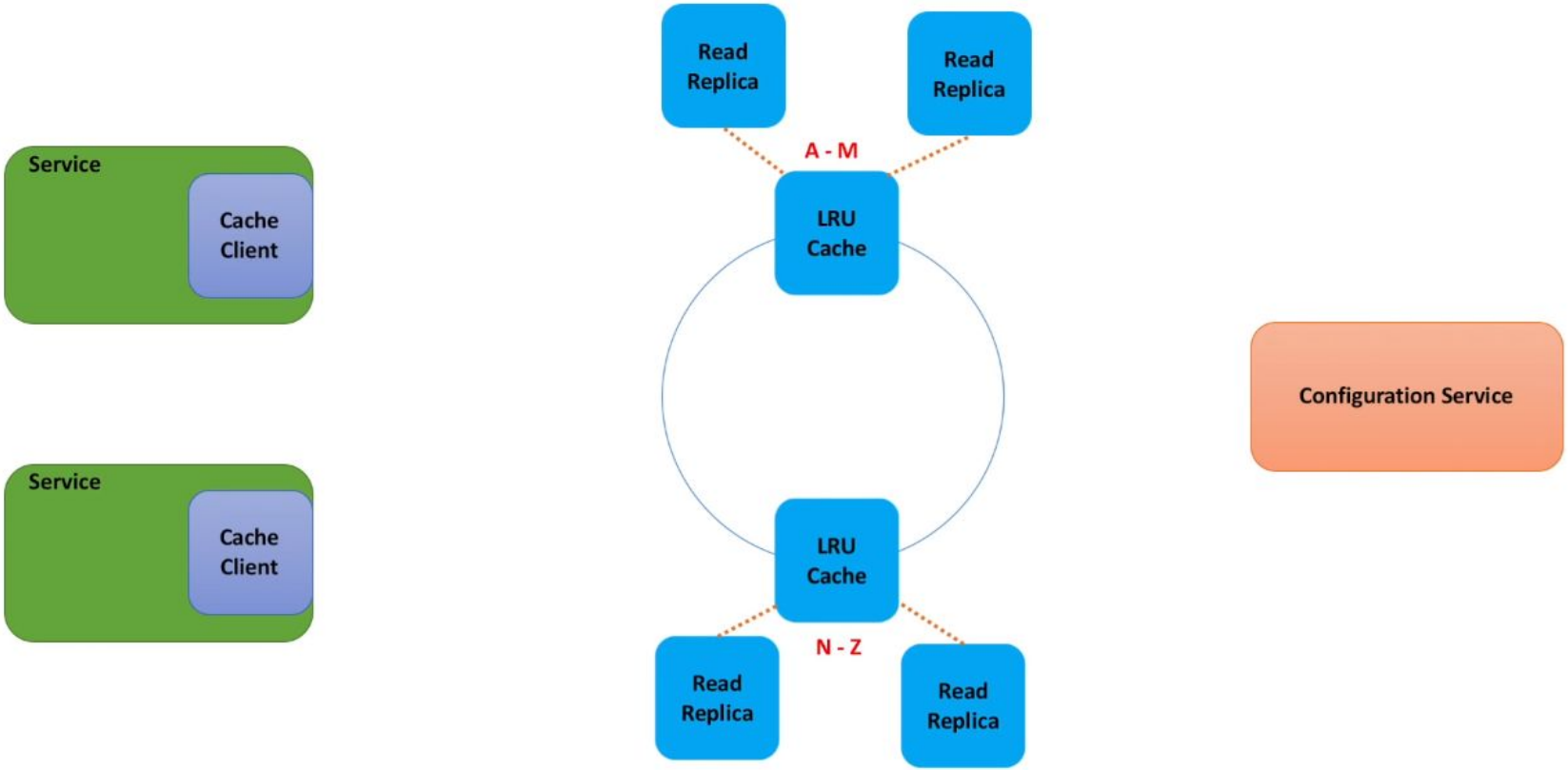
- Scalability
- High Availability
- Fault Tolerance
- Consistency

# Availability




# Consistency! 🤔


Configuration Service
Node 2
Node 1
Node 3
Service
Cache Client
put , get
get
Leader
Read Replica
Read Replica
Data Center A
Data Center B


Service
Cache Client
Service
Cache Client
Read Replica
Read Replica
A - M
LRU Cache
LRU Cache
N - Z
Read Replica
Read Replica
Configuration Service

# Distributed Cache

- Scalability
- High Availability
- Fault Tolerance
- Consistency

# Split to Smaller Shards!

# Popular Distributed Cache

- Memcache
- Redis

# Reference

- https://serhatgiydiren.github.io/system-design-interview-distributed-cache
- https://medium.com/system-design-concepts/distributed-cache-system-design-9560f7dd07f2
- https://medium.com/rtkal/distributed-cache-design-348cbe334df1
- https://www.youtube.com/watch?v=qI_g07C_Q5I&t=2562s
-