

Intro to Distributed Databases

Big Data Systems

Dr. Rubi Boim

Agenda for today

- Motivation
- Distributed relational database?
- “Going distributed”

מהשפעות התקלה ב-AWS: מיטות חכמות התחממו, קפאו ונתקעו בזווית שלא מאפשרת לישון

הגאדג'טים הכי מגניבים למיטה שלכם עוררו (סליחה) סערה השבוע, כשהתקלה של אמזון הובילה להרבה צרות אצל המשתמשים

{ נאדנ'טים } • עידן בן טובים • 24.10.25



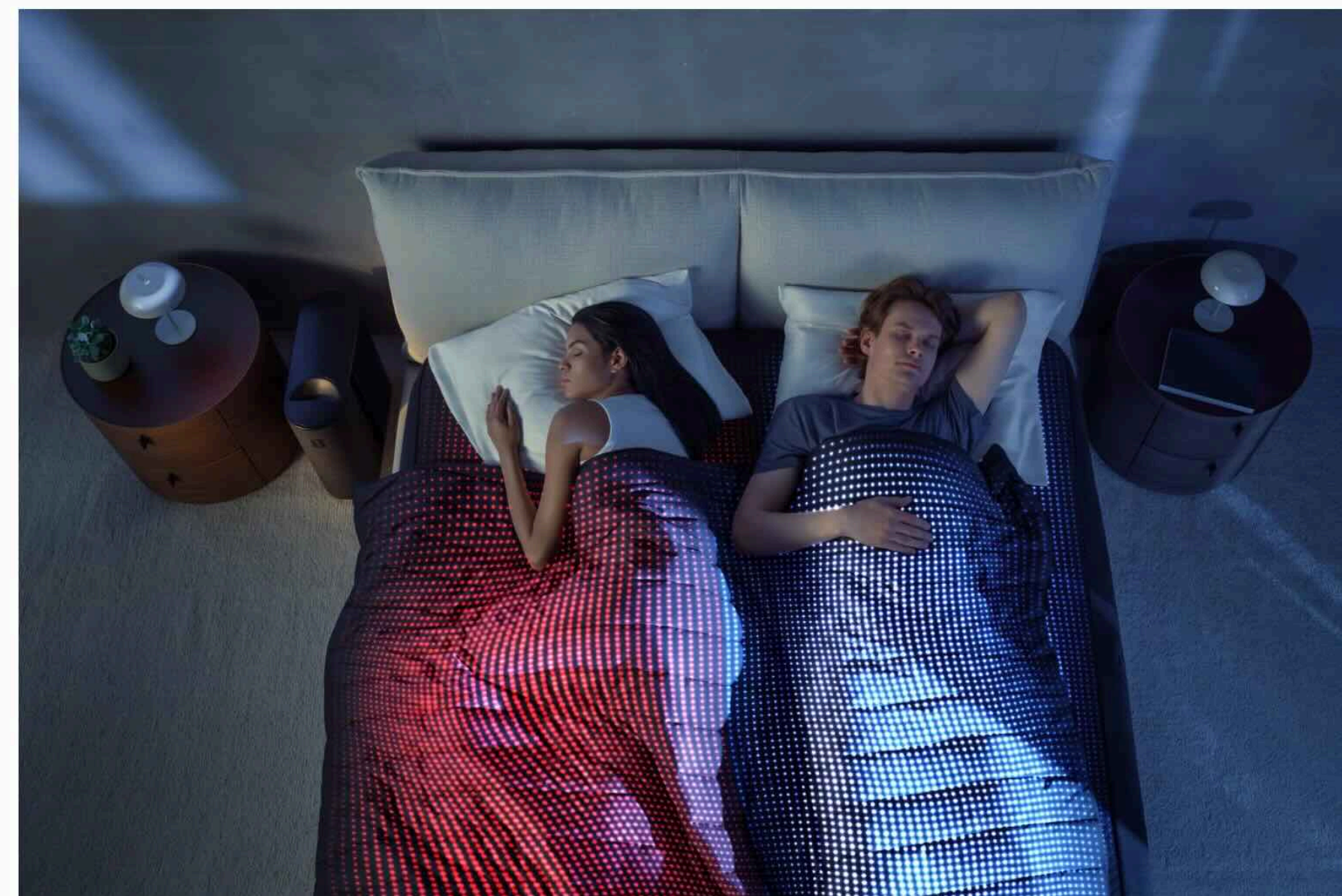
Powered by **Trinity Audio**

למה לקרוא כשאפשר להאזין?

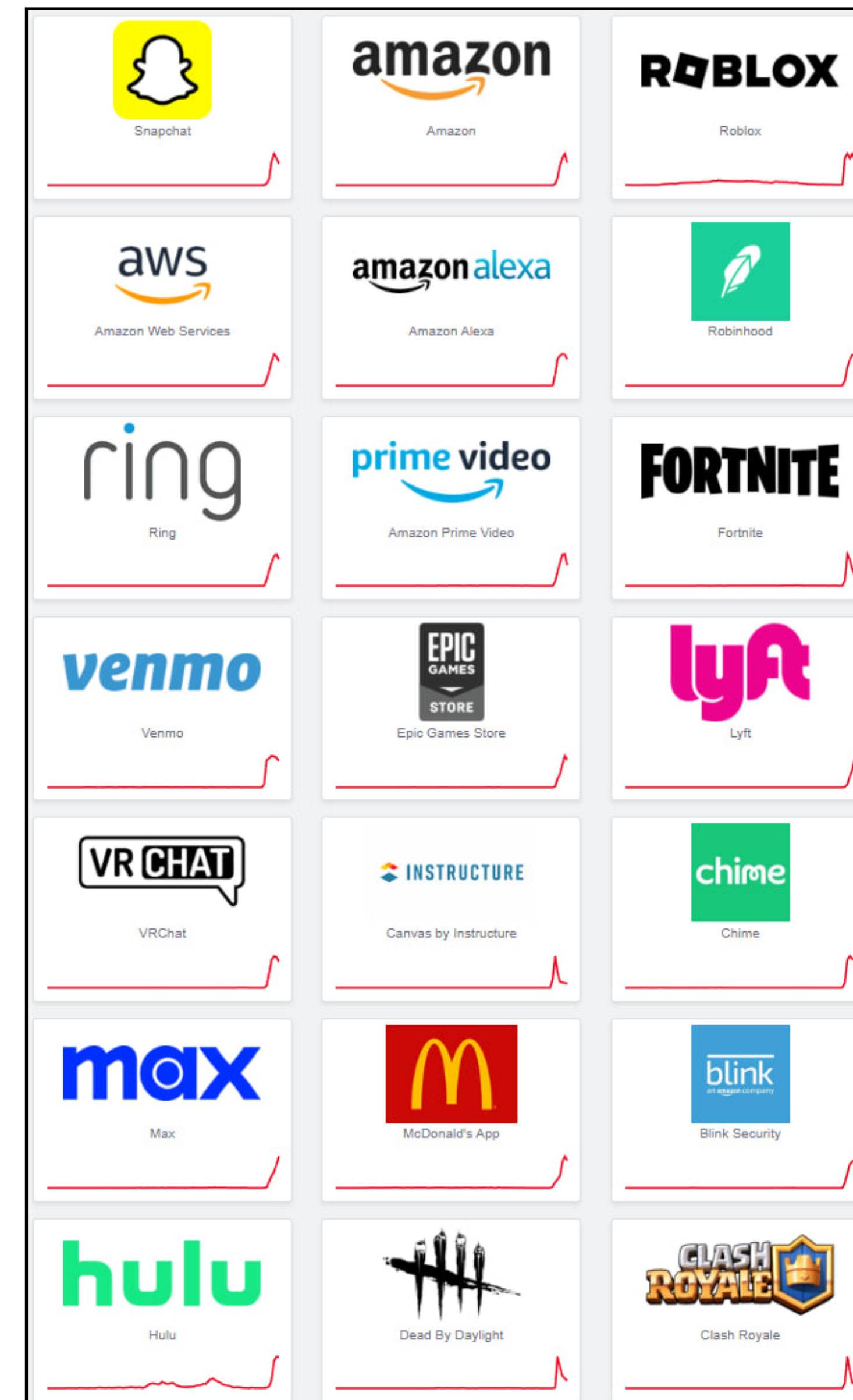
1.0x 10 10 A

00:00

04:25



תמונה: Eight Sleep



מהשפעות התקלה ב-AWS: מיטות



Snapchat



Amazon



Roblox

Main cause

- 1: a broken automation system that kept re-breaking DNS
- 2: recovering from a massive, global retry/backlog storm across hundreds of services once DNS was fixed

00:00 04:25



תמונה: Eight Sleep

venmo

Venmo



Epic Games Store



Lyft



VRChat



Canvas by Instructure



Chime



Max



McDonald's App



Blink Security



Hulu



Dead By Daylight



Clash Royale

מהשפעות התקלה ב-AWS: מיטות



Snapchat

amazon

Amazon

ROBLOX

Roblox

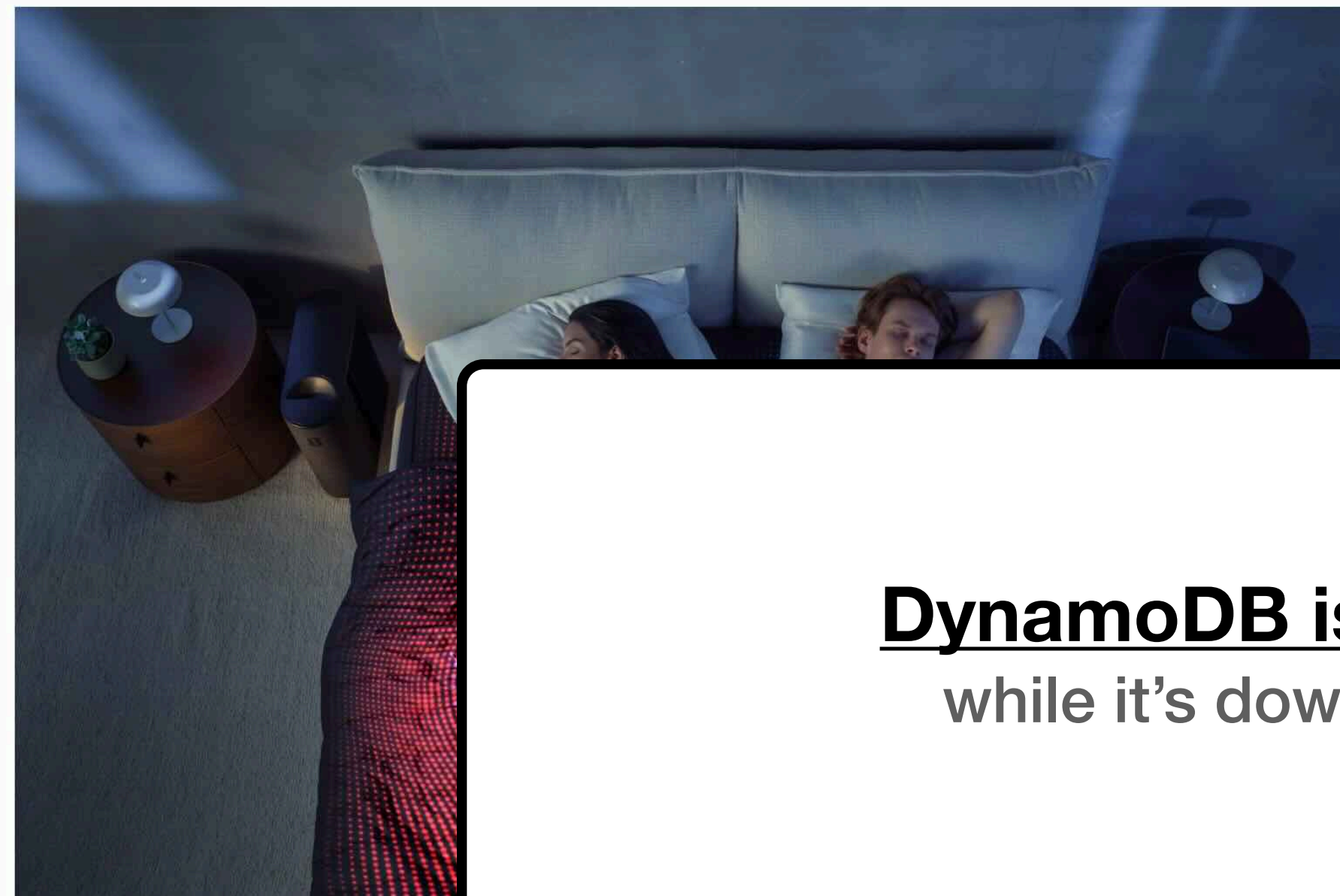
Main cause

1: a broken automation system that kept re-breaking DNS

2: recovering from a massive, global retry/backlog storm across hundreds of services once DNS was fixed

00:00

04:25



venmo

Venmo

EPIC
GAMES
STORE

Epic Games Store

lyft

Lyft

VR CHAT

VRChat

INSTRUCTURE

Canvas by Instructure

chime

Chime

blink

blink Security

ASH
ROYALE

ash Royale

DynamoDB is a foundational dependency

while it's down, everything retries & queues up

מהשפעות התקלה ב-AWS: מיטות



Snapchat



Amazon



Roblox

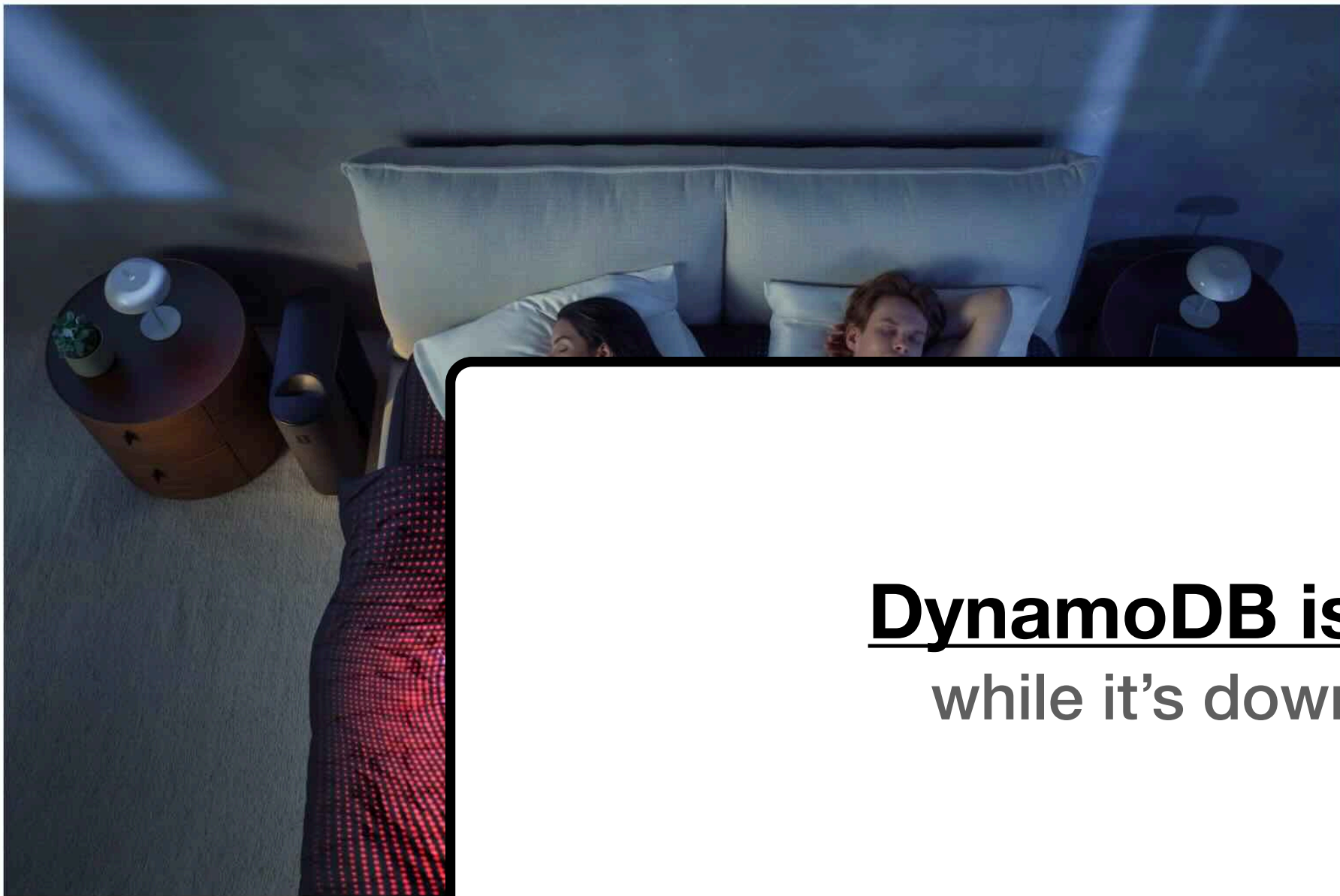
1: a broken au

2: recovering f

We need our main DB up and running all the time

NS was fixed

00:00



Venmo



Epic Games Store



Lyft



VRChat



Canvas by Instructure



Chime



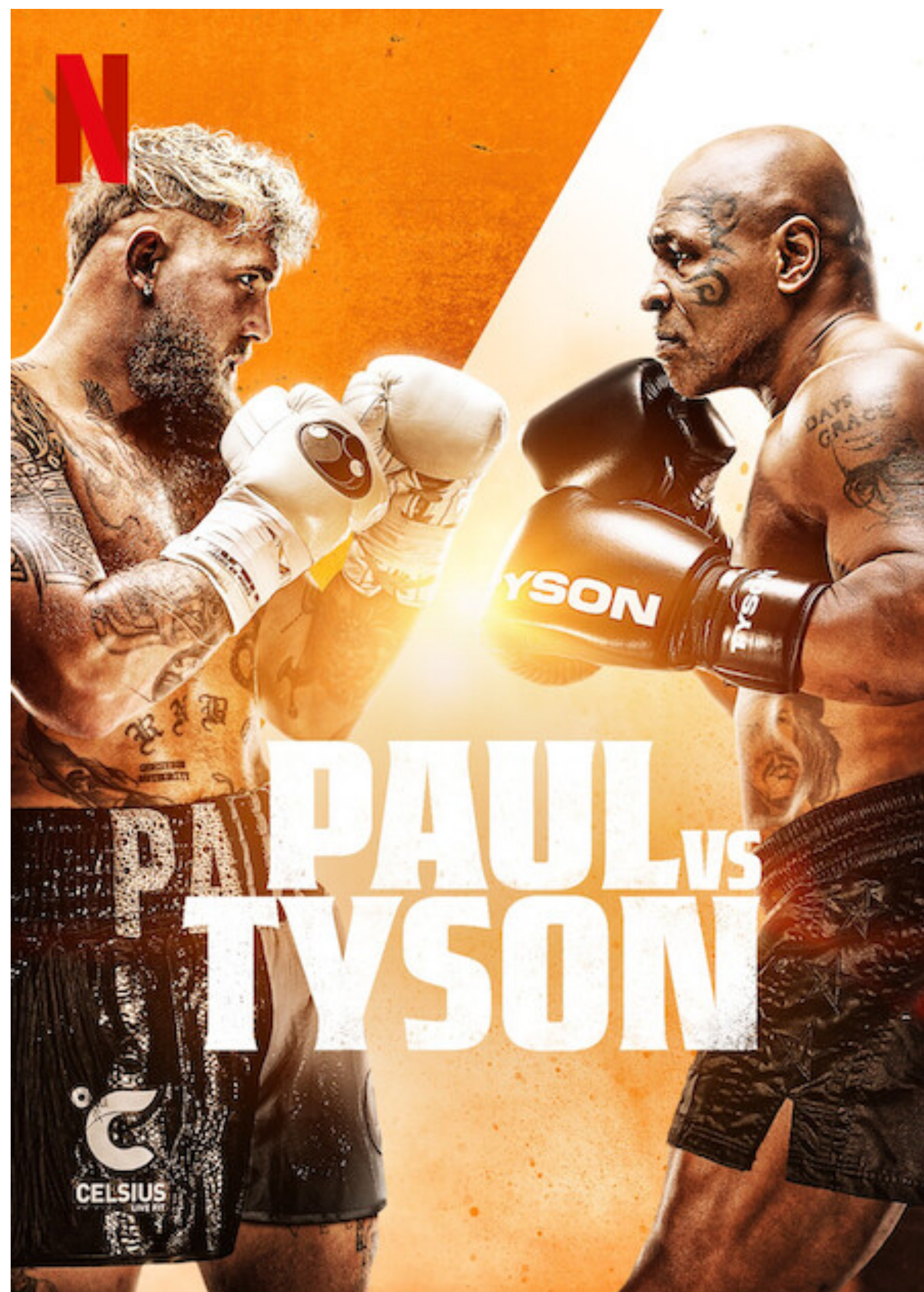
Blink Security



Hearthstone

DynamoDB is a foundational dependency
while it's down, everything retries & queues up





65 million concurrent viewers

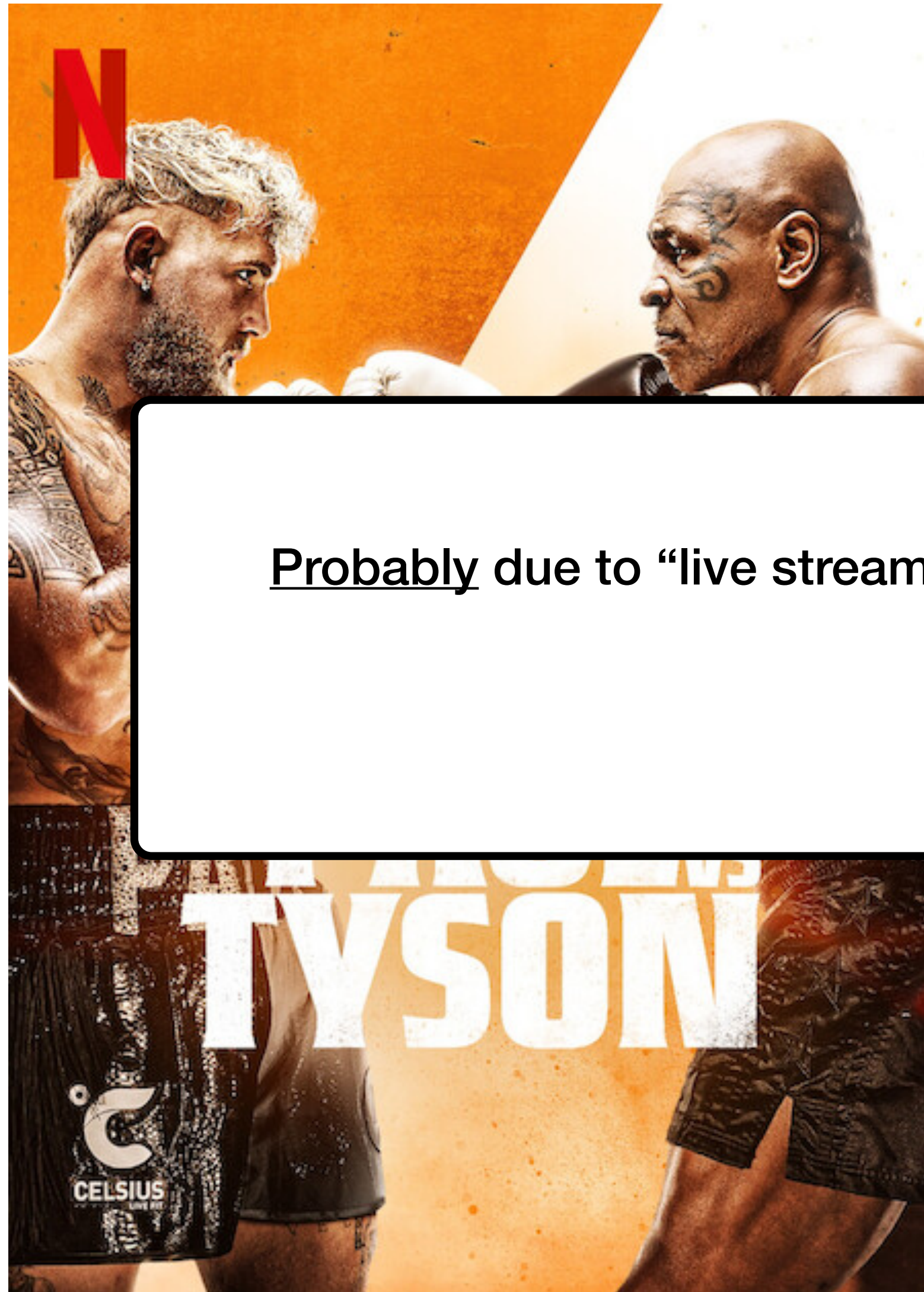
Netflix server crash during Mike Tyson vs Jake Paul boxing match leaves users fuming

People vented their frustration on social media platform as "#netflixcrash" became one of the top trends on X. The bout between the YouTuber-turned-boxer Paul, and Tyson, 58-year-old former heavyweight champion, at the AT&T Stadium in Arlington, Texas, was the biggest live sports event to date on Netflix.



By CNBCTV18.com

November 16, 2024, 3:45:37 PM IST (Updated)



65 million concurrent viewers

Netflix server crash during Mike Tyson vs Jake Paul boxing match leaves users

Probably due to “live streaming CDN load” and NOT due to Databases bottleneck
(personal speculation)

Biggest live sports event to date on Netflix.

By CNBCTV18.com

November 16, 2024, 3:45:37 PM IST (Updated)

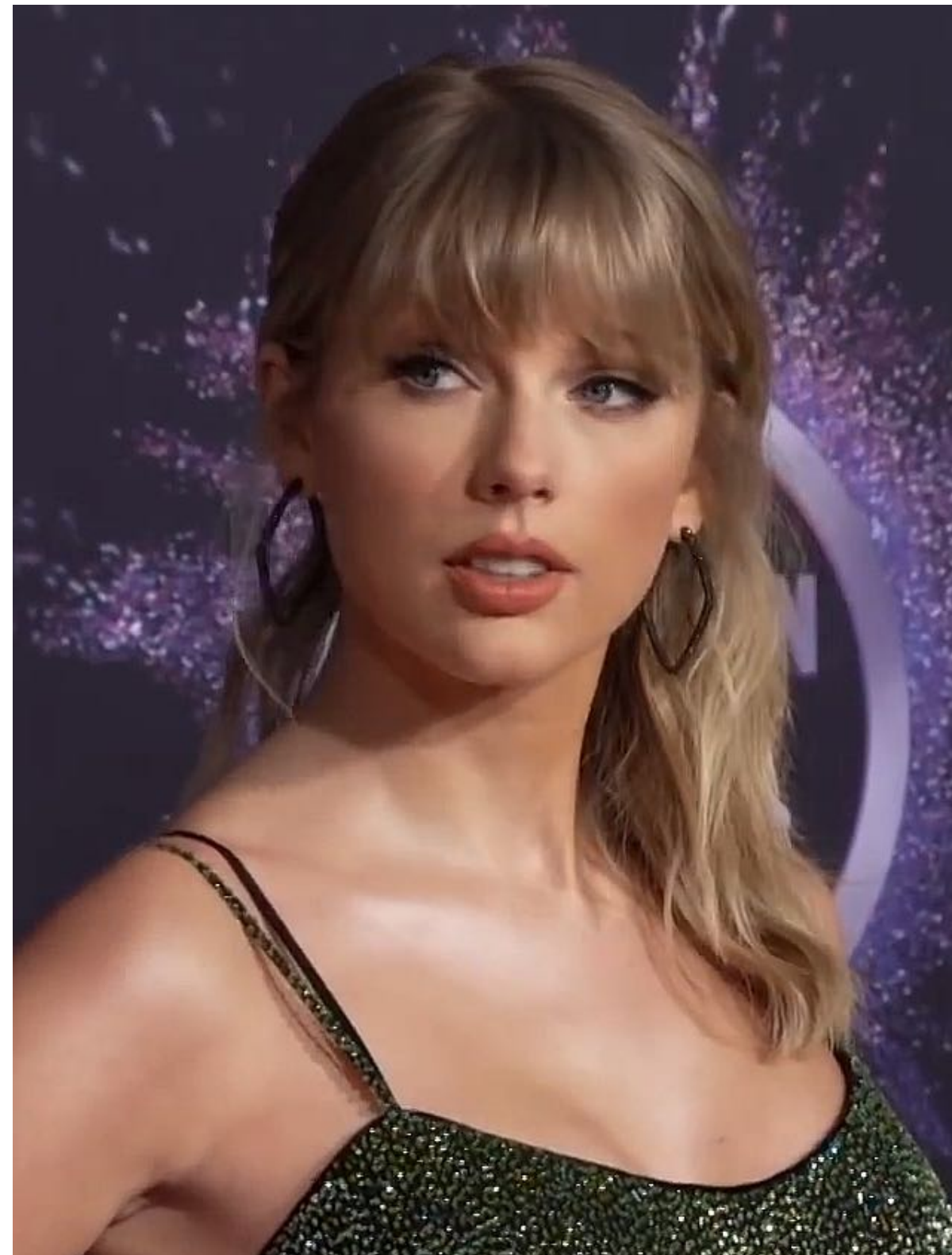
Example 1

Example 1



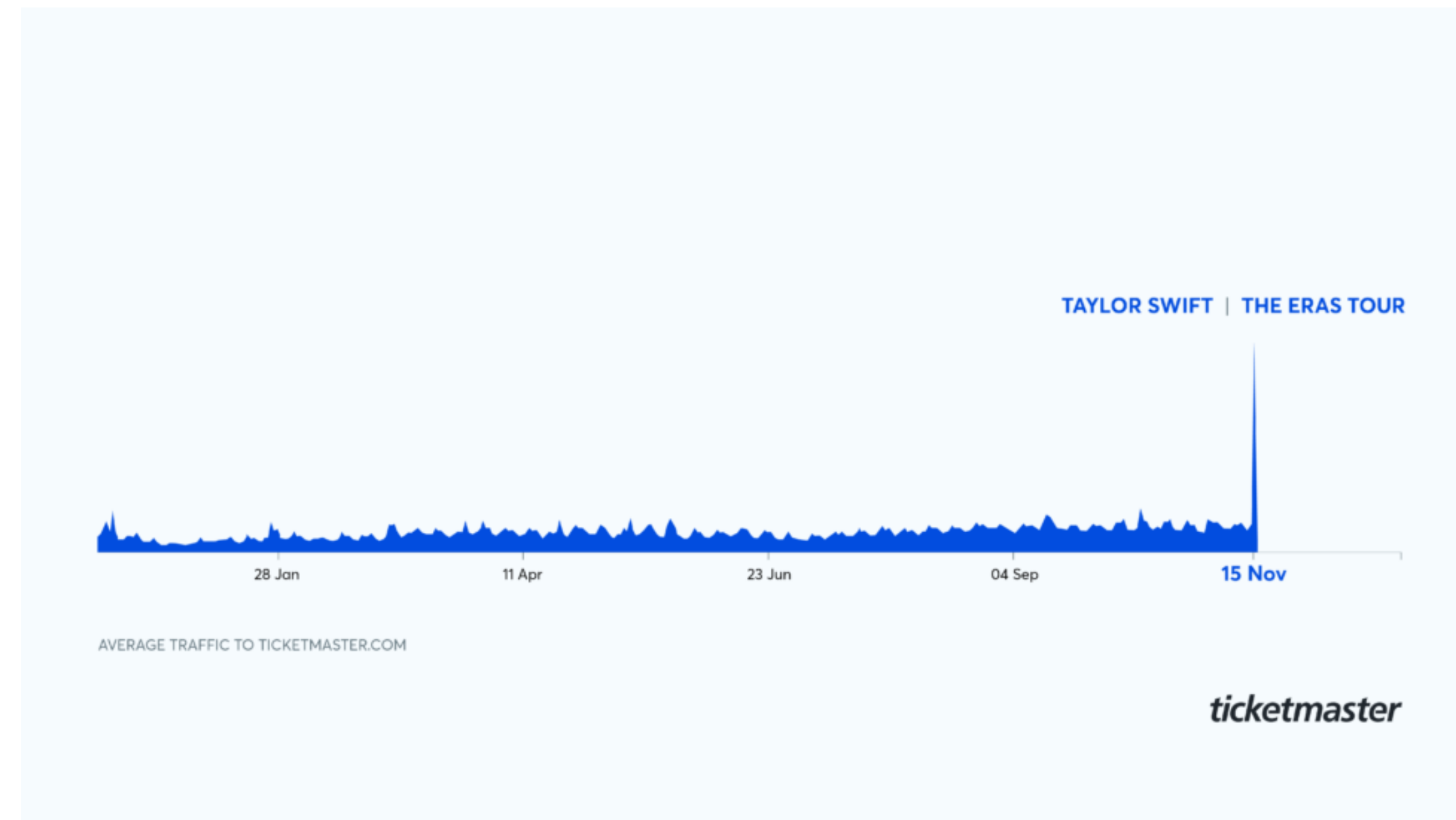
* image by Wikipedia

Example 1



HOME / BUSINESS / MUSIC

Taylor Swift Ticket Sales Crash Ticketmaster, Ignite Fan Backlash, Renew Calls To Break Up Service: “Ticketmaster Is A Monopoly”



**14m users spike, 2m tickets sold,
then Ticketmaster crashed**

Example 1 - a year later



Example 1 - a year later



Oasis fans 'break' Ticketmaster as website crashes before tickets go on sale

Fans have already branded the ticket process a 'disaster' as they were met with issues getting on to the website before the sale even began

This article contains affiliate links, we will receive a commission on any sales we generate from it. [Learn more](#)

Bookmark 

SHARE



By [Catherine Addison-Swan](#)

08:54, 31 AUG 2024

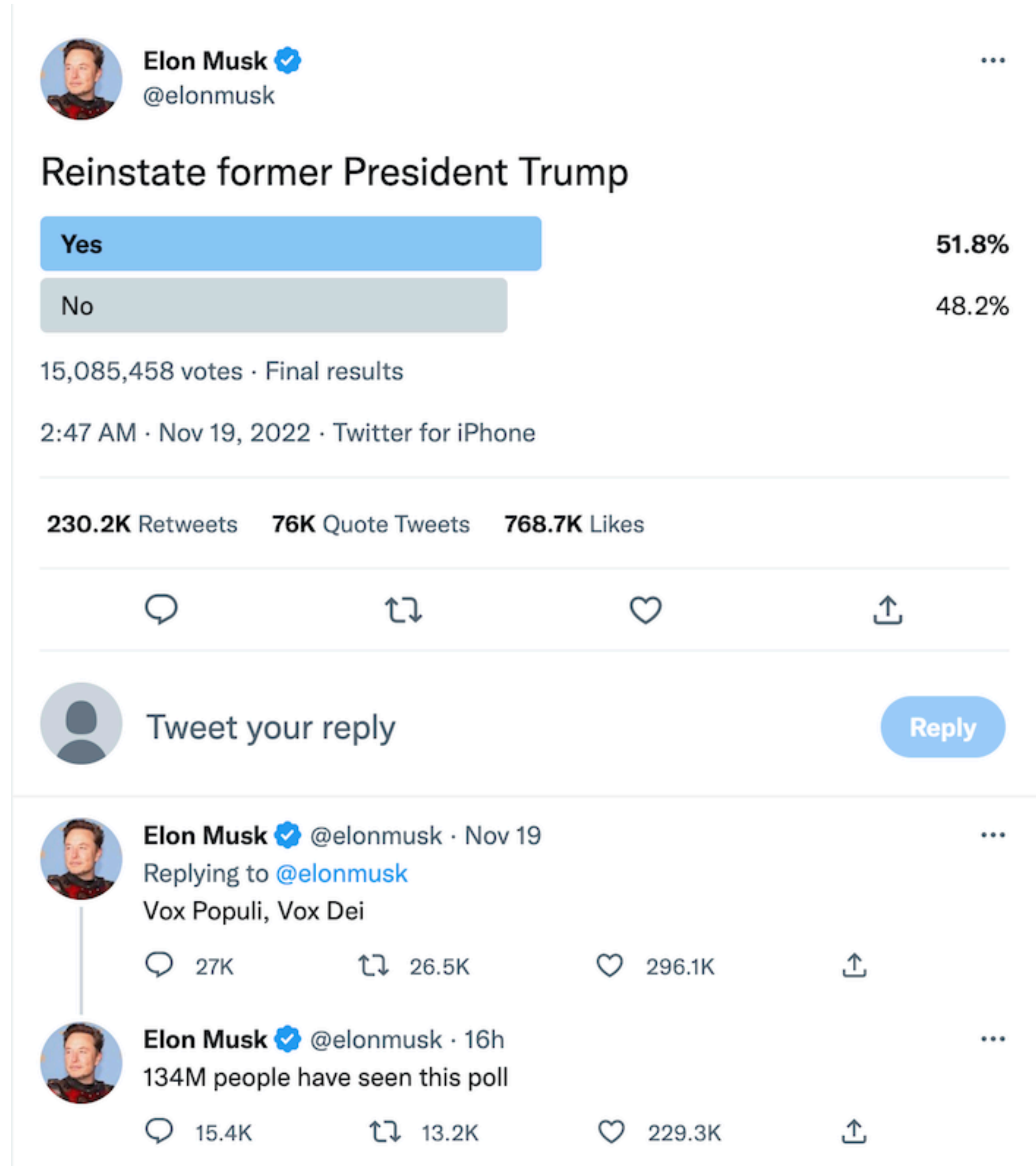
UPDATED 10:15, 31 AUG 2024

Example 2



* image by Wikipedia

Example 2



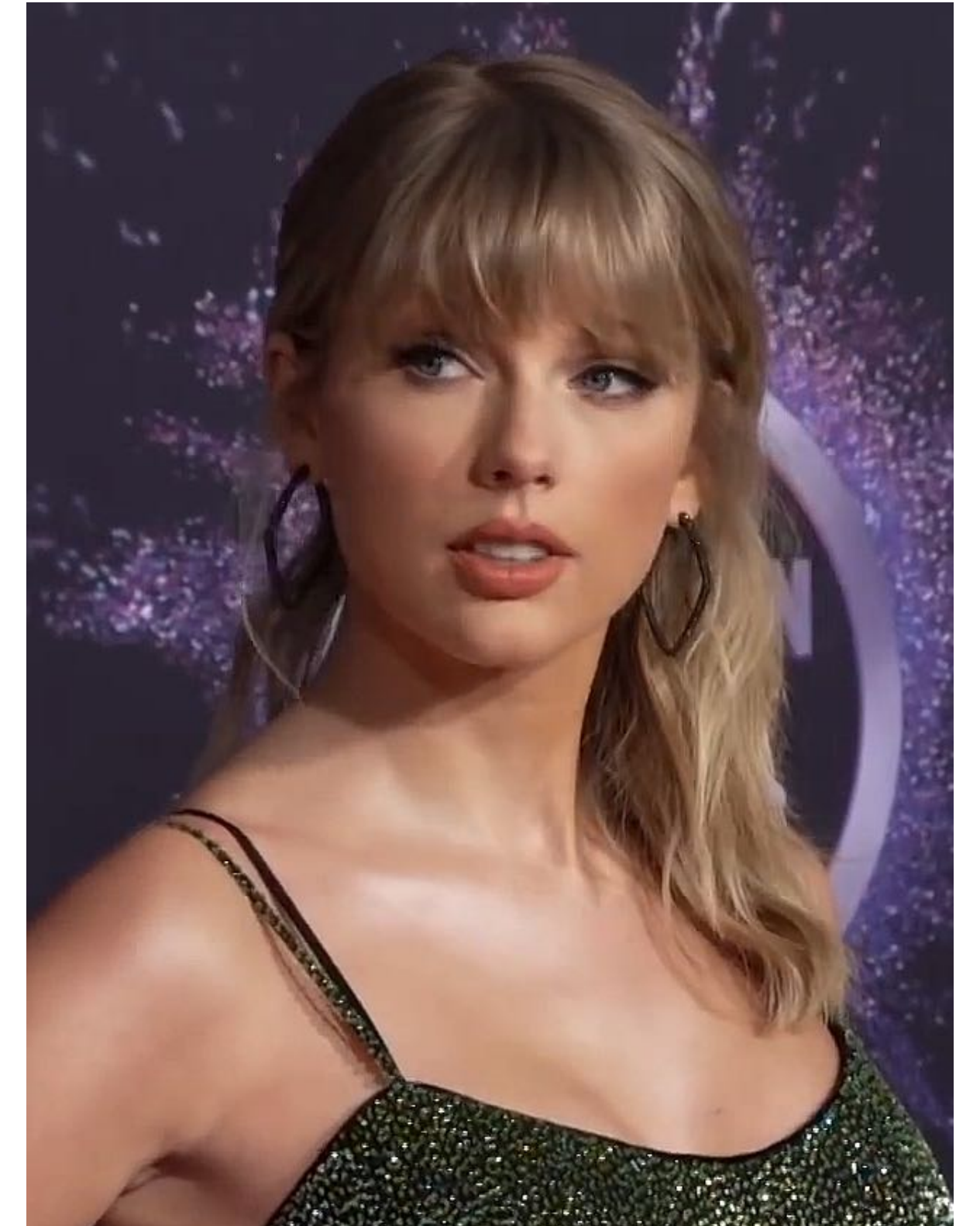
Example 2



Example



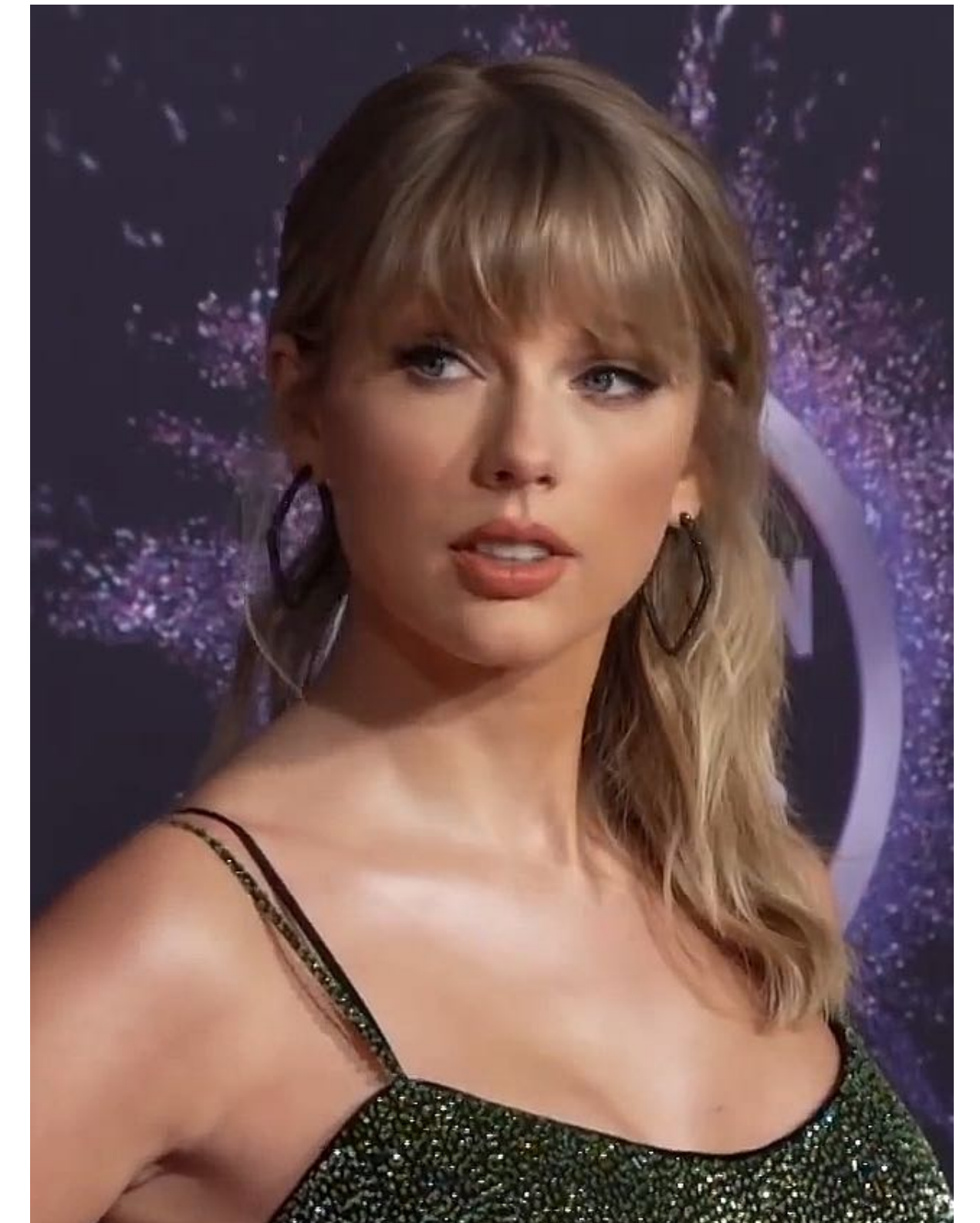
How X managed to
handle 10x more traffic than
Ticketmaster without any crash?



Example



How X managed to
handle 10x more traffic than
Ticketmaster without any crash?



What will happen if the poll will be less accurate
and show 15,084,534 instead of 15,085'458?

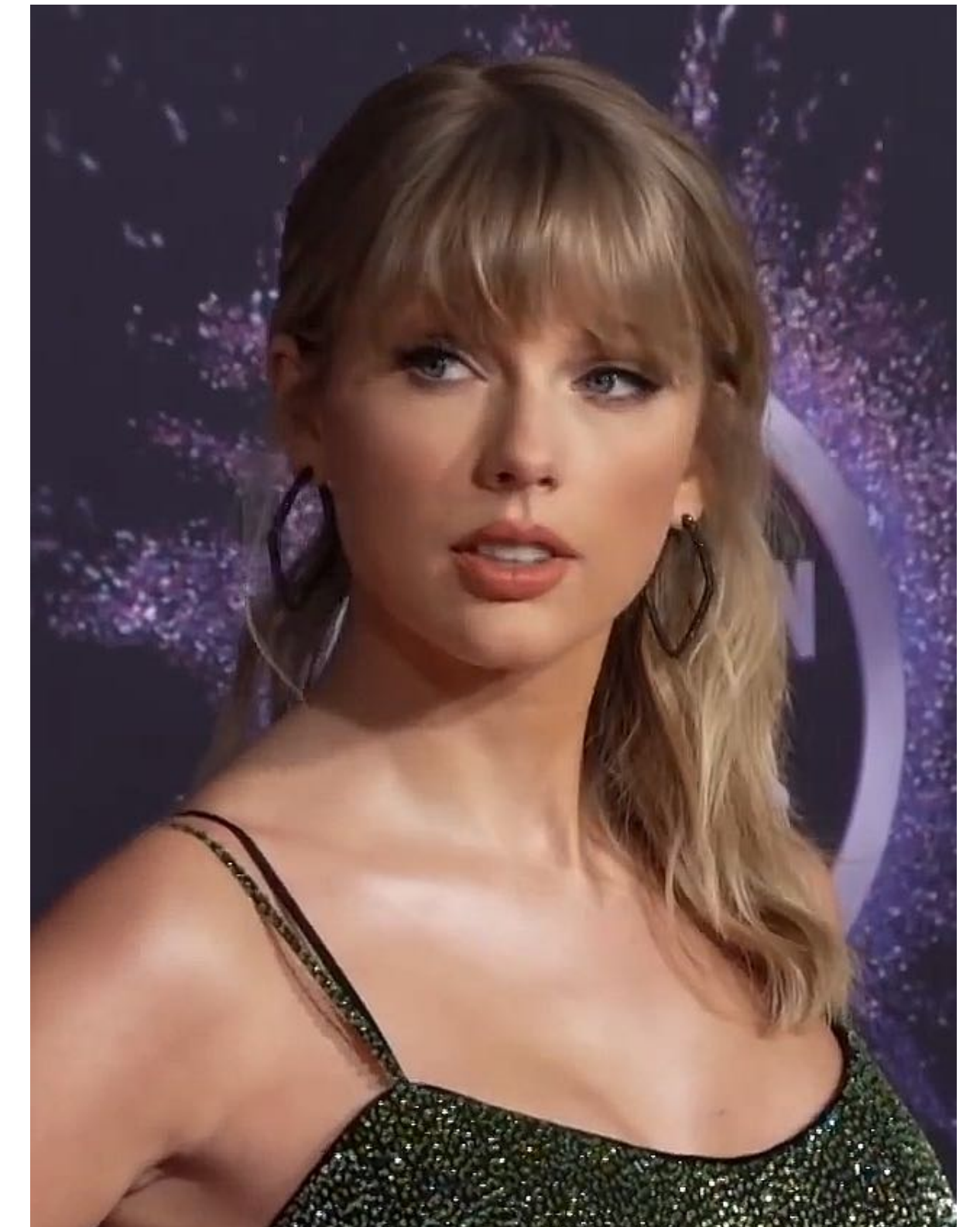
What will happen if show ticket row14/seat94 as
available although it was already sold?

Example



How X managed to
handle 10x more traffic than
Ticketmaster without any crash?

Consistency
(we will talk about it a lot)



What will happen if the poll will be less accurate
and show 15,084,534 instead of 15,085'458?

What will happen if show ticket row14/seat94 as
available although it was already sold?

Taylor > Apple / Spotify



Mirror

1898 (Taylor's Version) has finally dropped (Image: @taylorswift13/Twitter)

[NEWS](#)[POLITICS](#)[FOOTBALL](#)[CELEBS](#)[TV](#)[CHOICE](#)[ROYALS](#)

Spotify and Apple Music DOWN after Taylor Swift's 1989 rerecord release

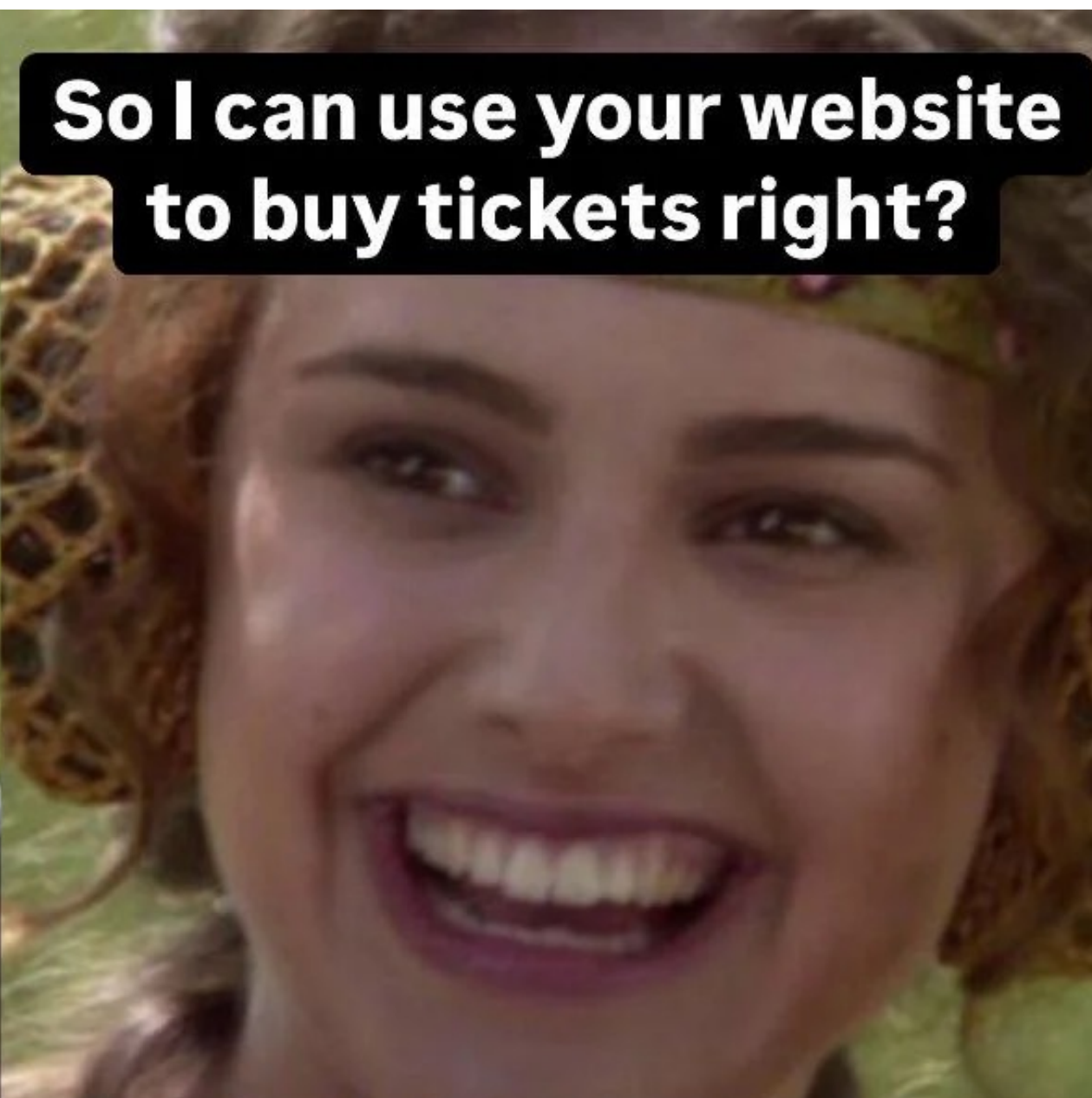
Taylor Swift's have been counting down the days to the release of 1989 (Taylor's Version), but some Swifties had to wait longer than they should have due to issues with Spotify and Apple Music

By [Zoe Forsey](#), Deputy Editor, Mirror US
05:23, 27 Oct 2023 | UPDATED 07:43, 27 OCT 2023

| | [BOOKMARK](#)



**We are the worlds
largest ticketing
platform**



**So I can use your website
to buy tickets right?**



Right?





**I'M GONNA ORDER
PIZZA FOR DINNER**



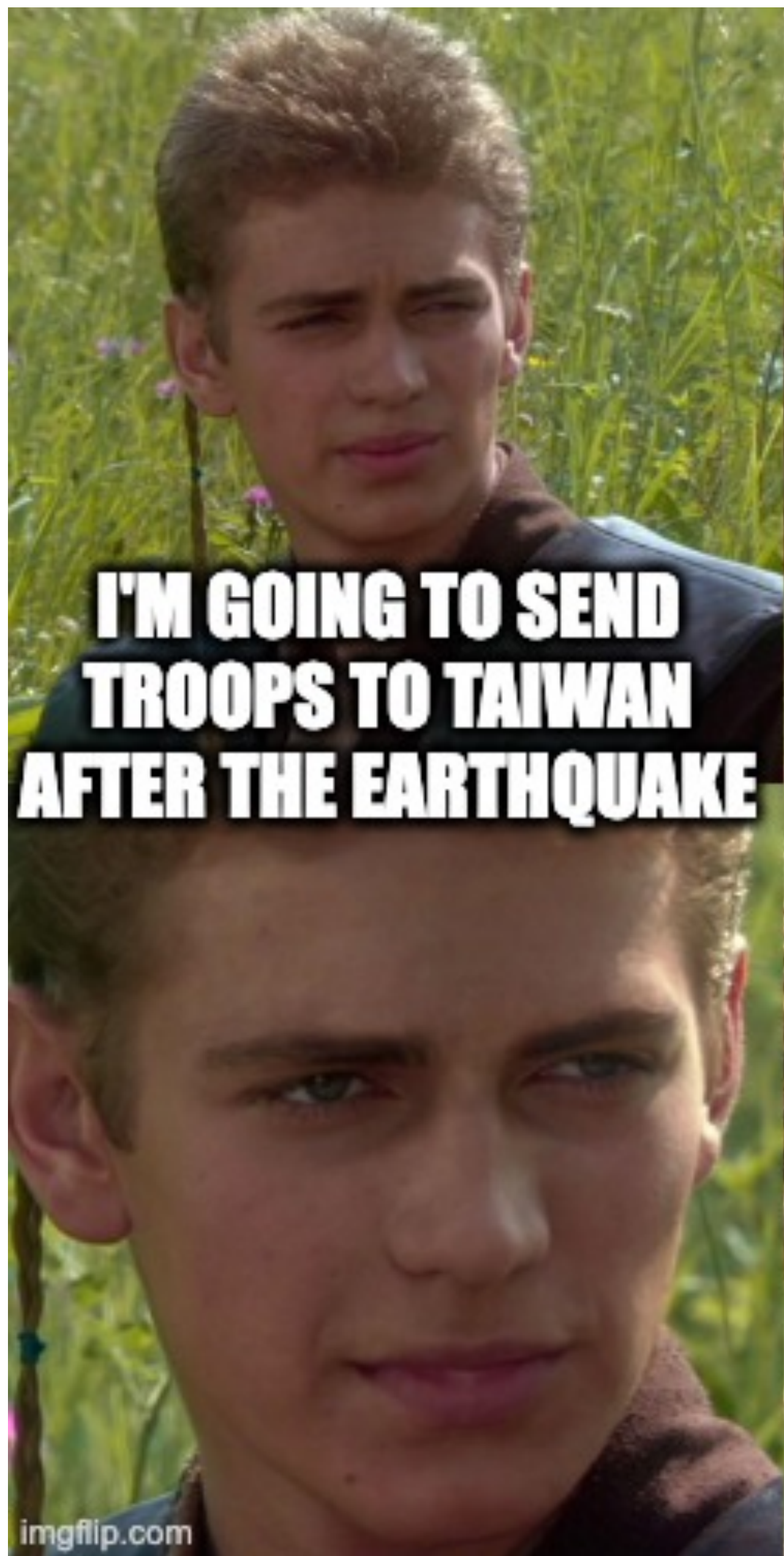
**IT'S JUST FOR
TONIGHT, RIGHT?**



made with AI on imgflip.com



RIGHT?





**MY FAV COLOR
IS ULTRAVIOLENT**

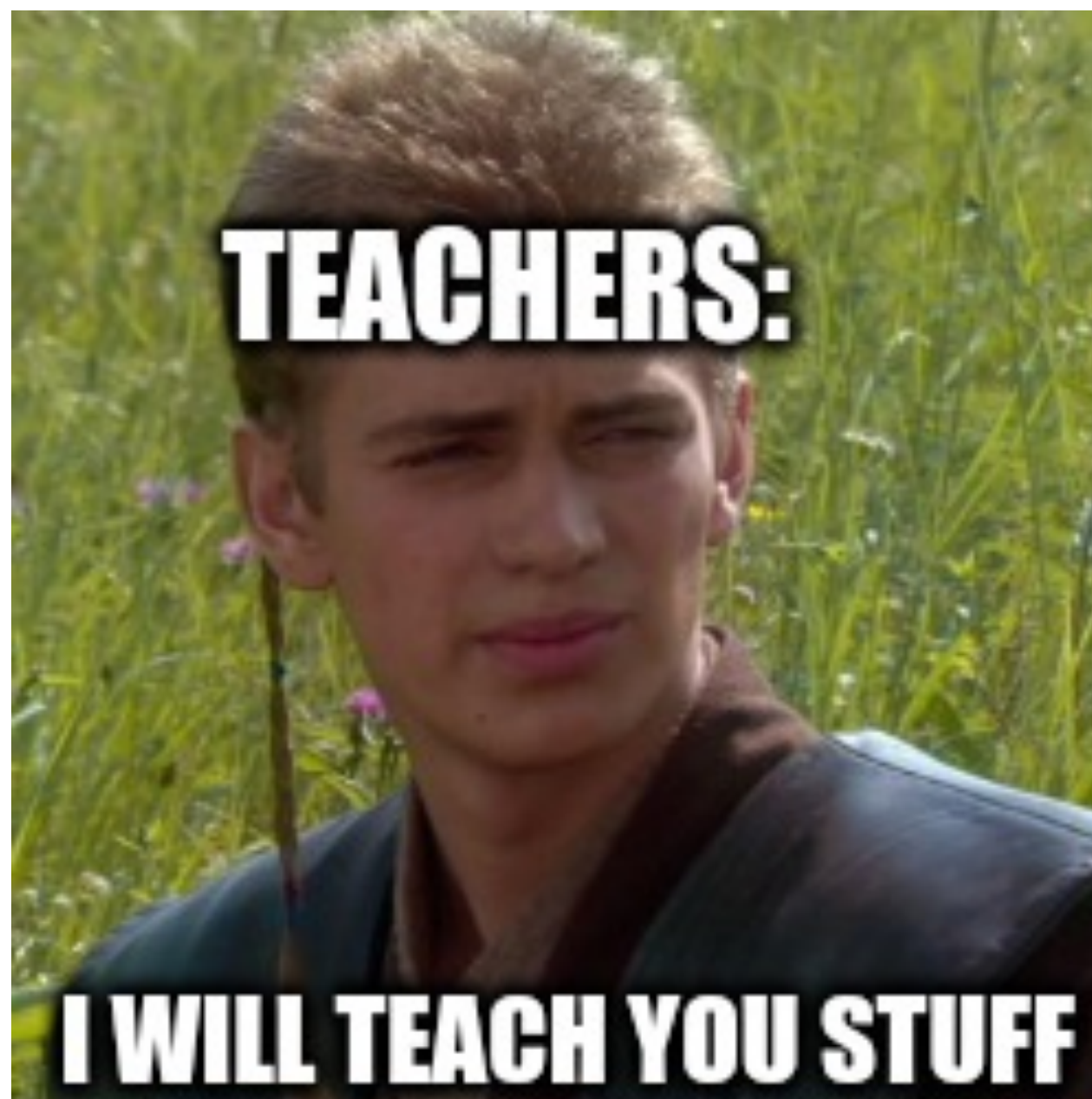


**YOU MEAN
ULTRAVIOLET, RIGHT?**



imgflip.com





TEACHERS:

I WILL TEACH YOU STUFF



**IT WILL HELP
ME IN THE FUTURE**



imgflip.com



**IT WILL HELP ME
IN THE FUTURE RIGHT?**

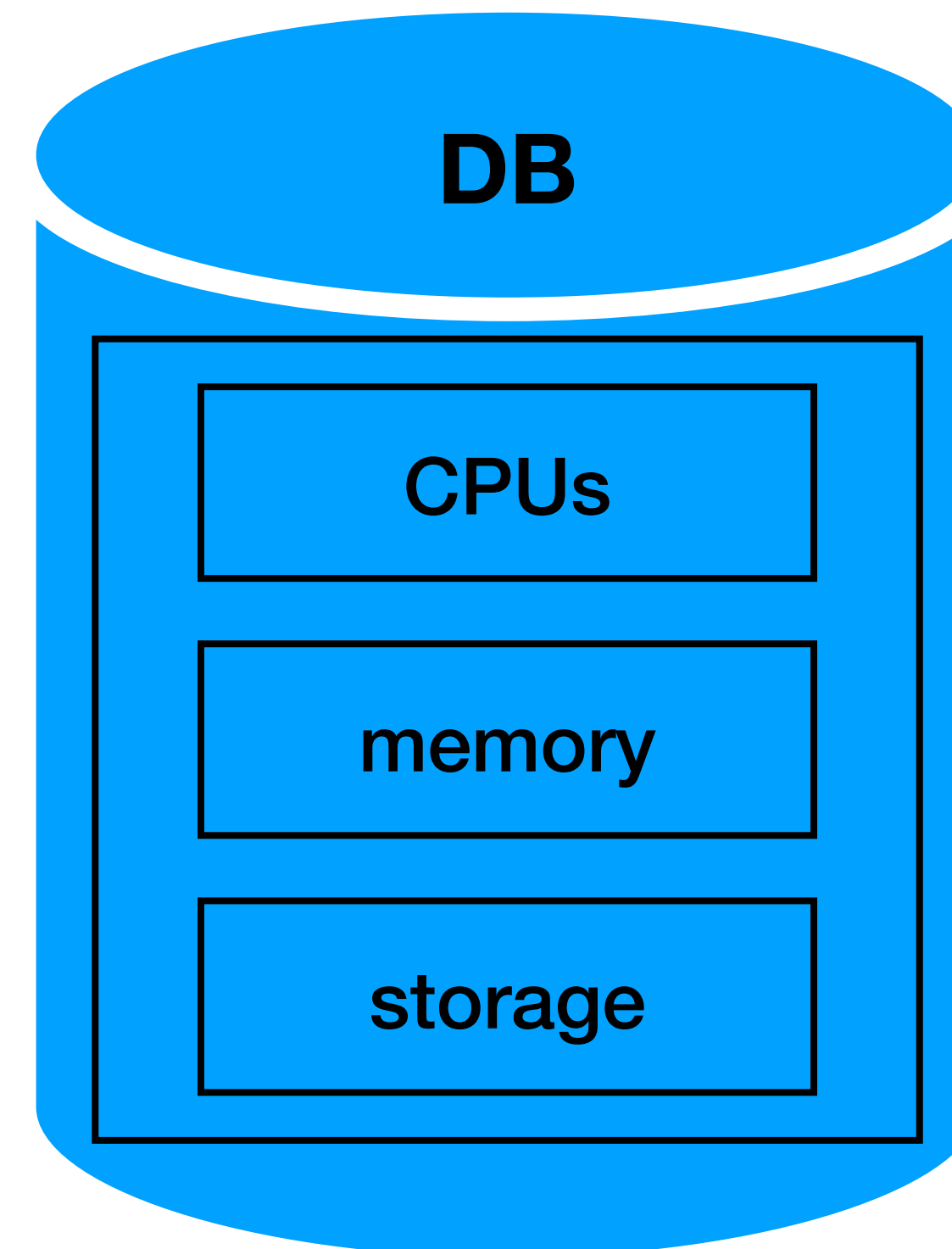
Motivation

Centralized RDBMS

- Used everywhere
- Proven technology
- So why do we need anything else?

Motivation

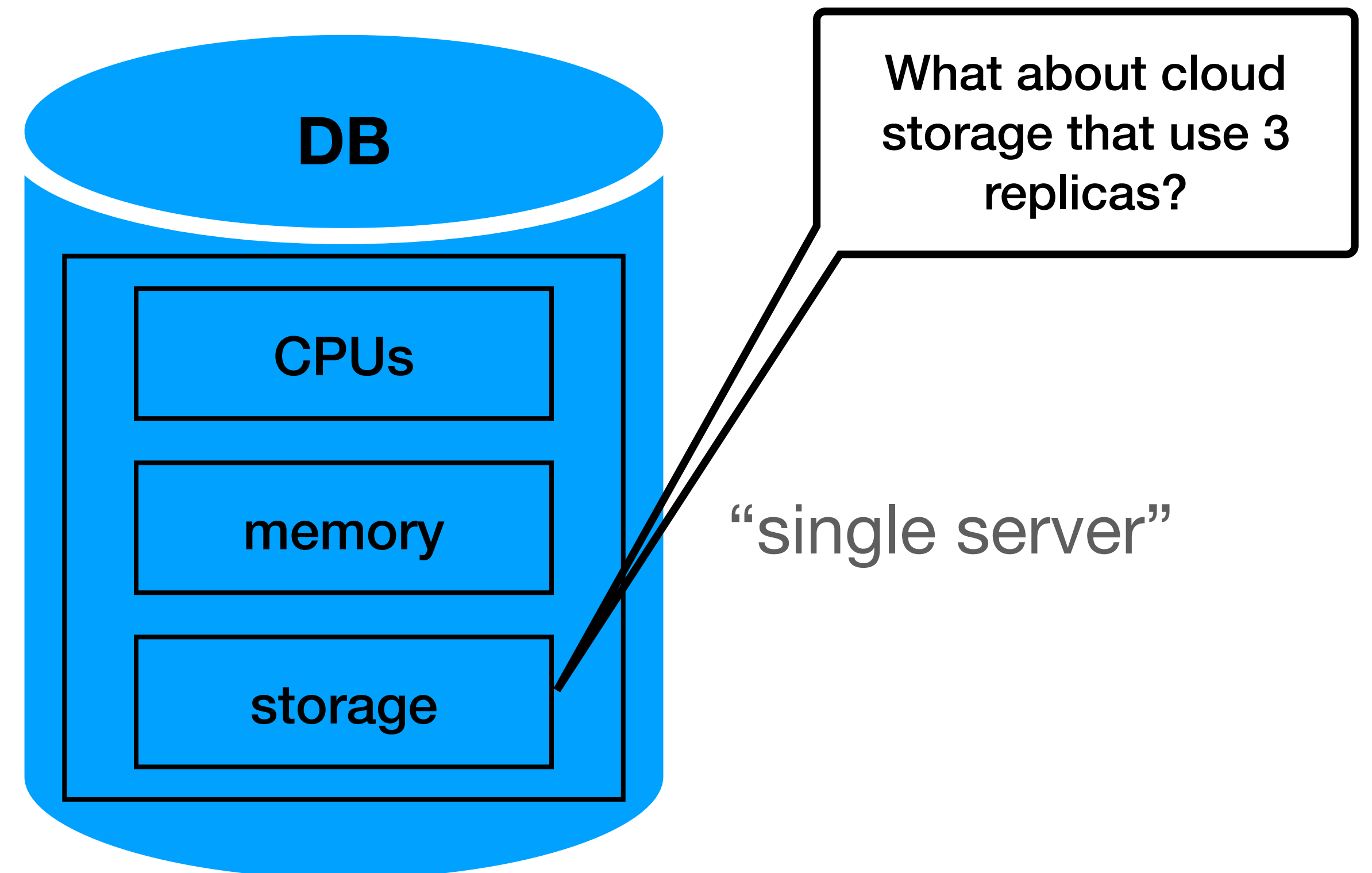
Centralized RDBMS (simplified)



“single server”

Motivation

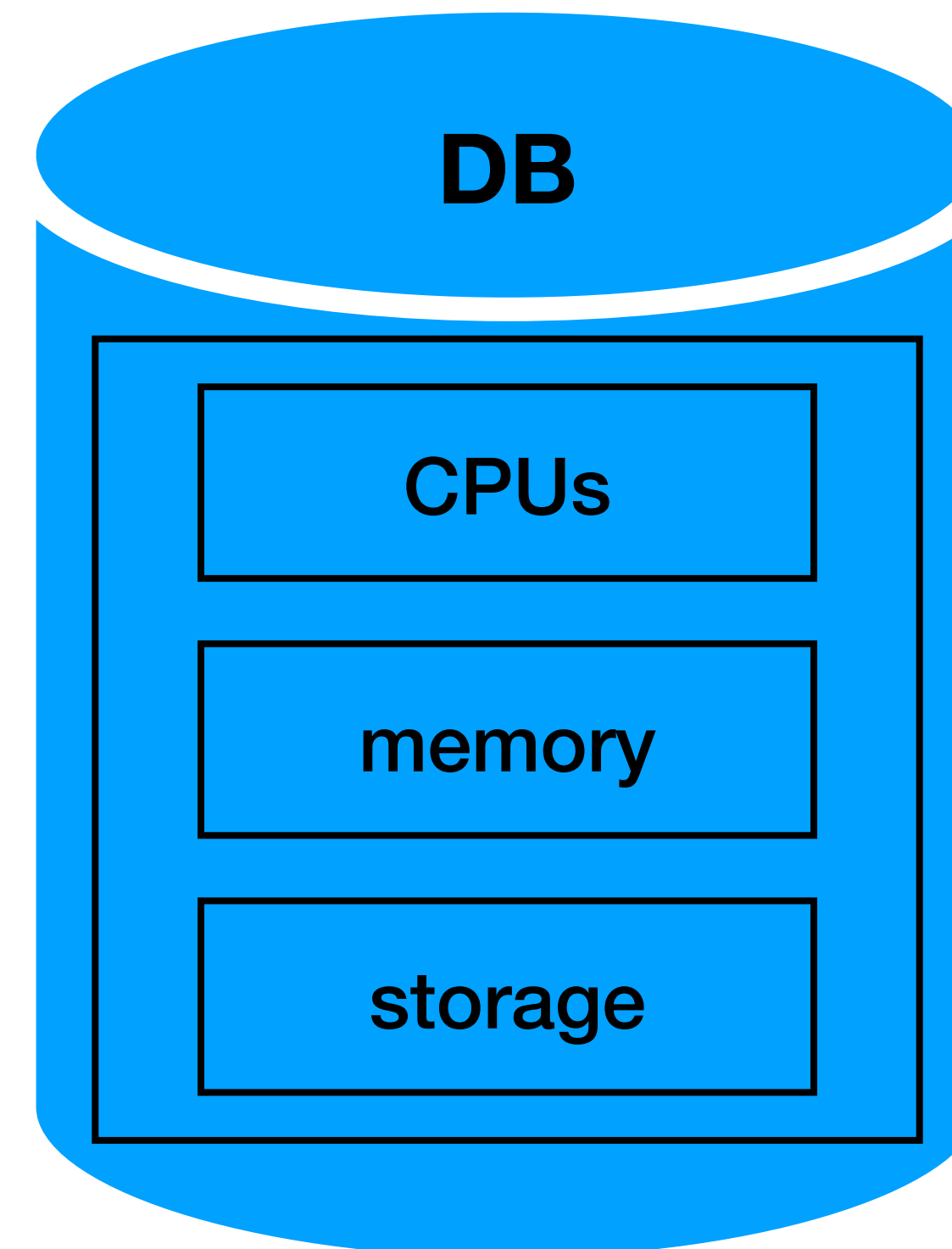
Centralized RDBMS (simplified)



Motivation

Centralized RDBMS (simplified)

- What happens if we have more than 100m rows?
 - Storage
 - Query time

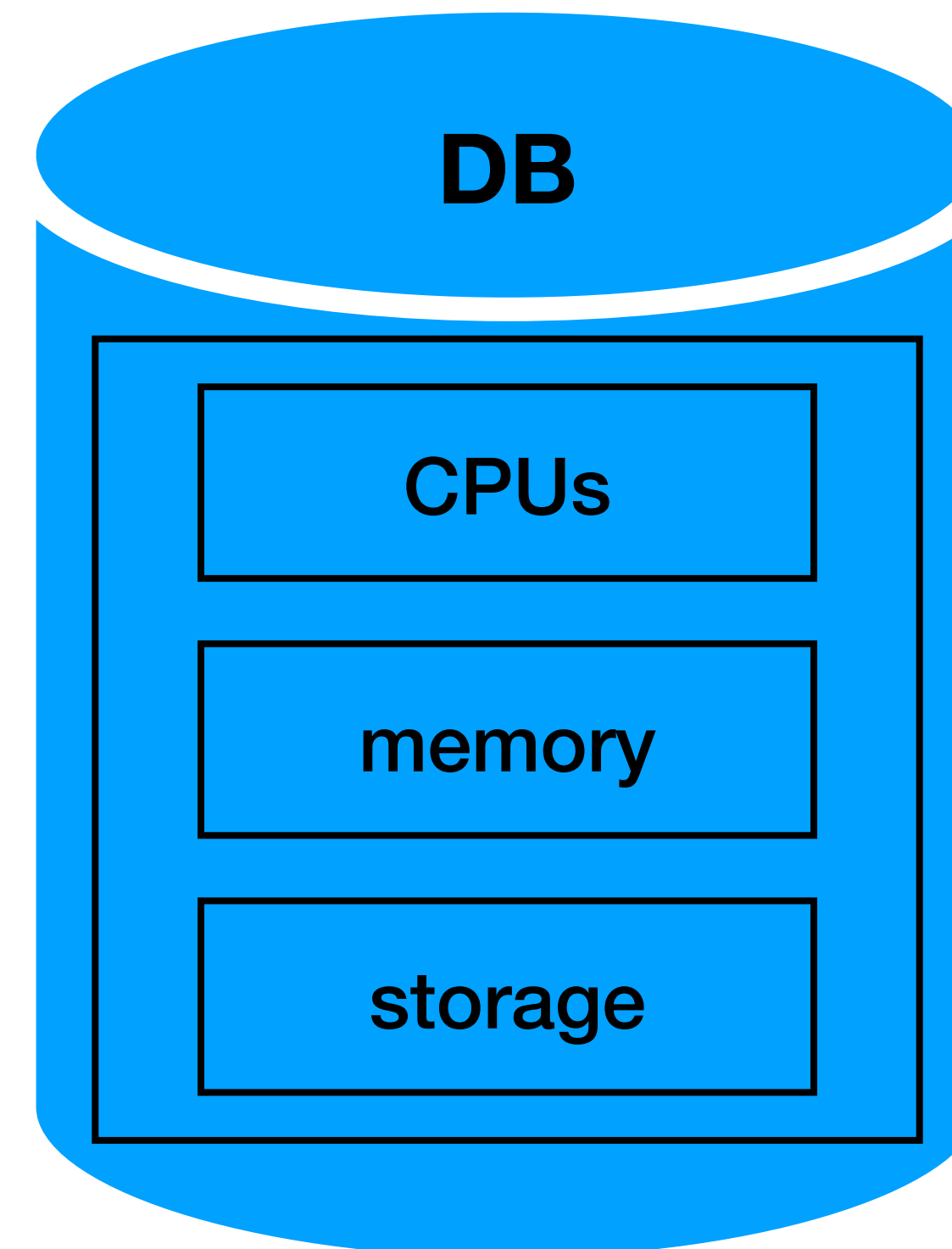


“single server”

Motivation

Centralized RDBMS (simplified)

- What happens if we have more than 100m rows?
 - Storage
 - Query time
- The index is crucial

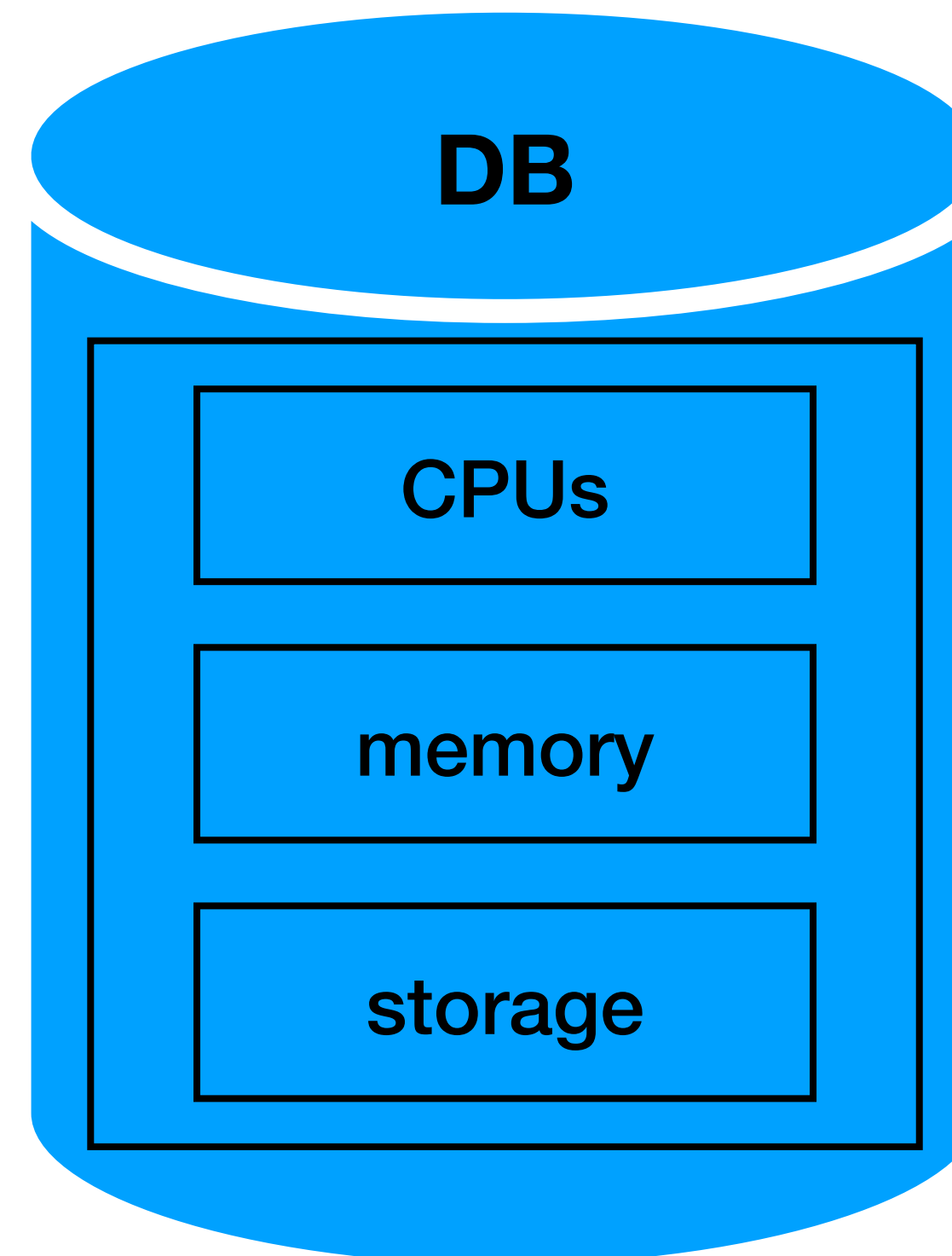


“single server”

Motivation

Centralized RDBMS (simplified)

- What happens if we have more than 100m rows?
 - Storage
 - Query time
- What happens if the index is bigger than the memory?



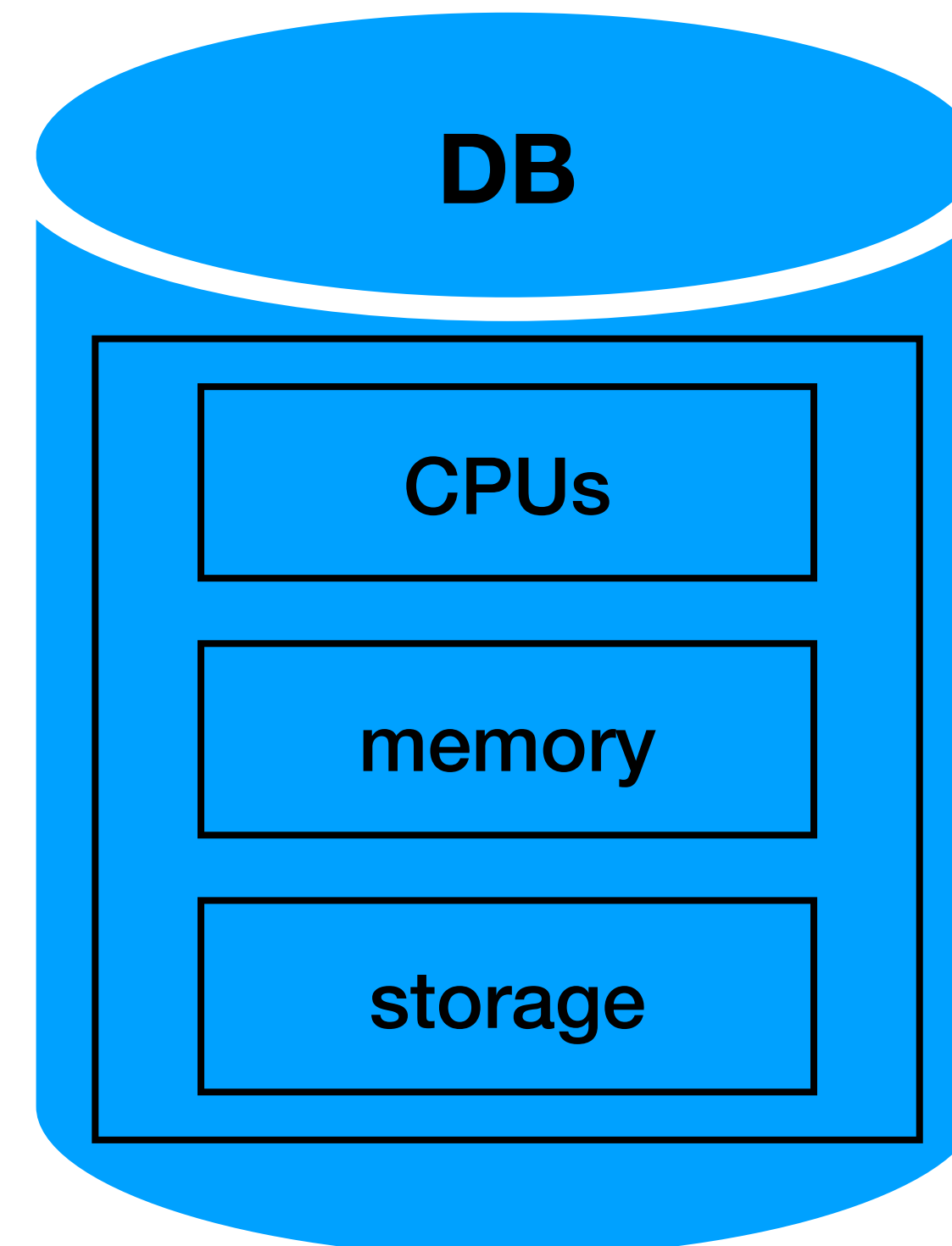
“single server”

Motivation

Centralized RDBMS (simplified)

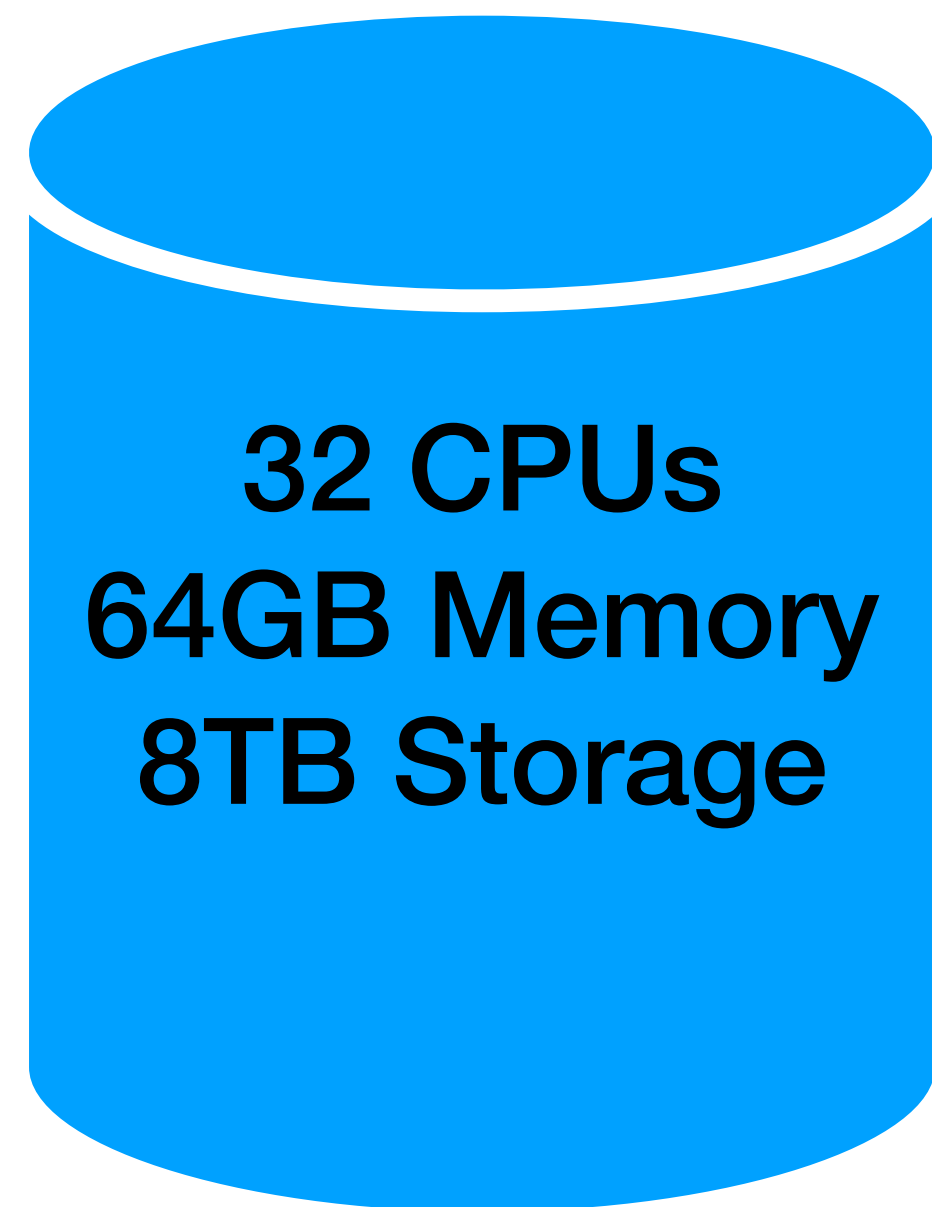
- What happens if we have more than 100m rows?
 - Storage
 - Query time
- What happens if the index is bigger than the memory?

Too many “page reads” can be slow

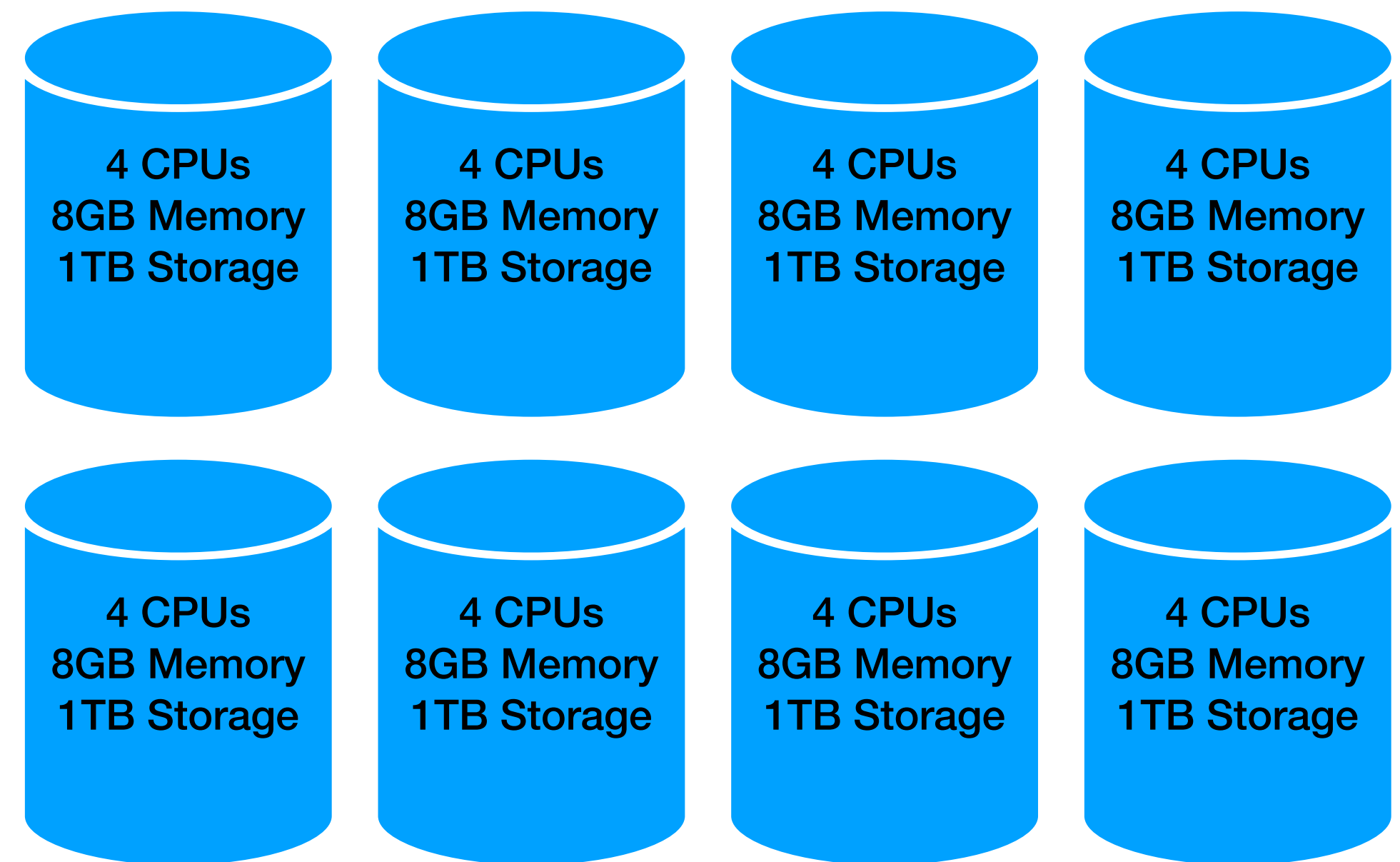


“single server”

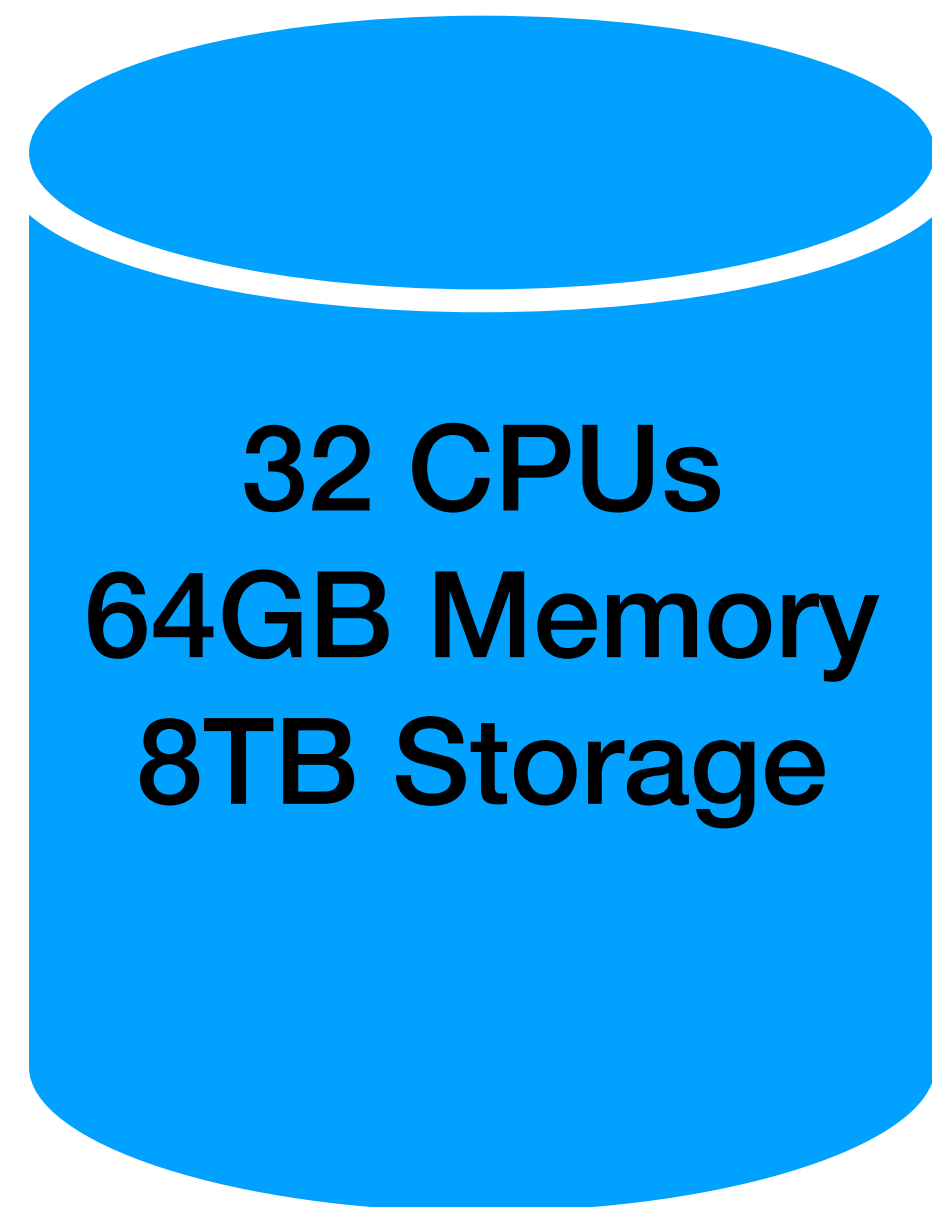
Scale up vs Scale out



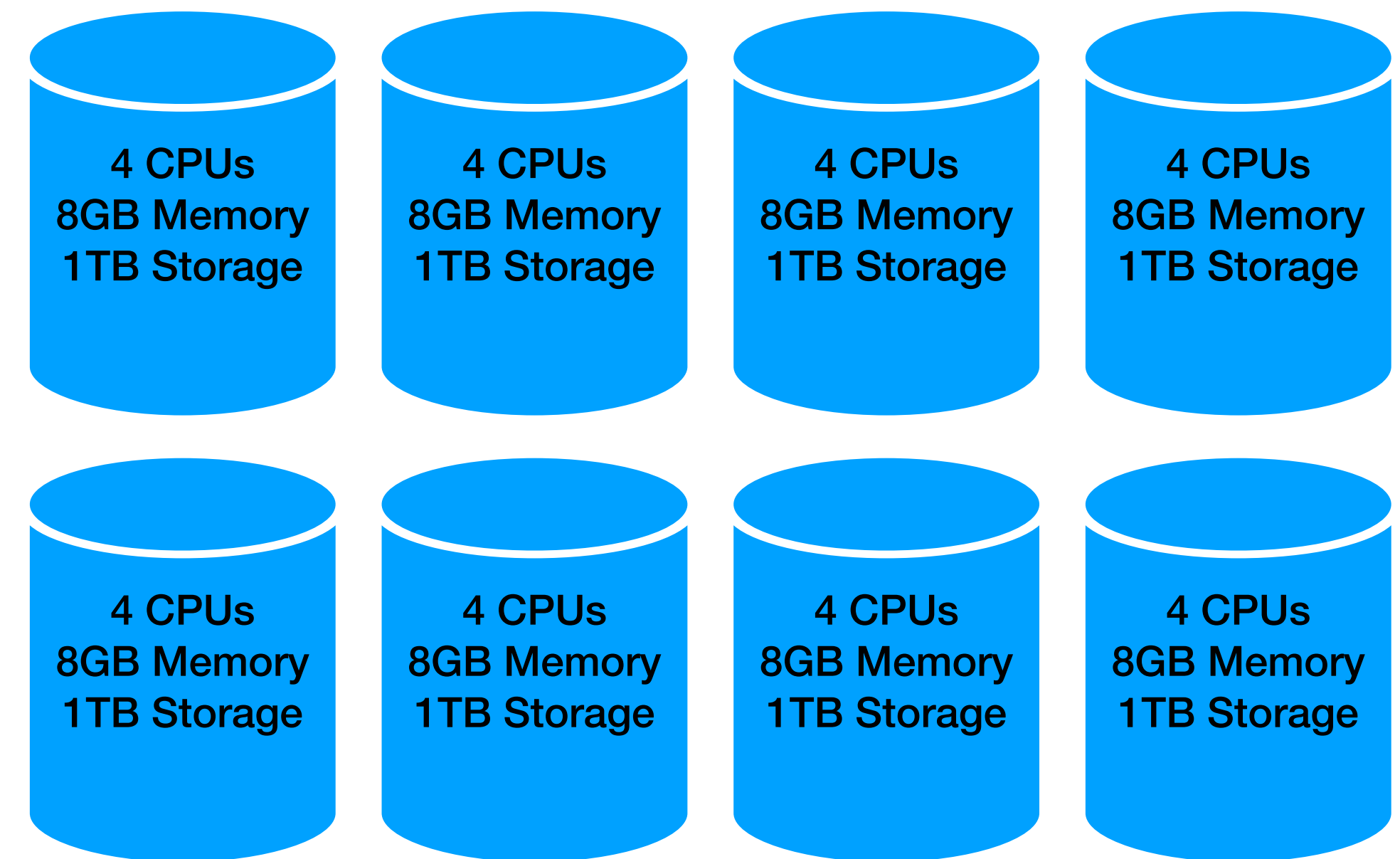
VS



Scale up vs Scale out



VS



Centralized / “Parallel” database

- scale has a limit
- “simple” management

Distributed database

- infinite scale
- “complex” management

RDBMS limitations

- Limited performance
vertical scale vs horizontal scale
- Data protection
no replication (*cloud storage layer with redundancy can be used)
- Up time
downtime on upgrades / hardware problems
- Cost
*resources are not elastic, pay even if unused

A note on distributed scaling (hot spots)

- Going distributed —> more problems
- In 2019 Jennifer Aniston joined Instagram and posted a single photo
- ...
- **Instagram crashed temporarily**
(much) more on that on “advance modeling”



Distributed relational database?

Distributed relational database

- Let's distribute a relational database
- How would you do it?

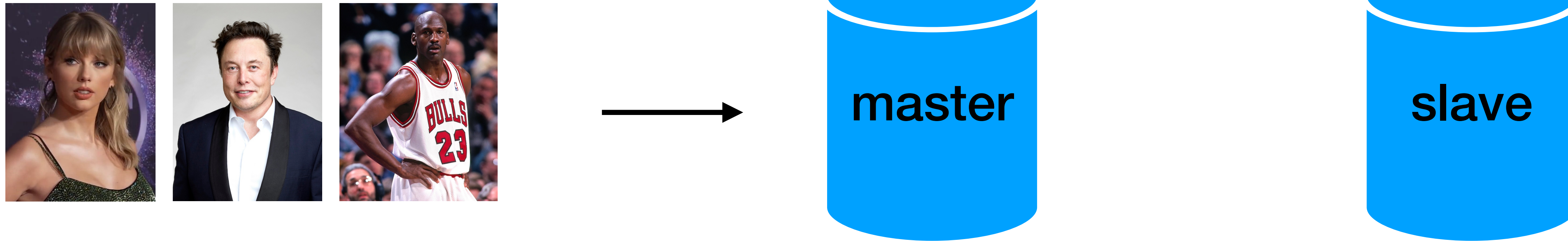
RDBMS master - slave

- On each write to the master we sync the slave
- If we detect an error, we transition to the slave



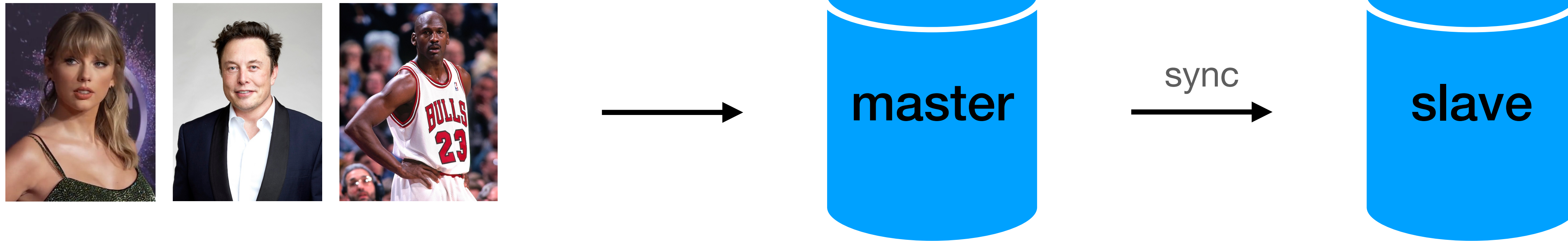
RDBMS master - slave

- On each write to the master we sync the slave
- If we detect an error, we transition to the slave



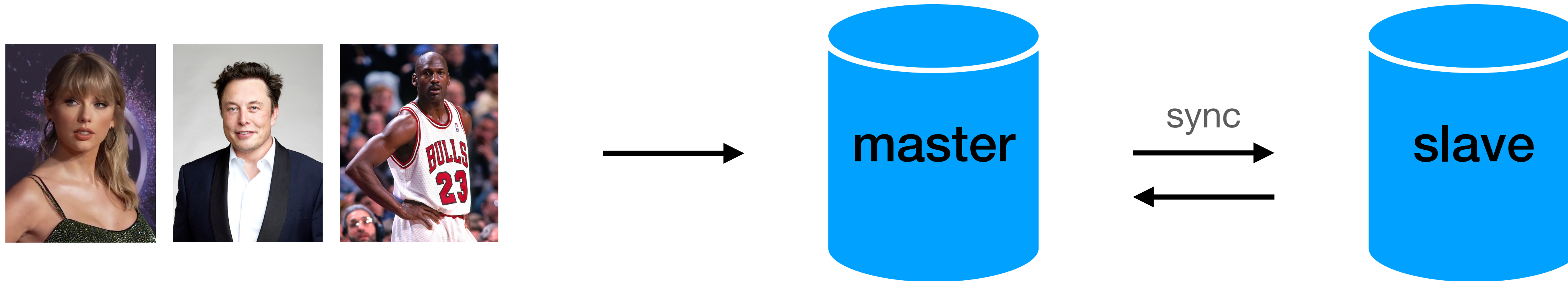
RDBMS master - slave

- On each write to the master we sync the slave
- If we detect an error, we transition to the slave



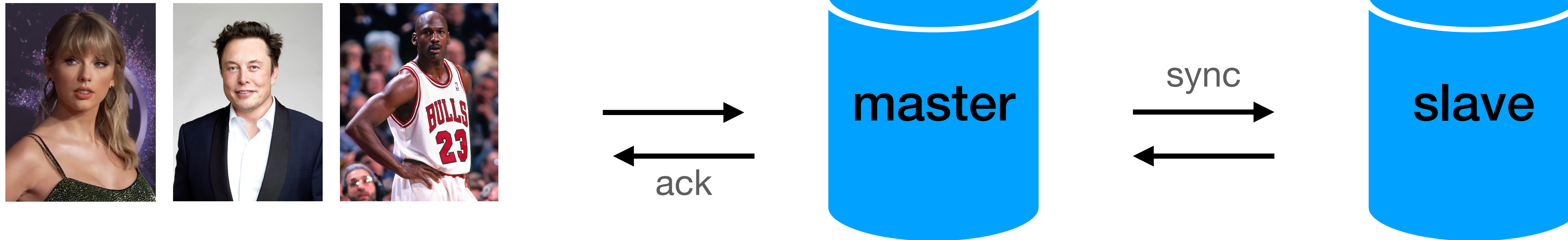
RDBMS master - slave

- On each write to the master we sync the slave
- If we detect an error, we transition to the slave



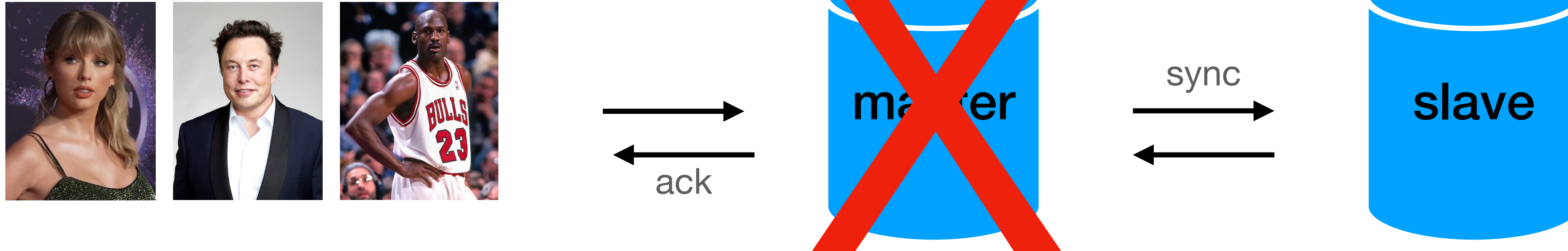
RDBMS master - slave

- On each write to the master we sync the slave
- If we detect an error, we transition to the slave



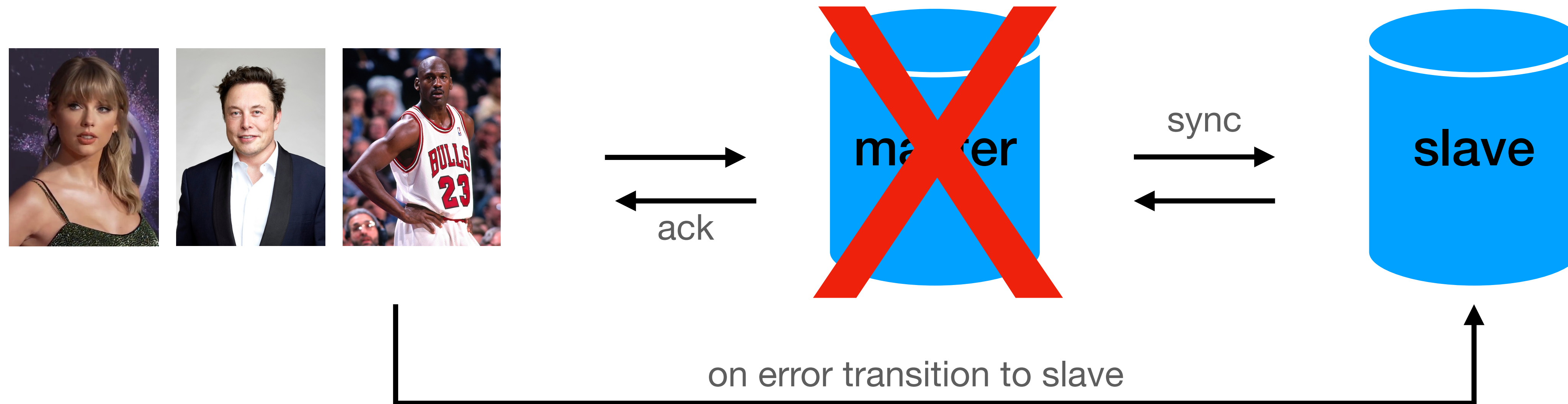
RDBMS master - slave

- On each write to the master we sync the slave
- If we detect an error, we transition to the slave



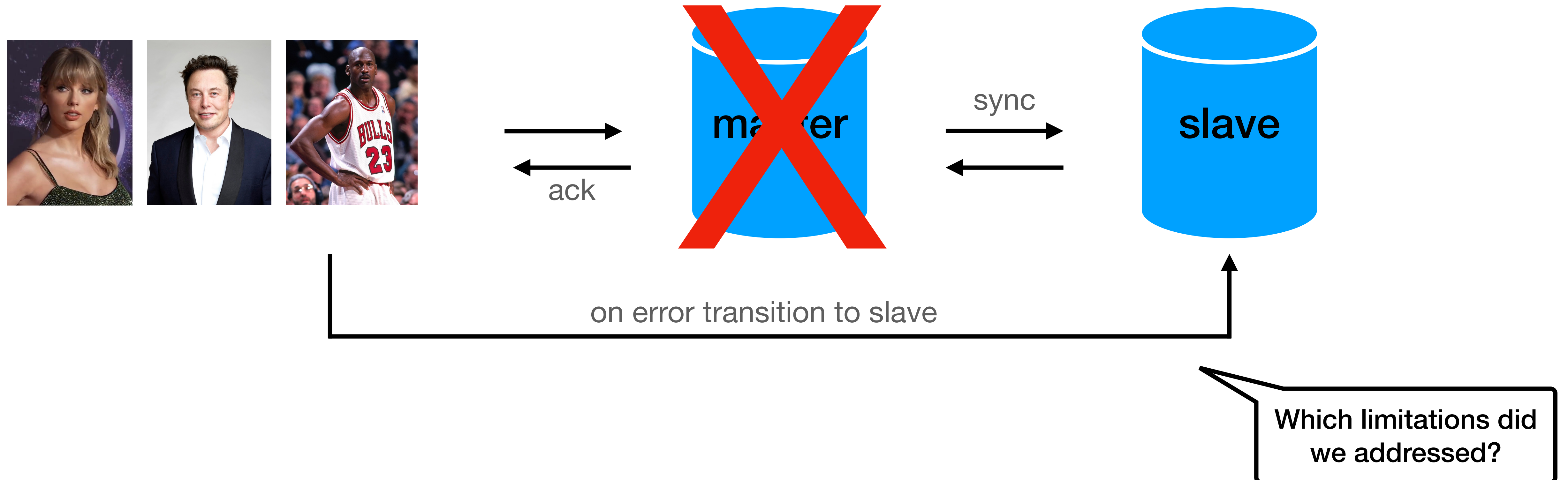
RDBMS master - slave

- On each write to the master we sync the slave
- If we detect an error, we transition to the slave



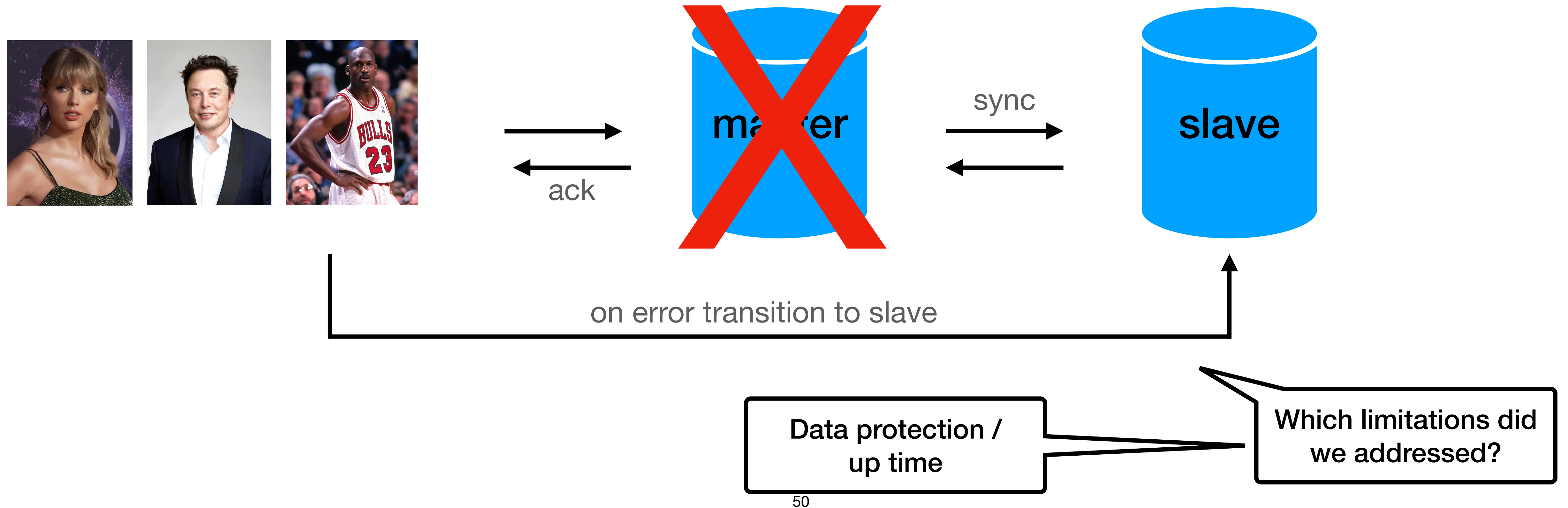
RDBMS master - slave

- On each write to the master we sync the slave
- If we detect an error, we transition to the slave



RDBMS master - slave

- On each write to the master we sync the slave
- If we detect an error, we transition to the slave



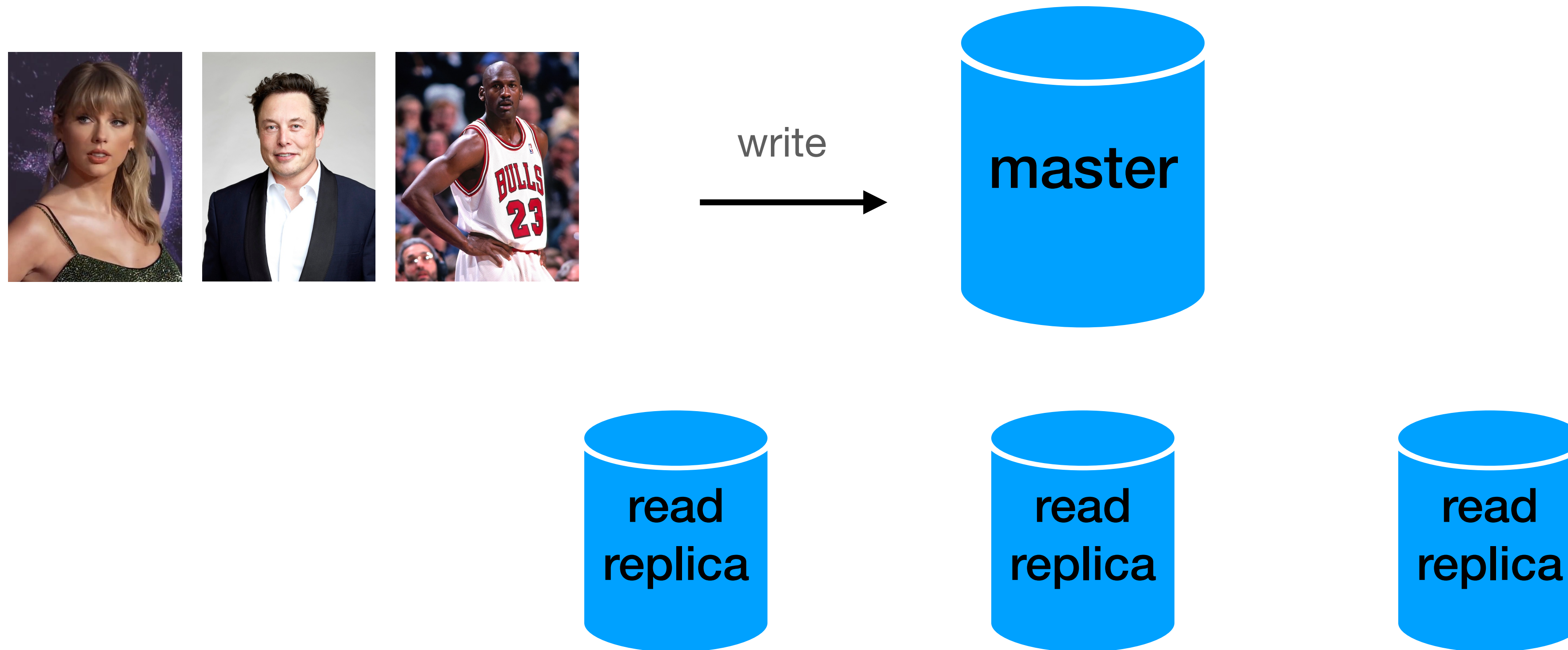
RDBMS read replicas

- On each write to the master we sync the replicas
- If we have a read query, we can use the replicas



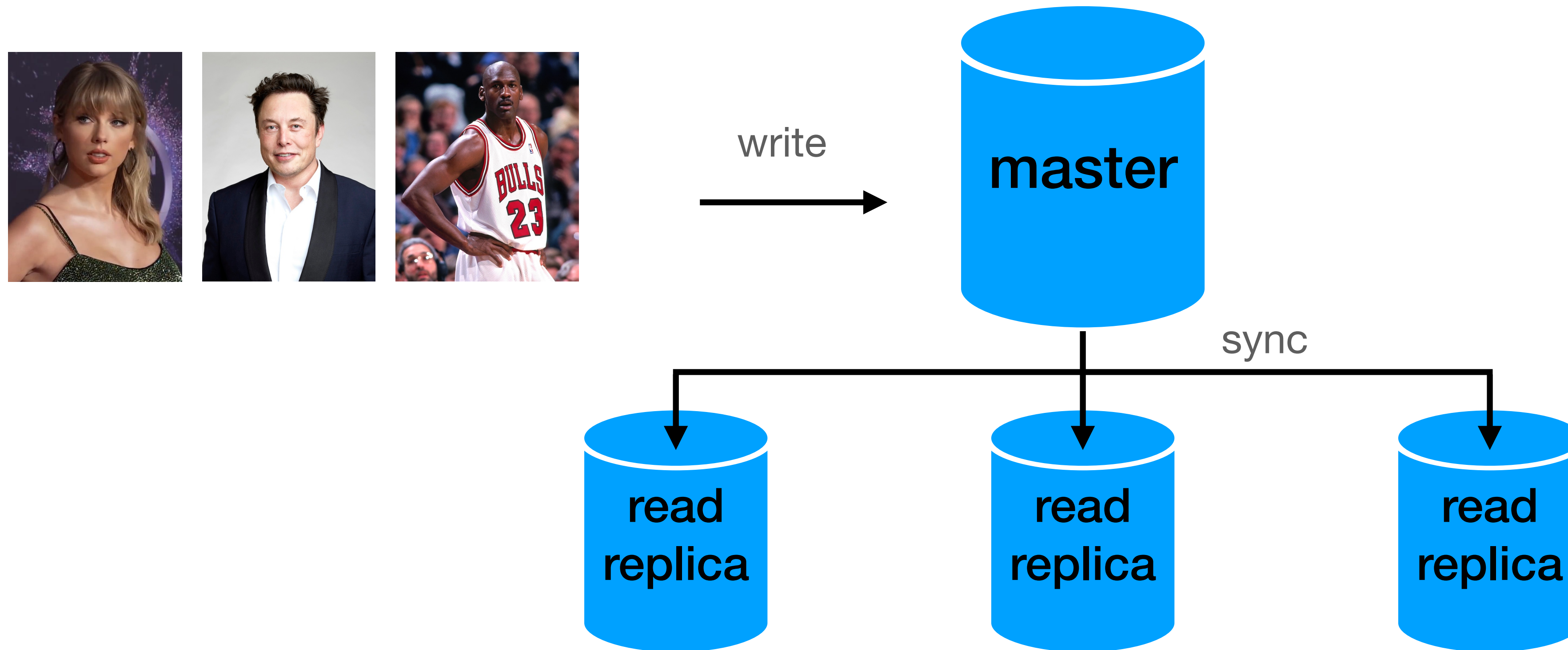
RDBMS read replicas

- On each write to the master we sync the replicas
- If we have a read query, we can use the replicas



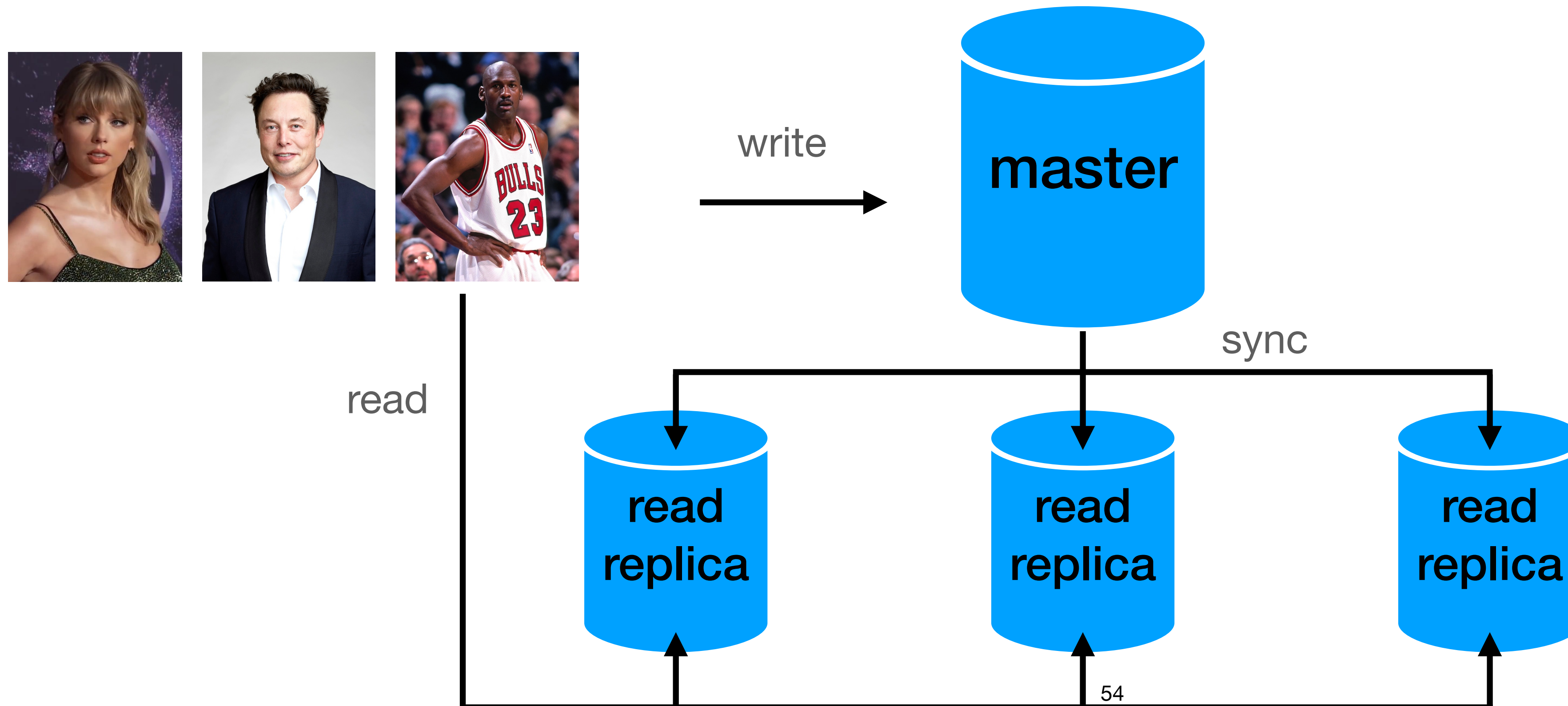
RDBMS read replicas

- On each write to the master we sync the replicas
- If we have a read query, we can use the replicas



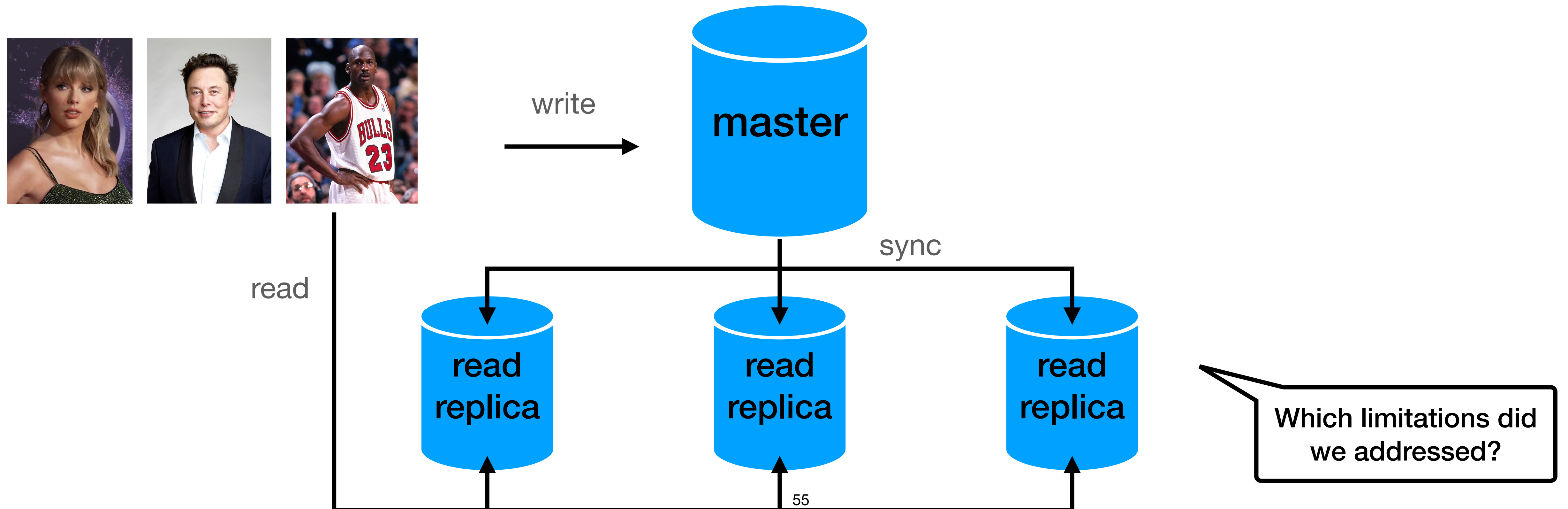
RDBMS read replicas

- On each write to the master we sync the replicas
- If we have a read query, we can use the replicas



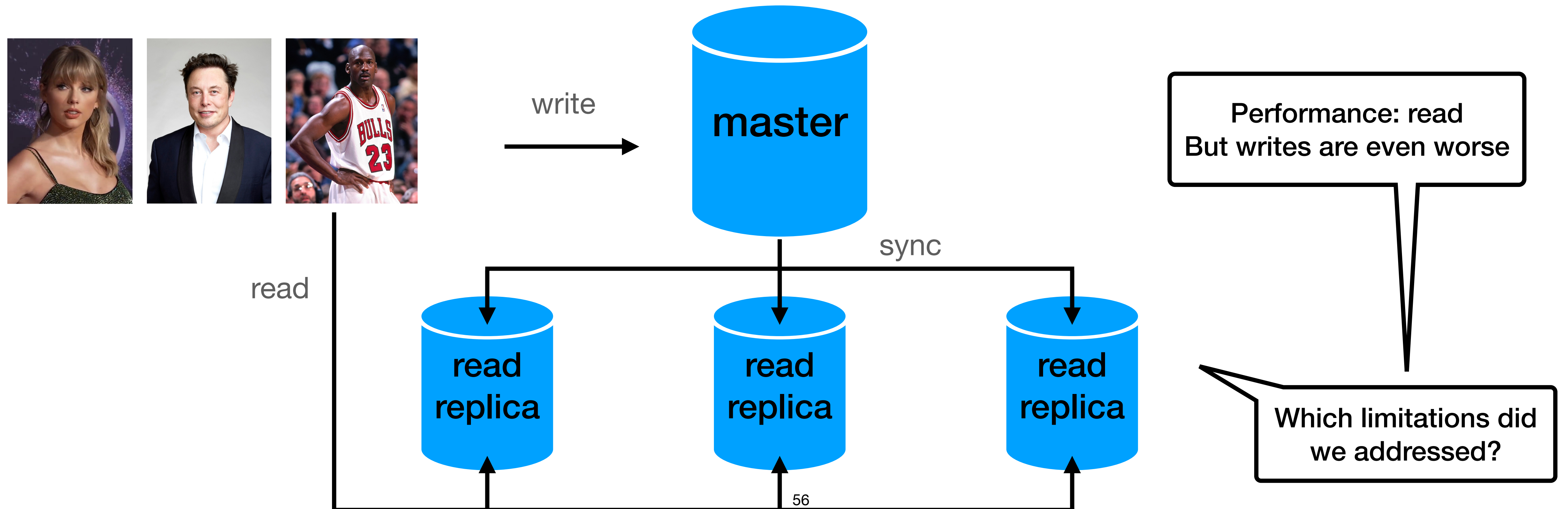
RDBMS read replicas

- On each write to the master we sync the replicas
- If we have a read query, we can use the replicas



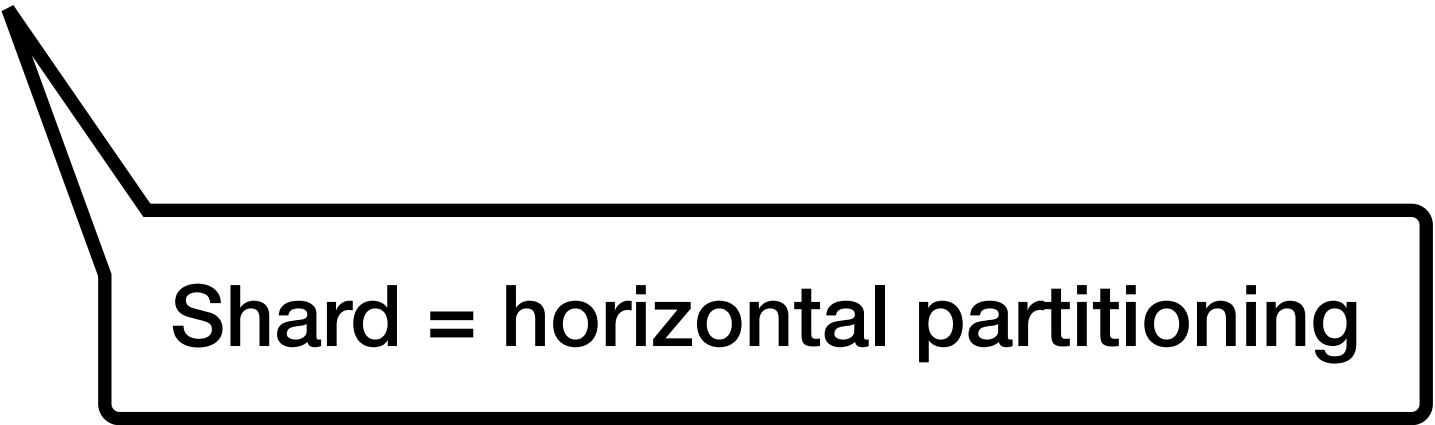
RDBMS read replicas

- On each write to the master we sync the replicas
- If we have a read query, we can use the replicas



Scaling RDBMS

- In practice, we use both methods
- Works great
especially for data protection and up time
- Performance - until a point as we can't scale the master writes
 - * we can use manual data sharding / fragmentation to a limit
- So what do we do?



Shard = horizontal partitioning

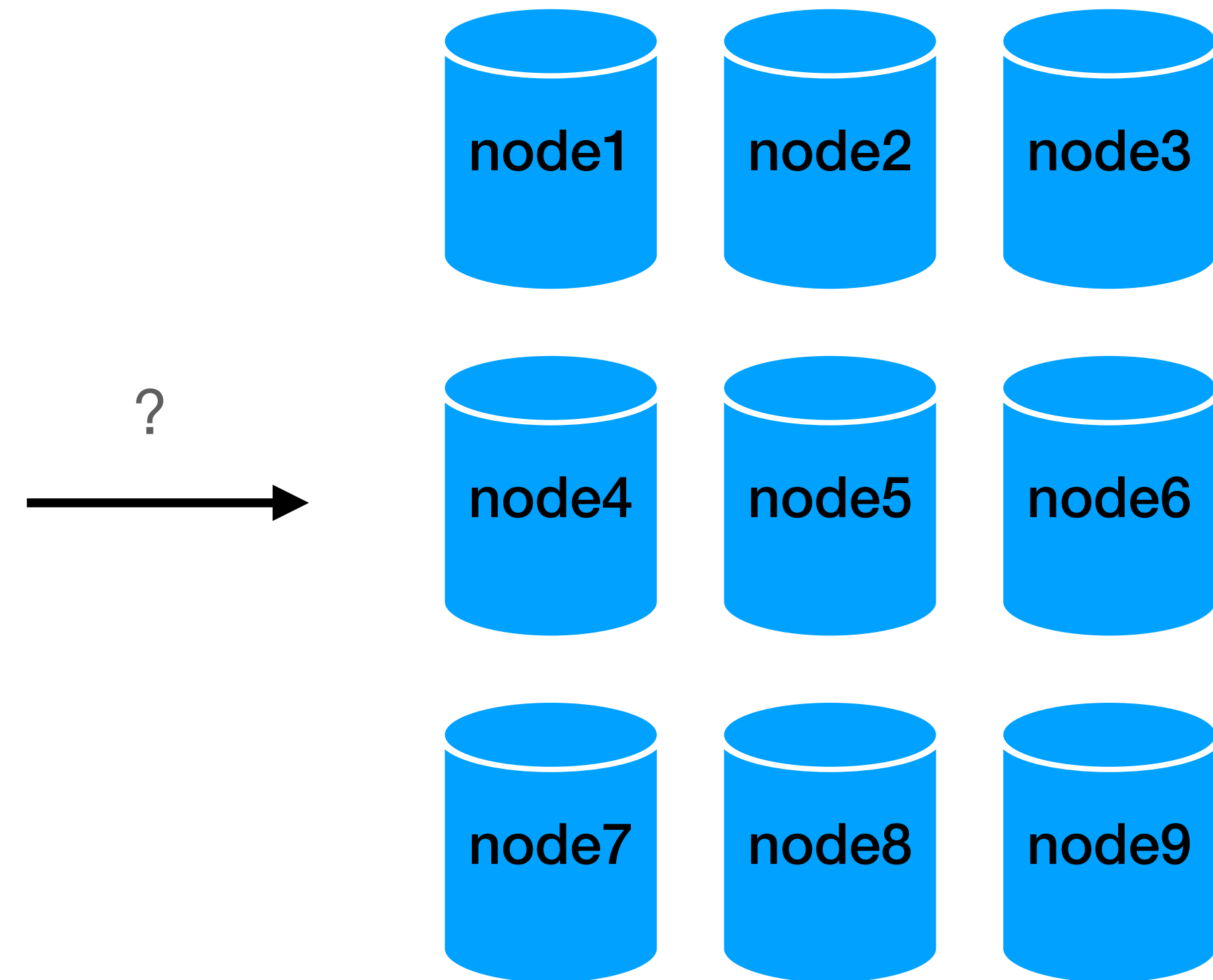
“Going distributed”

Disclaimer before going distributed

- RDBMS are great!
- Distributed databases do NOT replace RDBMS
- They are used for different use cases next to RDBMS

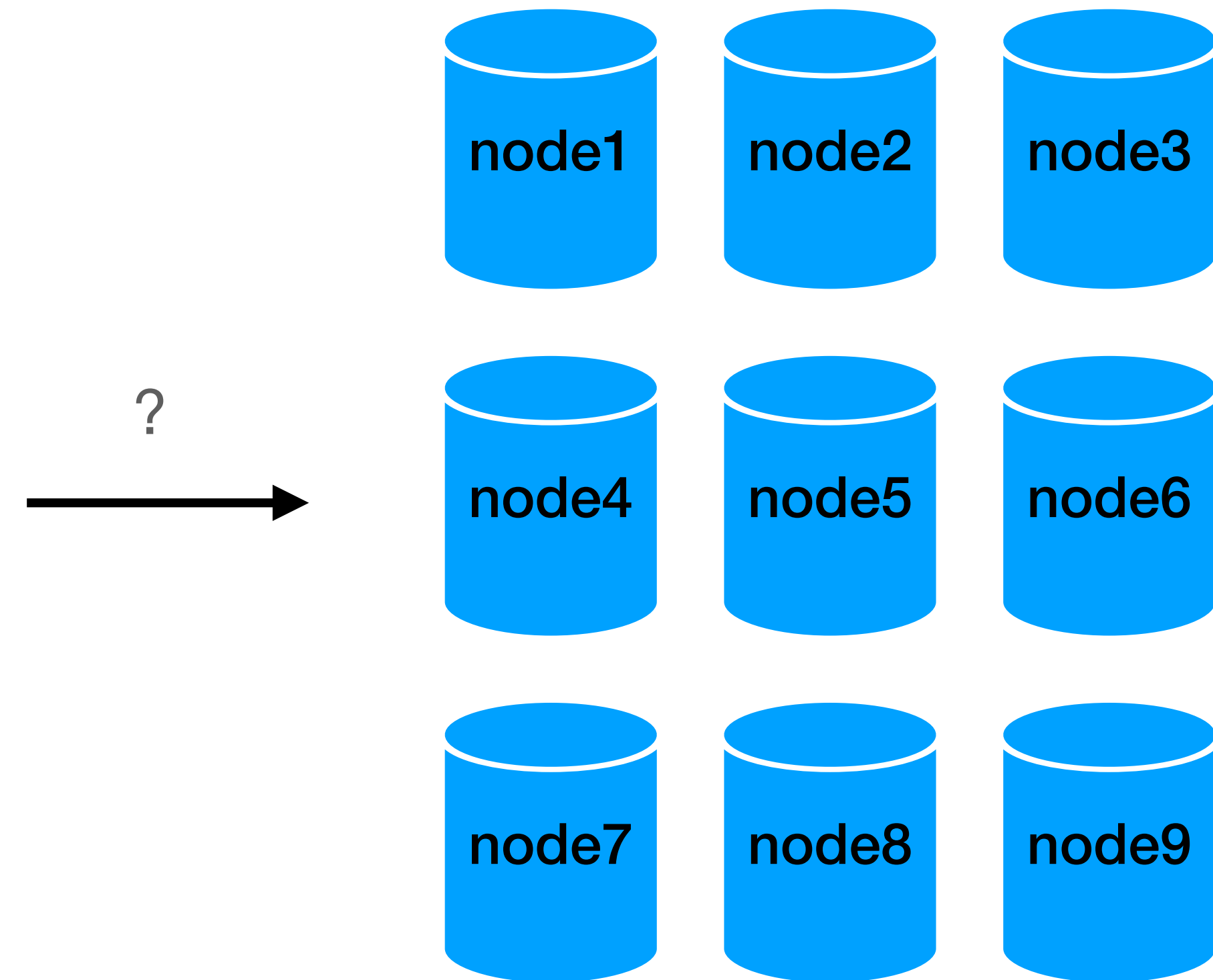
Going distributed

- **Not trivial... :)**
- Starting with:
 - Data fragmentation
 - Data distribution
 - Data replication



Going distributed

- **Not trivial... :)**
- Starting with:
 - **Data fragmentation**
 - Data distribution
 - Data replication



Data fragmentation

- How can we “break” a relation?
- How to implement reads / writes?

<u>user_id</u>	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

Horizontal fragmentation

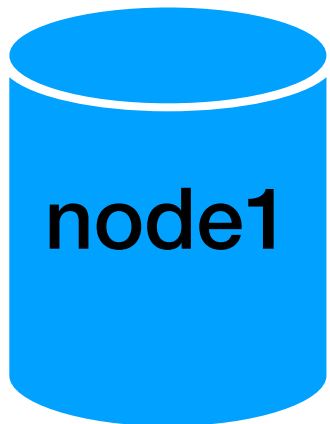
<u>user_id</u>	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

- Choose an attribute
- Assign a “range” to each “node”

Horizontal fragmentation

<u>user id</u>	fname	Iname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

- Choose an attribute
- Assign a “range” to each “node”



<u>user id</u>	fname	Iname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

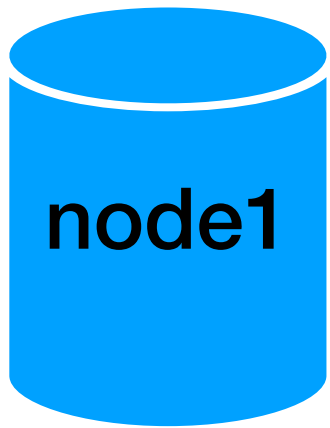


<u>user id</u>	fname	Iname	city	country	account	brithdate
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984

Horizontal fragmentation

<u>user id</u>	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

- Choose an attribute
- Assign a “range” to each “node”



<u>user id</u>	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

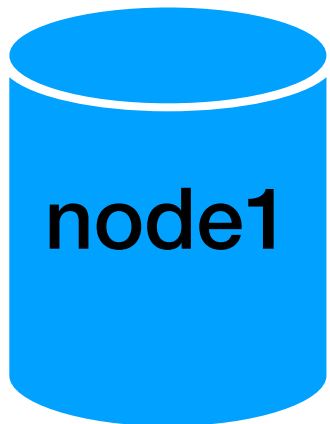


<u>user id</u>	fname	lname	city	country	account	brithdate
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984

Horizontal fragmentation

<u>user_id</u>	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

- Choose an attribute
- Assign a “range” to each “node”



<u>user_id</u>	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963



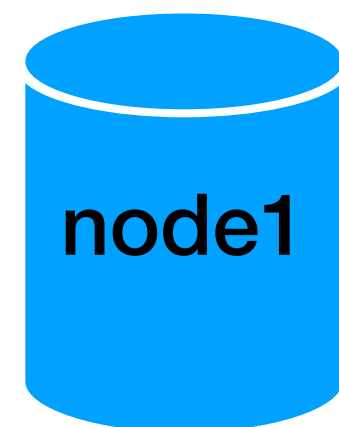
<u>user_id</u>	fname	lname	city	country	account	brithdate
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984

how do we implement reads and writes?

Horizontal fragmentation

<u>user_id</u>	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

- Choose an attribute
- Assign a “range” to each “node”



<u>user_id</u>	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963



<u>user_id</u>	fname	lname	city	country	account	brithdate
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984

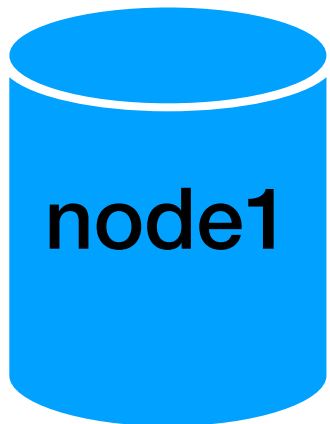
how do we implement reads and writes?

Is “SELECT * FROM users WHERE user_id = 101” valid?
How the DB finds the right node?

Horizontal fragmentation

<u>user_id</u>	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

- Choose an attribute
- Assign a “range” to each “node”



<u>user_id</u>	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963



<u>user_id</u>	fname	lname	city	country	account	brithdate
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984

what will be the range if we choose
user_id or birthdate?

Vertical fragmentation

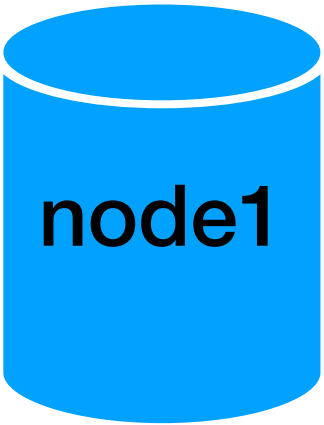
- Choose attributes
- Assign a “attributes” to each “node”

<u>user_id</u>	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

Vertical fragmentation

<u>user_id</u>	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

- Choose attributes
- Assign a “attributes” to each “node”



node1

<u>user_id</u>	fname	lname
101	Rubi	Boim
102	Tova	Milo
103	Lebron	James
104	Michael	Jordan



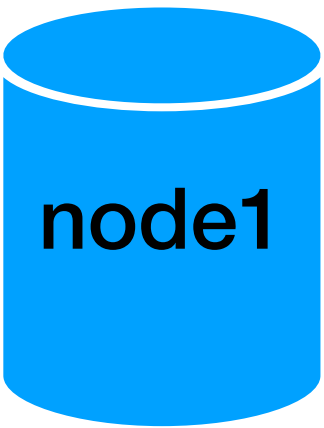
node2

<u>user_id</u>	city	country	account	brithdate
101	Tel Aviv	Israel	Normal	<null>
102	Tel Aviv	Israel	Premium	<null>
103	Los Angeles	USA	Premium	30/12/1984
104	Chicago	USA	Normal	17/02/1963

Vertical fragmentation

<u>user_id</u>	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

- Choose attributes
- Assign a “attributes” to each “node”



<u>user_id</u>	fname	lname
101	Rubi	Boim
102	Tova	Milo
103	Lebron	James
104	Michael	Jordan



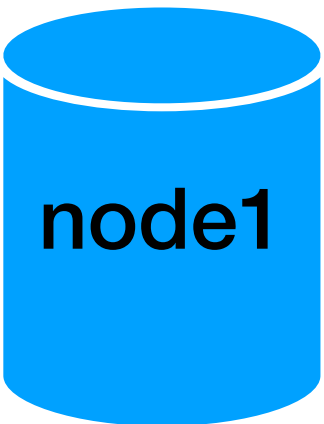
<u>user_id</u>	city	country	account	brithdate
101	Tel Aviv	Israel	Normal	<null>
102	Tel Aviv	Israel	Premium	<null>
103	Los Angeles	USA	Premium	30/12/1984
104	Chicago	USA	Normal	17/02/1963

how do we implement reads and writes?

Vertical fragmentation

<u>user_id</u>	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

- Choose attributes
- Assign a “attributes” to each “node”



<u>user_id</u>	fname	lname
101	Rubi	Boim
102	Tova	Milo
103	Lebron	James
104	Michael	Jordan

What happen if the key is defined by 3 different attributes?



<u>user_id</u>	city	country	account	brithdate
101	Tel Aviv	Israel	Normal	<null>
102	Tel Aviv	Israel	Premium	<null>
103	Los Angeles	USA	Premium	30/12/1984
104	Chicago	USA	Normal	17/02/1963

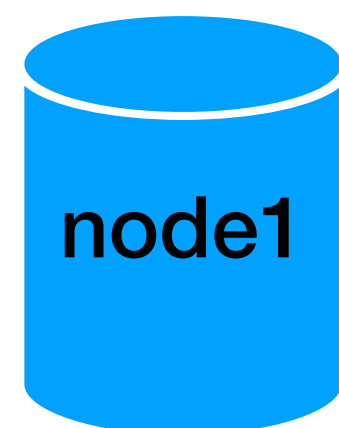
Vertical fragmentation

<u>user_id</u>	fname	lname	city	country	account	birthdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

SIDE NOTE

- There is a class of databases that is called “**columnar database**”
- Used mainly in data warehouse

<u>user_id</u>	fname
101	Rubi
102	Tova
103	Lebron
104	Michael

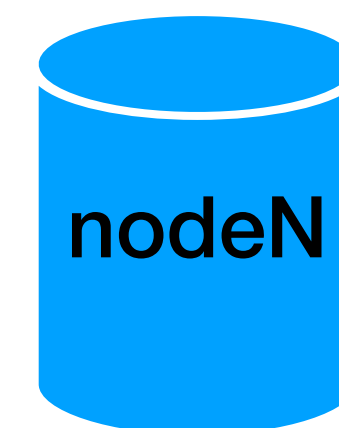


<u>user_id</u>	lname
101	Boim
102	Milo
103	James
104	Jordan



...

<u>user_id</u>	birthdate
101	<null>
102	<null>
103	30/12/1984
104	17/02/1963



Vertical fragmentation

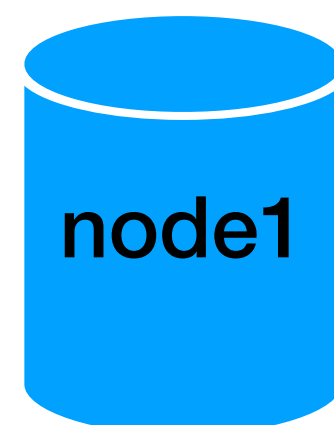
<u>user_id</u>	fname	lname	city	country	account	birthdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

SIDE NOTE

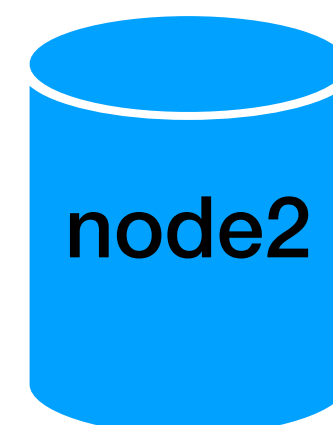
- There is a class of databases that is called “**columnar database**”
- Used mainly in data warehouse

Can be highly compressible.
Question - for which attributes?

<u>user_id</u>	fname
101	Rubi
102	Tova
103	Lebron
104	Michael

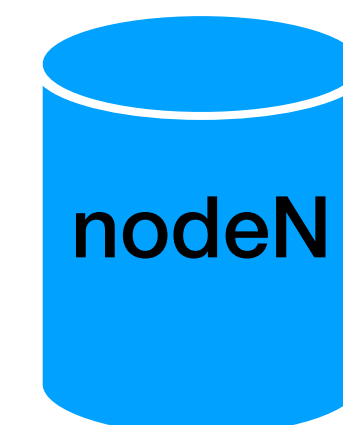


<u>user_id</u>	lname
101	Boim
102	Milo
103	James
104	Jordan



...

<u>user_id</u>	birthdate
101	<null>
102	<null>
103	30/12/1984
104	17/02/1963



Vertical fragmentation

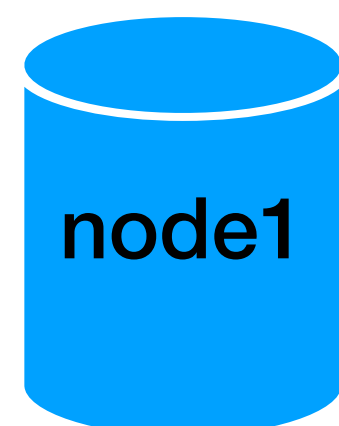
<u>user_id</u>	fname	lname	city	country	account	birthdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

SIDE NOTE

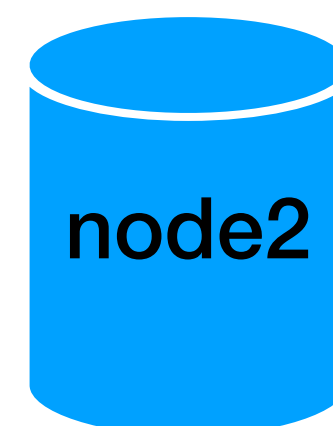
- There is a class of databases that is called “**columnar database**”
- Used mainly in data warehouse

Popular databases:
Redshift, BigQuery, Apache Druid, Maria DB

<u>user_id</u>	fname
101	Rubi
102	Tova
103	Lebron
104	Michael

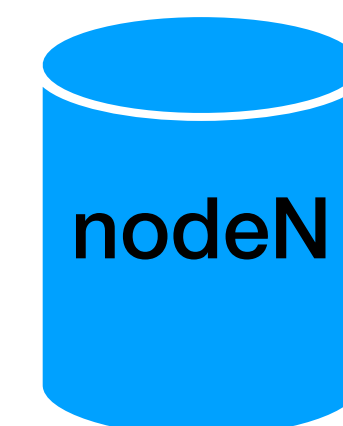


<u>user_id</u>	lname
101	Boim
102	Milo
103	James
104	Jordan



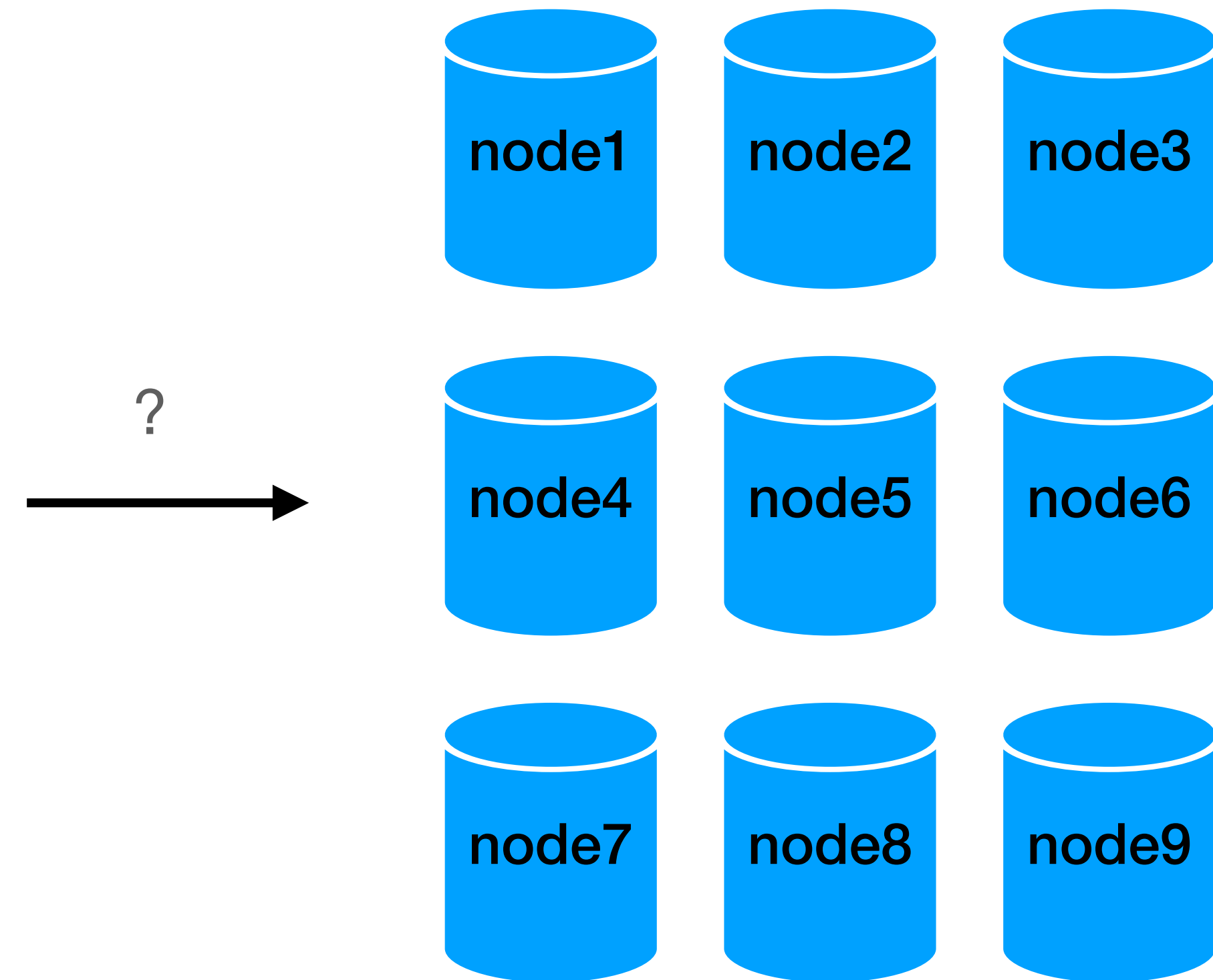
...

<u>user_id</u>	birthdate
101	<null>
102	<null>
103	30/12/1984
104	17/02/1963



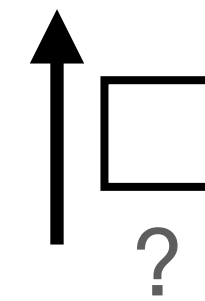
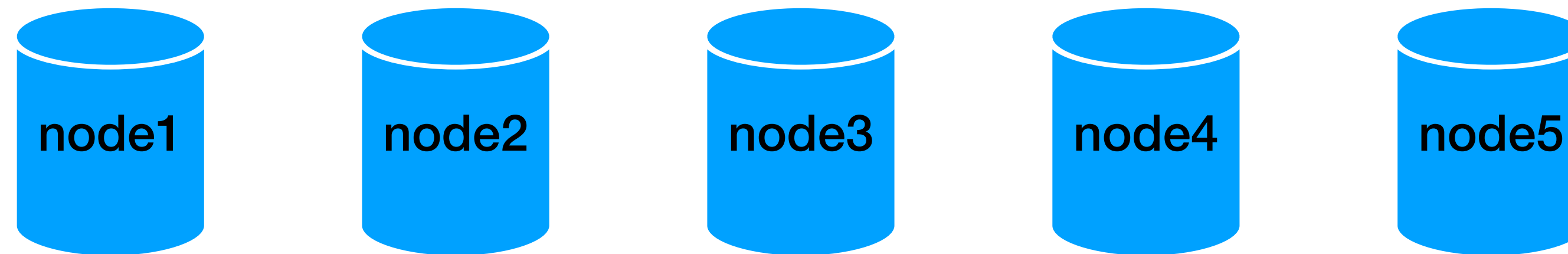
Going distributed

- **Not trivial... :)**
- Starting with:
 - Data fragmentation
 - **Data distribution**
 - Data replication



Data distribution

- How can the DB decide where the data is located?



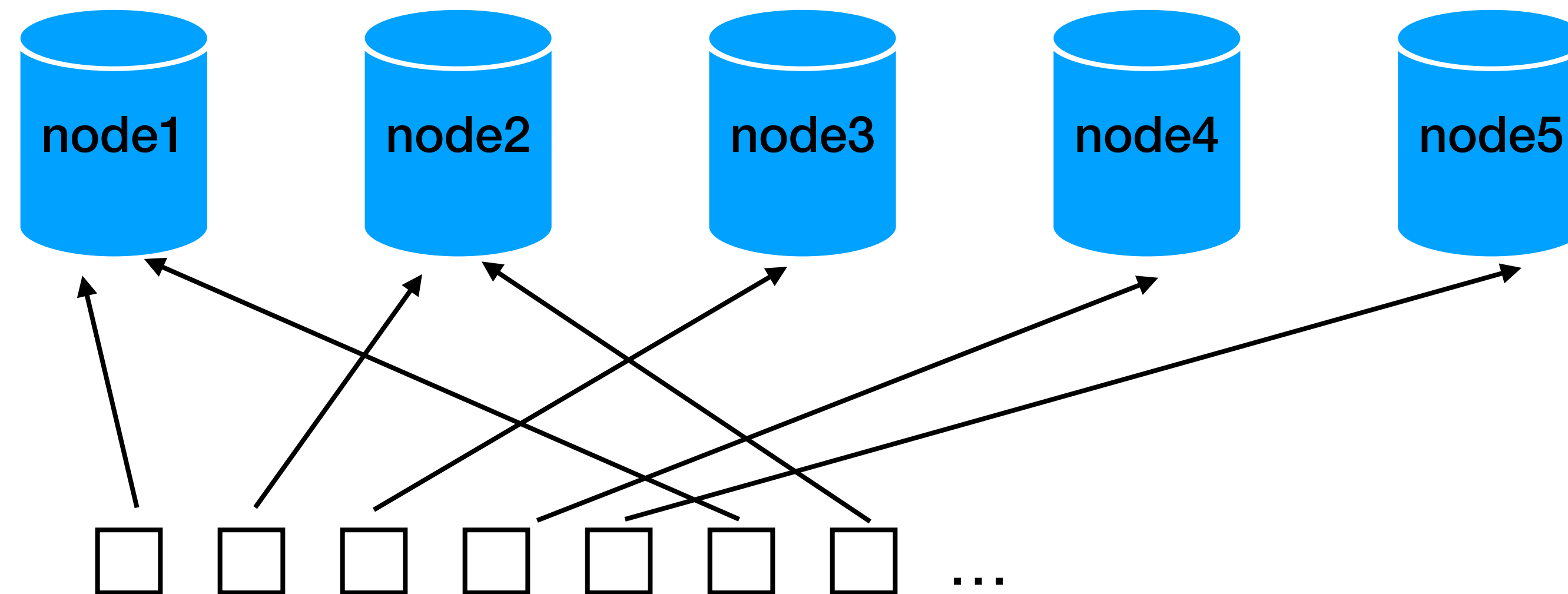
INSERT INTO users VALUES(x,y,z)



add new data /
query existing data

Data distribution (1) - Round robin

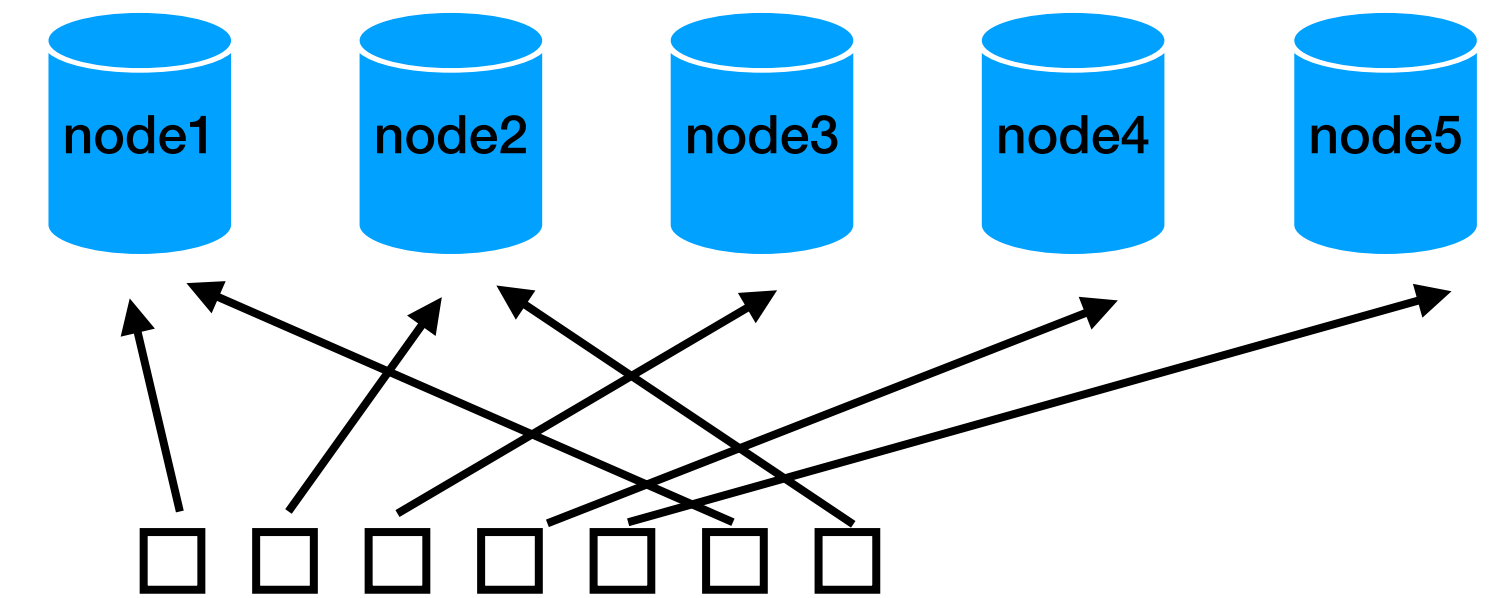
- Or random...



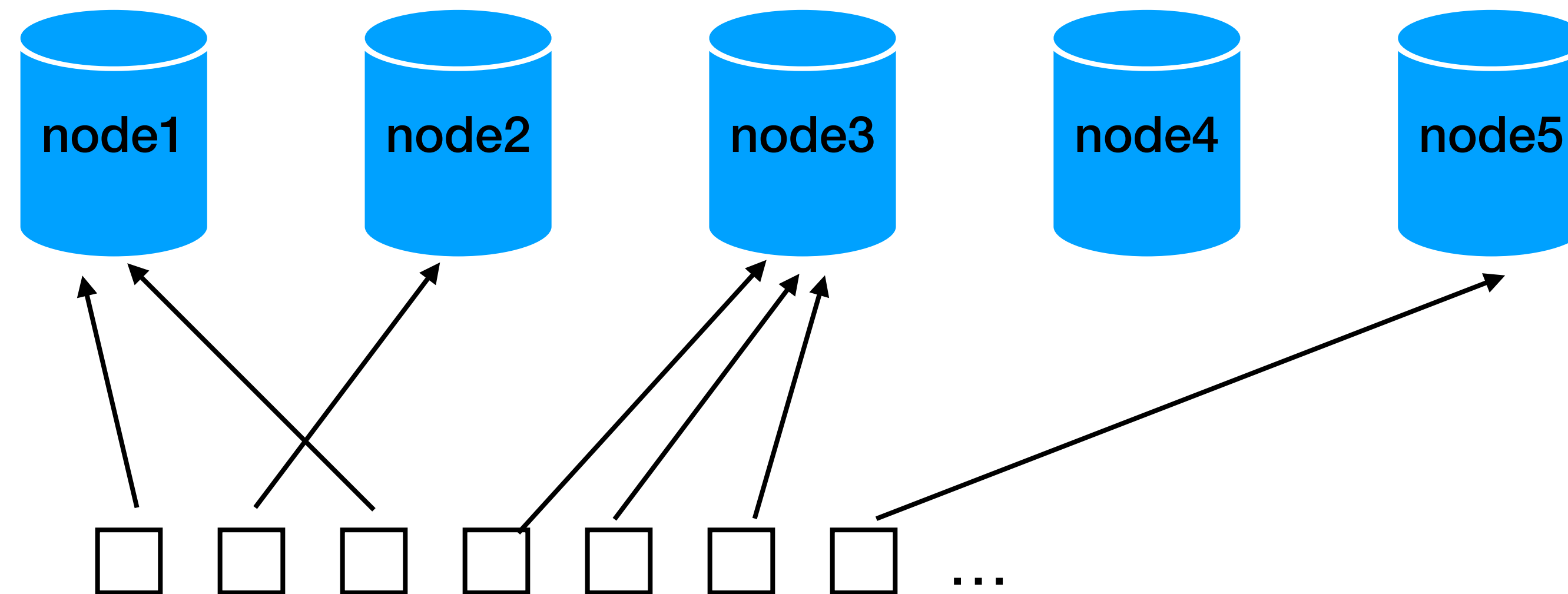
pros / cons?

Data distribution (1) - Round robin

- ☑ “Uniform” distribution
for both CPU and storage (for write)
- ☐ The “driver” needs to save the last node for each different query
- ☐ To query we need to save the node “used” for each “package”
- we need a DB to implement a DB...

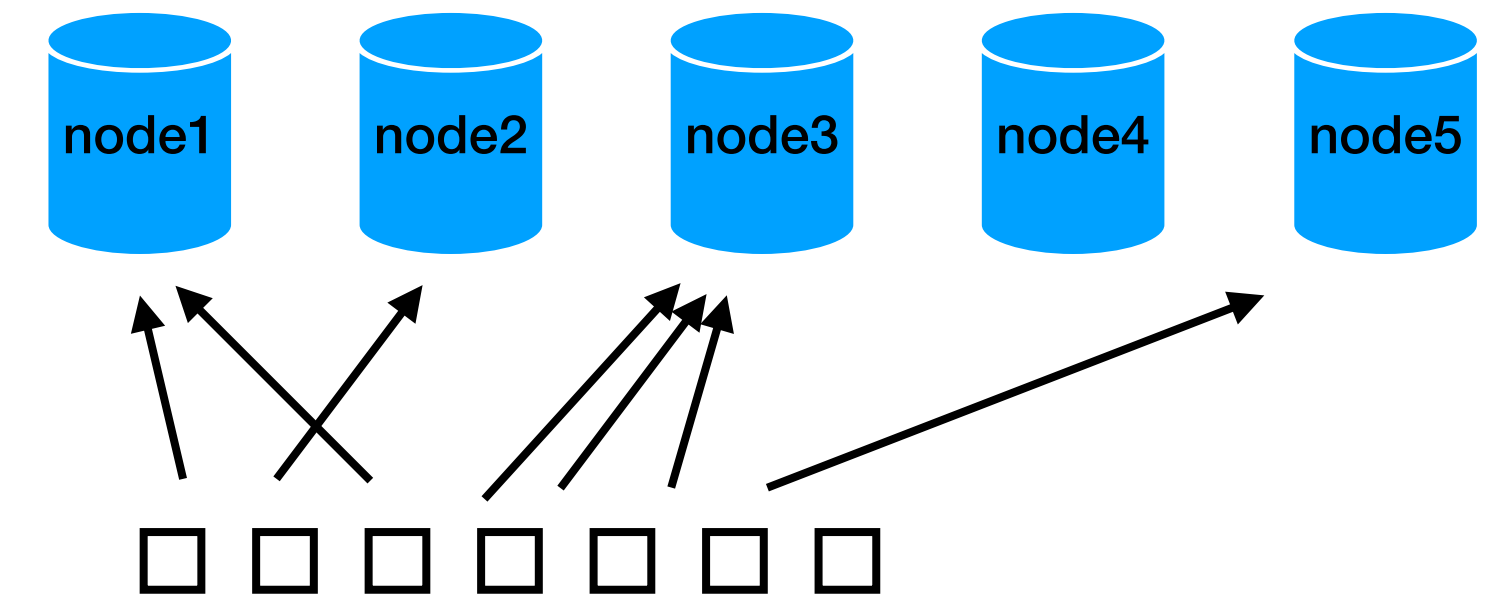


Data distribution (2) - Hashing



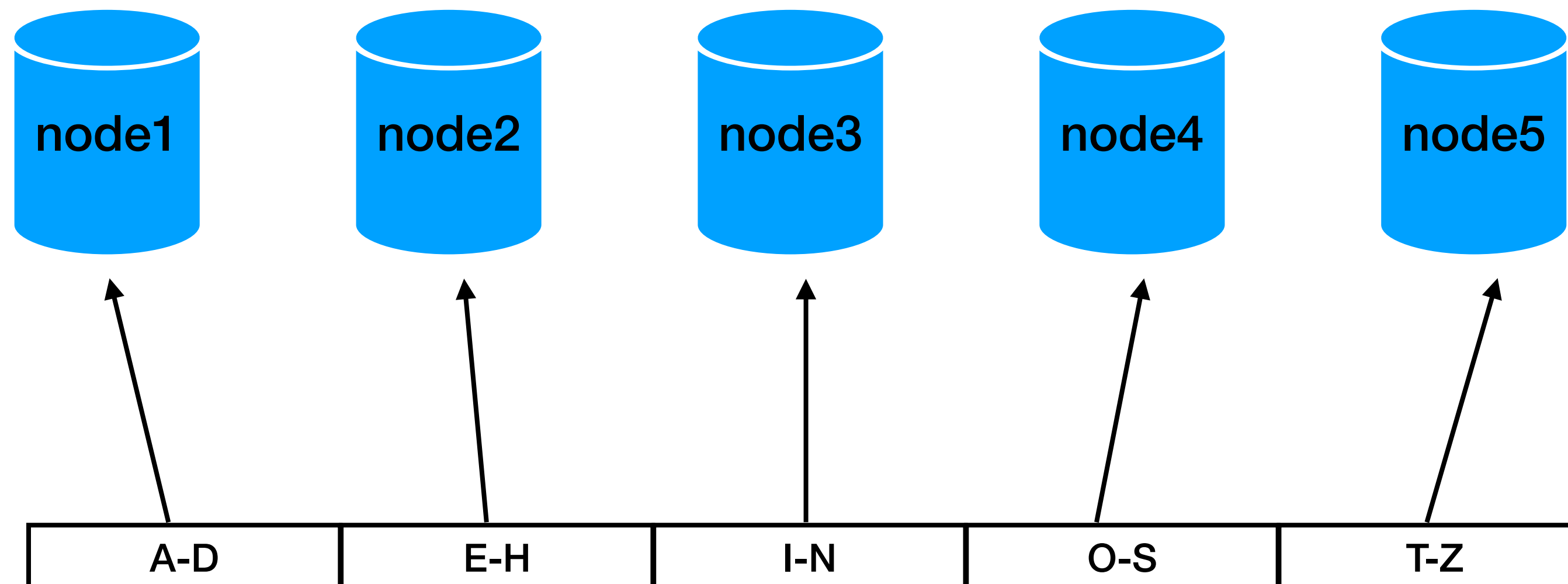
pros / cons?

Data distribution (2) - Hashing



- Similar to Round robin:
To query we need to save the node “used” for each “package” (or hash)
- we need a DB to implement a DB...

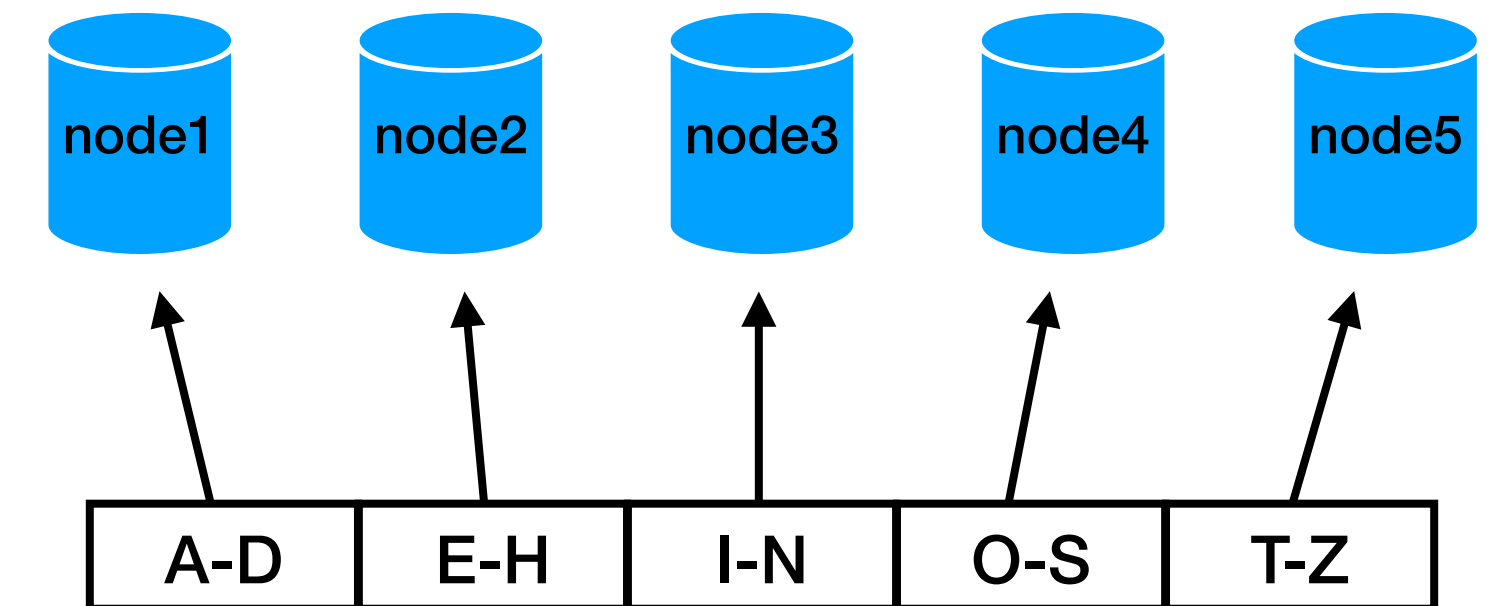
Data distribution (3) - Range



pros / cons?

Data distribution (3) - Range

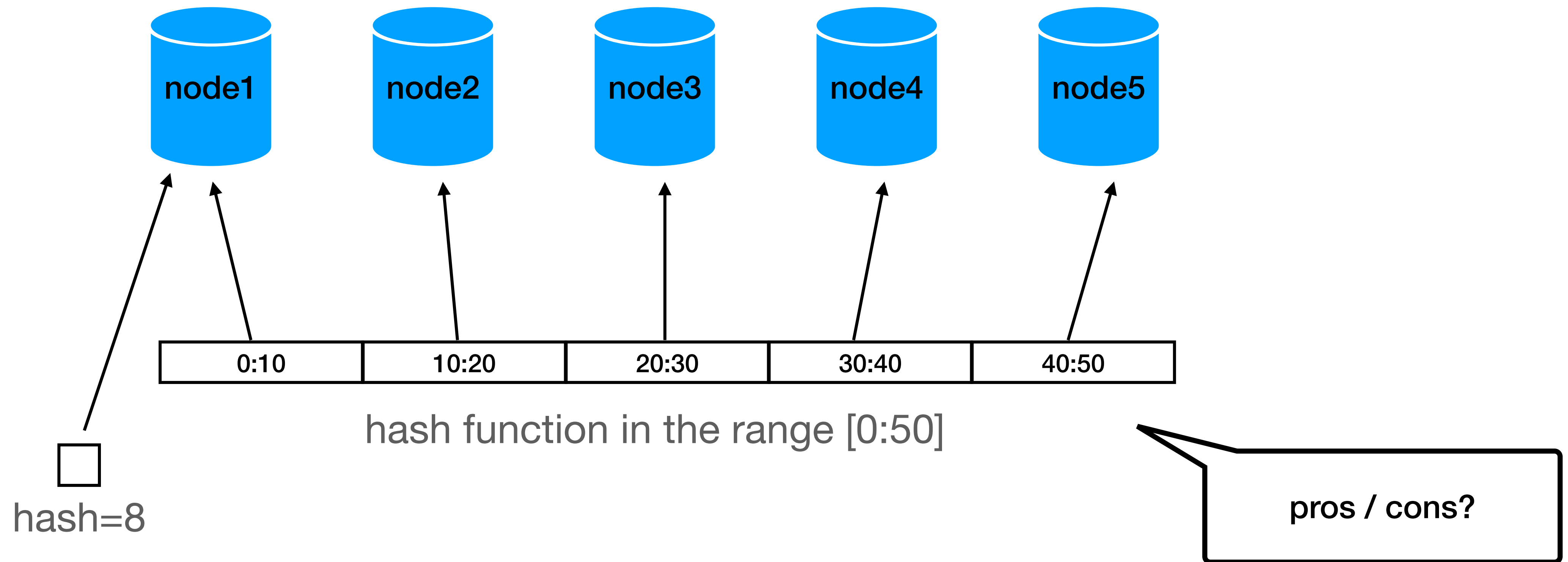
- ☑ Only the range mapping is needed to be saved



- ☑ No need for an “extra DB”
but we still need to store them
(very little information - can be cached on the driver)

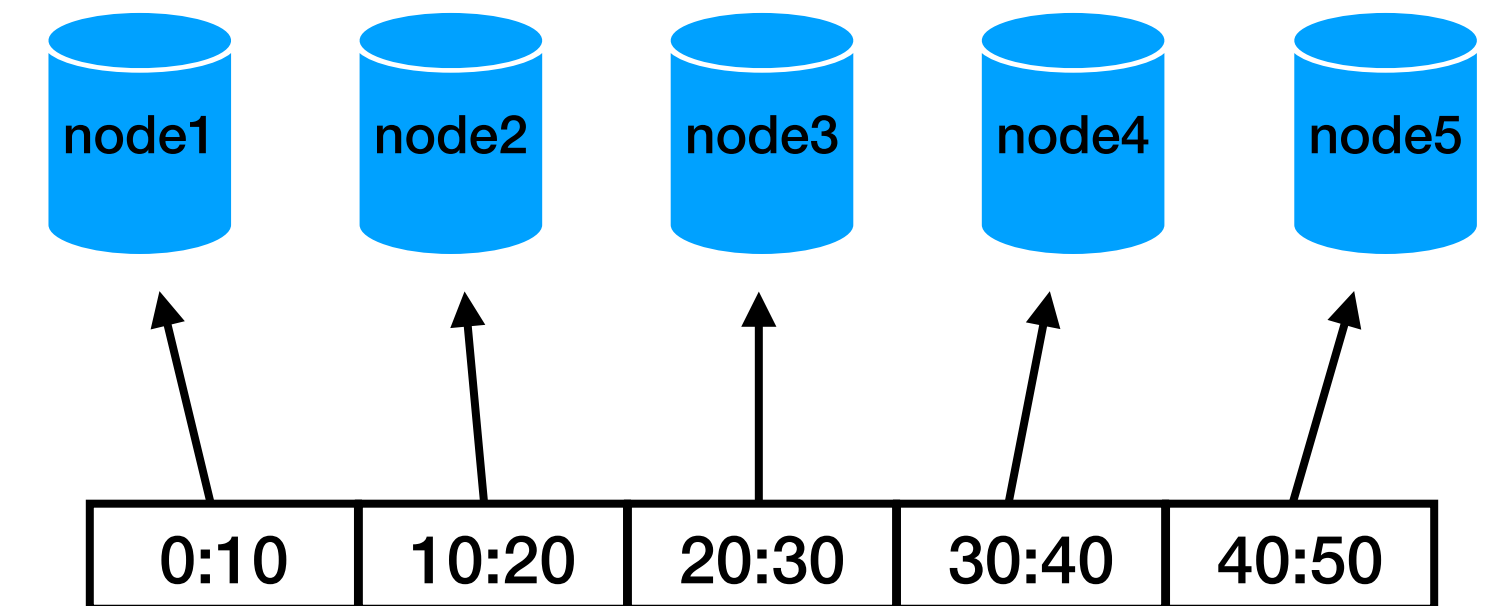
- ☐ Works differently for strings, dates, int...
- ☐ Data distribution
most names starts with “A” for example

Data distribution (4) - Range on hashes



Data distribution (4) - Range on hashes

- ☑ Only the range mapping is needed to be saved

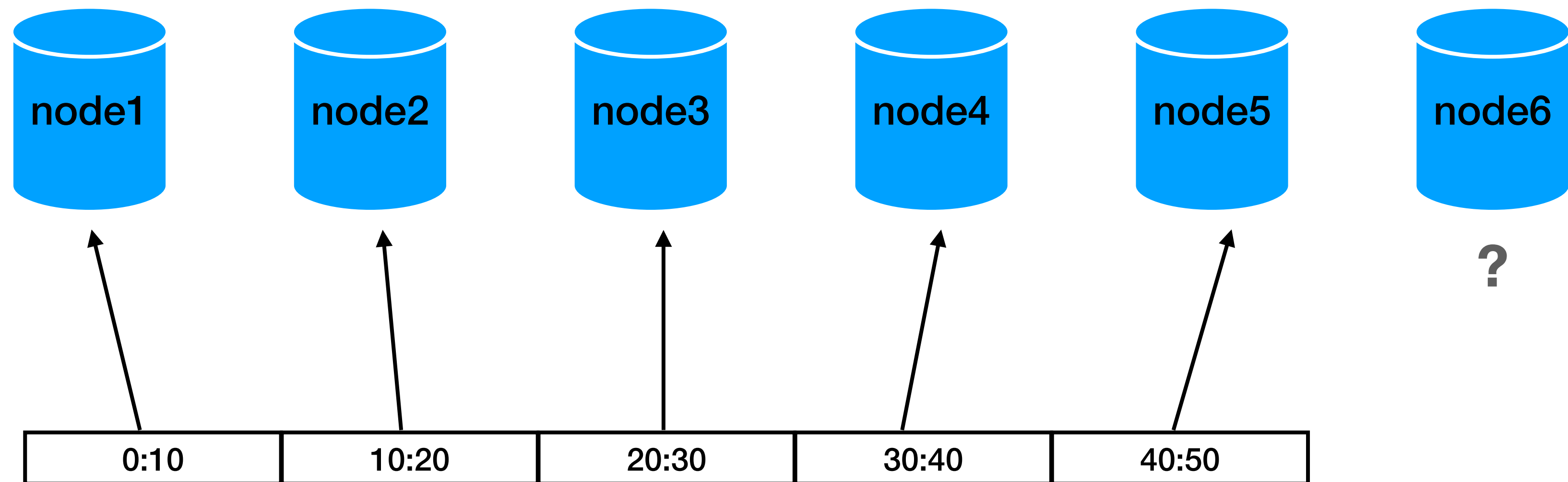


- ☑ No need for an “extra DB”
but we still need to store them
(very little information - can be cached on the driver)

- ☑ Data agnostic
assuming the hash function works properly
- ☑ Data distribution
assuming the hash function works properly

Data distribution - scaling

- What happens if we want to add a node?
 - new data?
 - existing data?



Data distribution - scaling

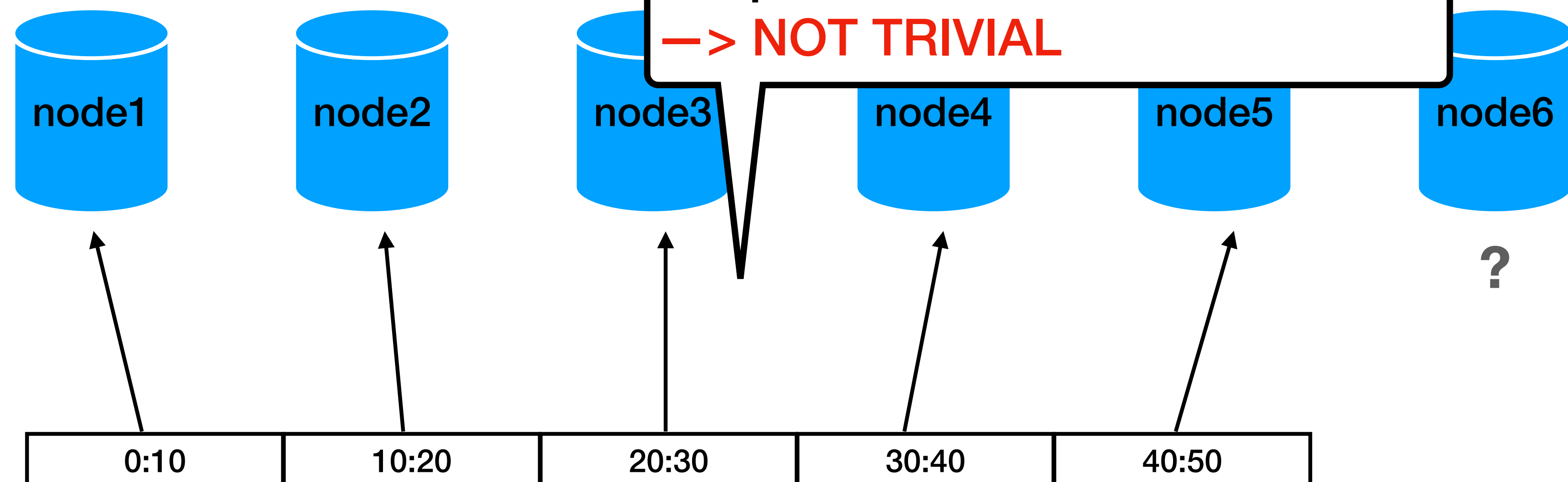
- What happens if we want to add a node?

- new data?
- existing data?

We need to:

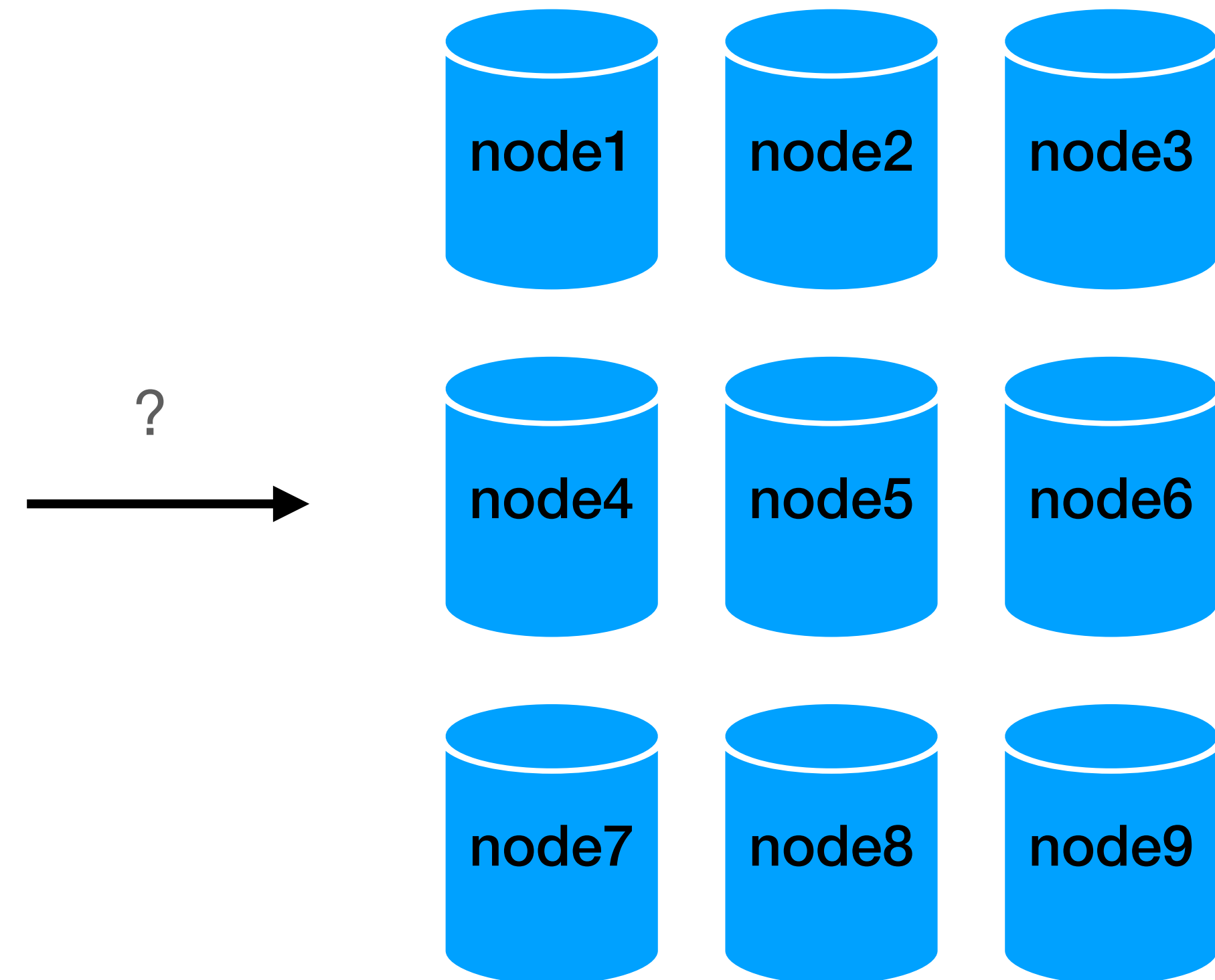
1. Change the range mapping
2. Redistribute the existing data
3. Update the DB for future data

→ NOT TRIVIAL



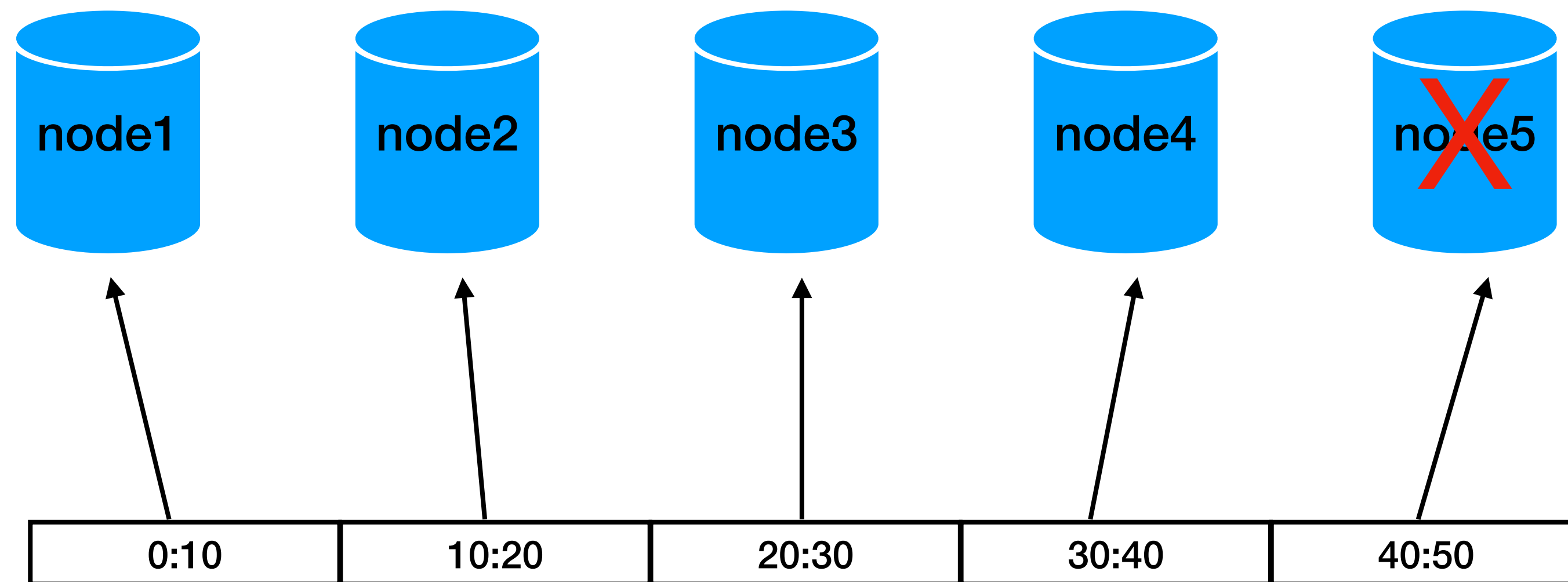
Going distributed

- **Not trivial... :)**
- Starting with:
 - Data fragmentation
 - Data distribution
 - **Data replication**



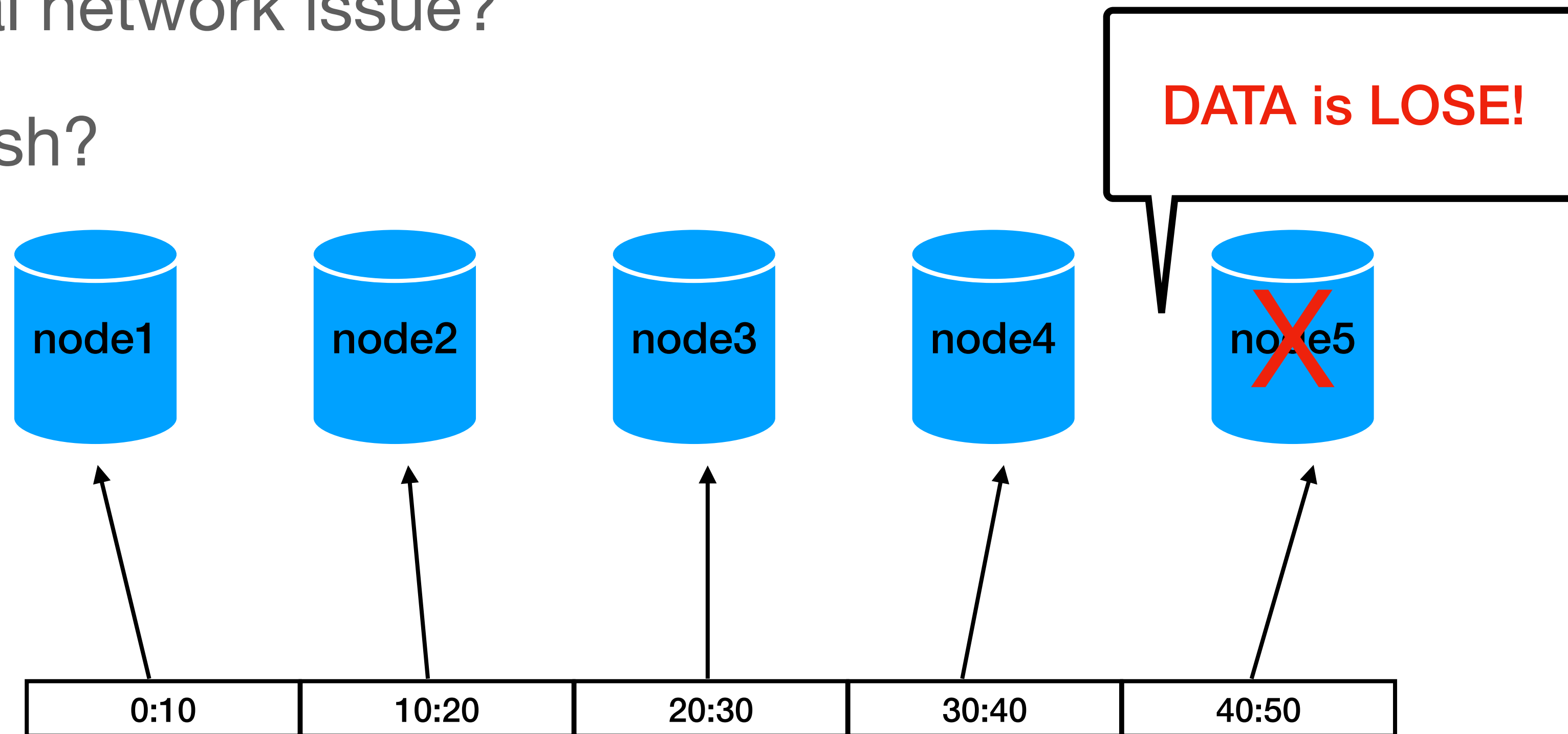
Stuff happens

- What happens if a node fails?
 - temporal network issue?
 - disk crash?



Stuff happens

- What happens if a node fails?
 - temporal network issue?
 - disk crash?



Data replication

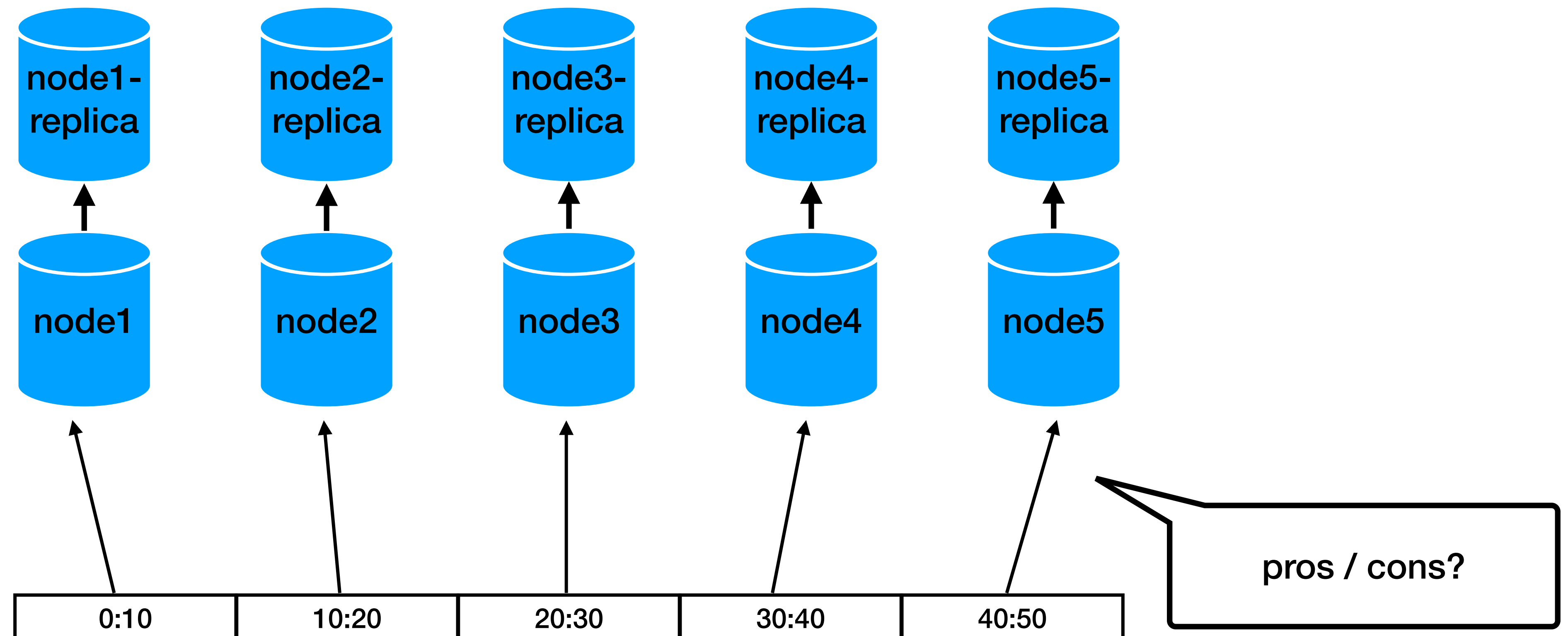
- Replicate the data on more than one node
and more than one data center

Solves 2 issues

- On errors (network / hardware) we have a backup
- On Spikes we can utilize more than 1 node

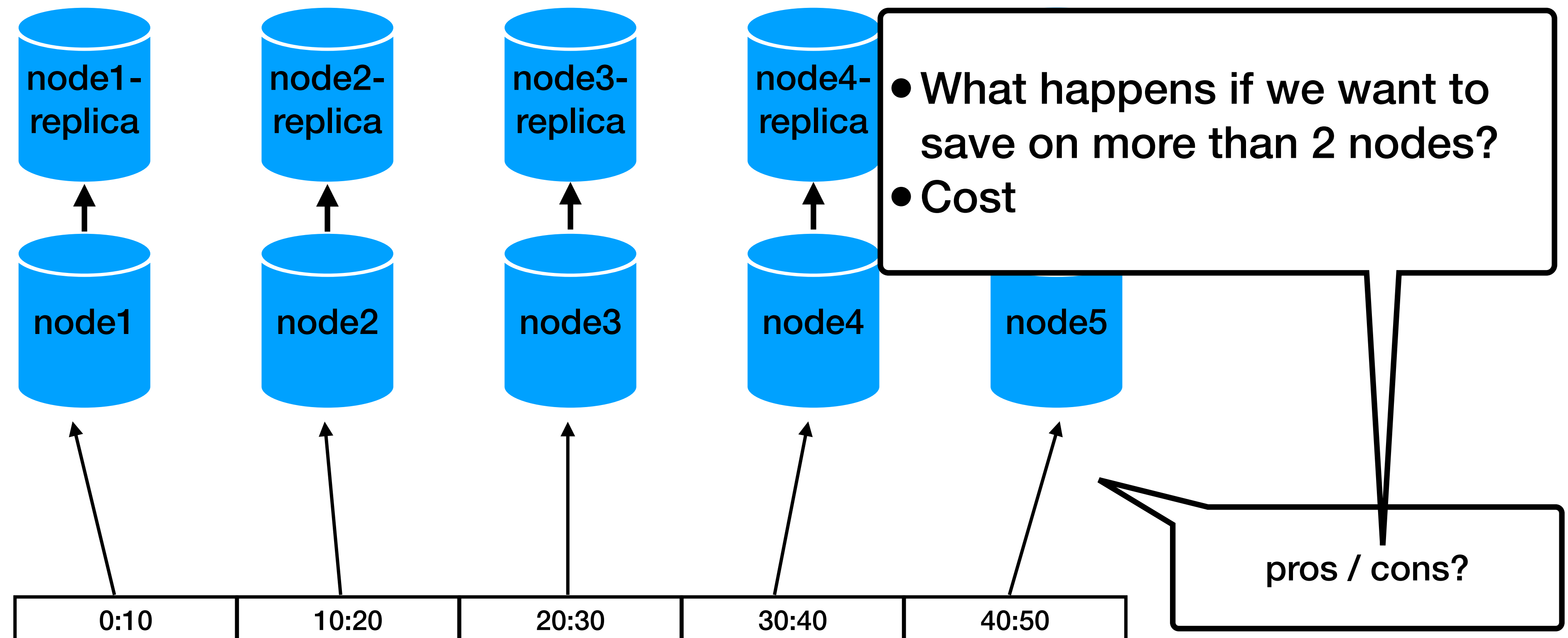
Data replication

- Read replicas + master slave?



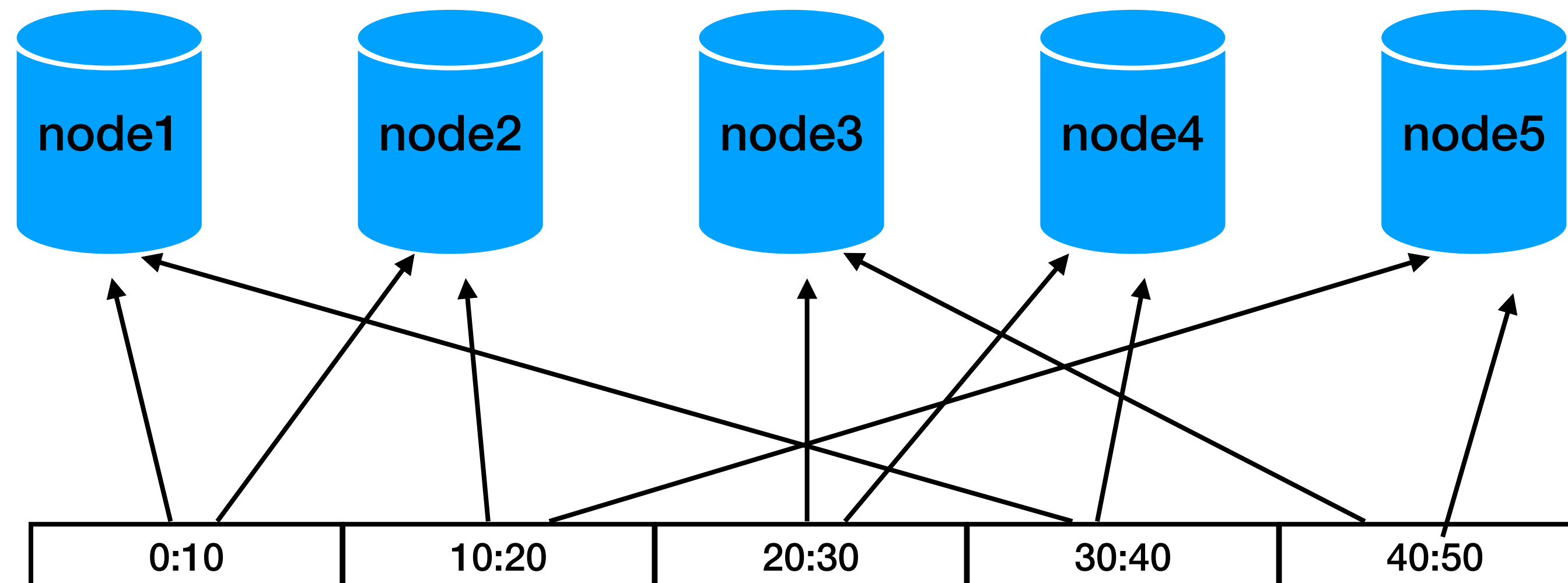
Data replication

- Read replicas + master slave?



Data replication

- (re)distribute among all nodes

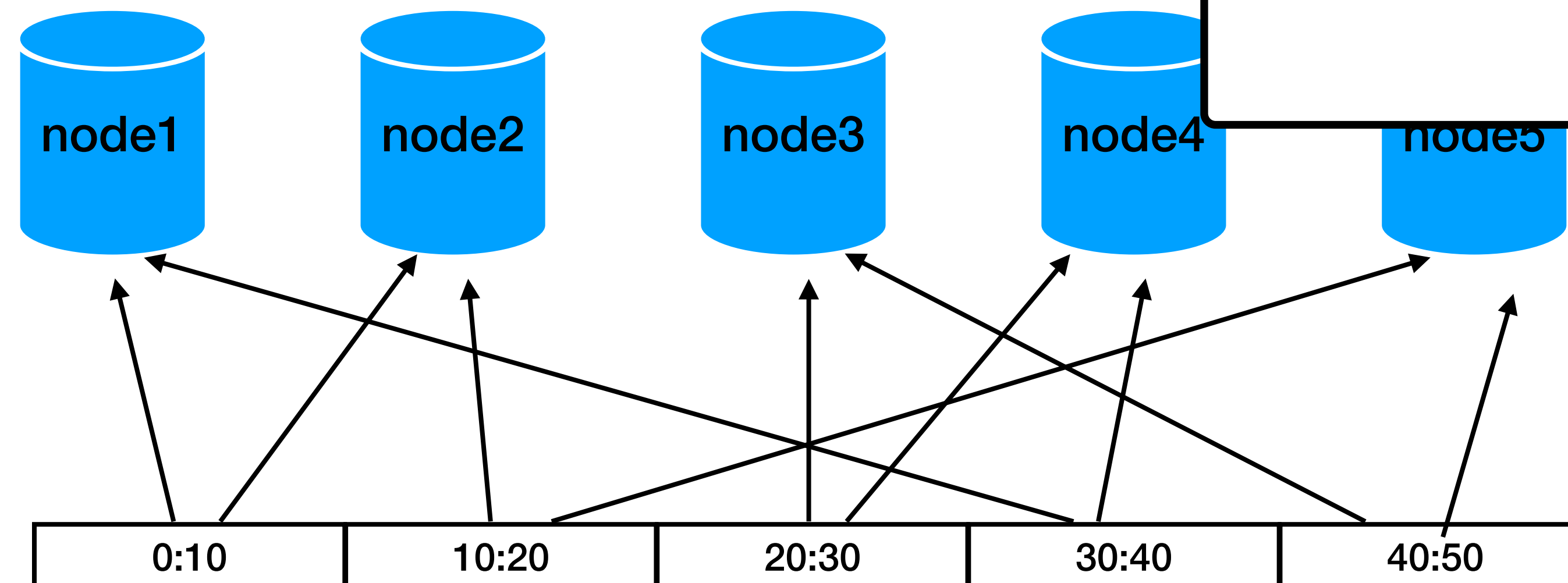


replication factor = 2

pros / cons?

Data replication

- (re)distribute among all nodes



replication factor = 2

How to manage the mapping
between range / nodes /
replicas?

pros / cons?

Spoiler alert

- Get ready for the “Dynamo” lecture

We are just getting started

- Managing the fragmentation, distribution and replication of the data is a hard problem
- We also need to support:
 - **Consistency**
Every read receives the most recent write or an error
 - **Availability**
Every request receives a (non-error) response, without the guarantee that it contains the most recent write
 - **Partition tolerance**
The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network

