# Big Data, Techniques and Platforms
# Class 2/8: Data Distribution

Francesca Bugiotti

CentraleSupélec

October 3, 2023

# Objectives

- Data distribution models
- Consistency in distributed databases
- Lesson 5: Availability in distributed databases
- Lesson 5: CAP theorem

# Plan

# Summary

## Data Science

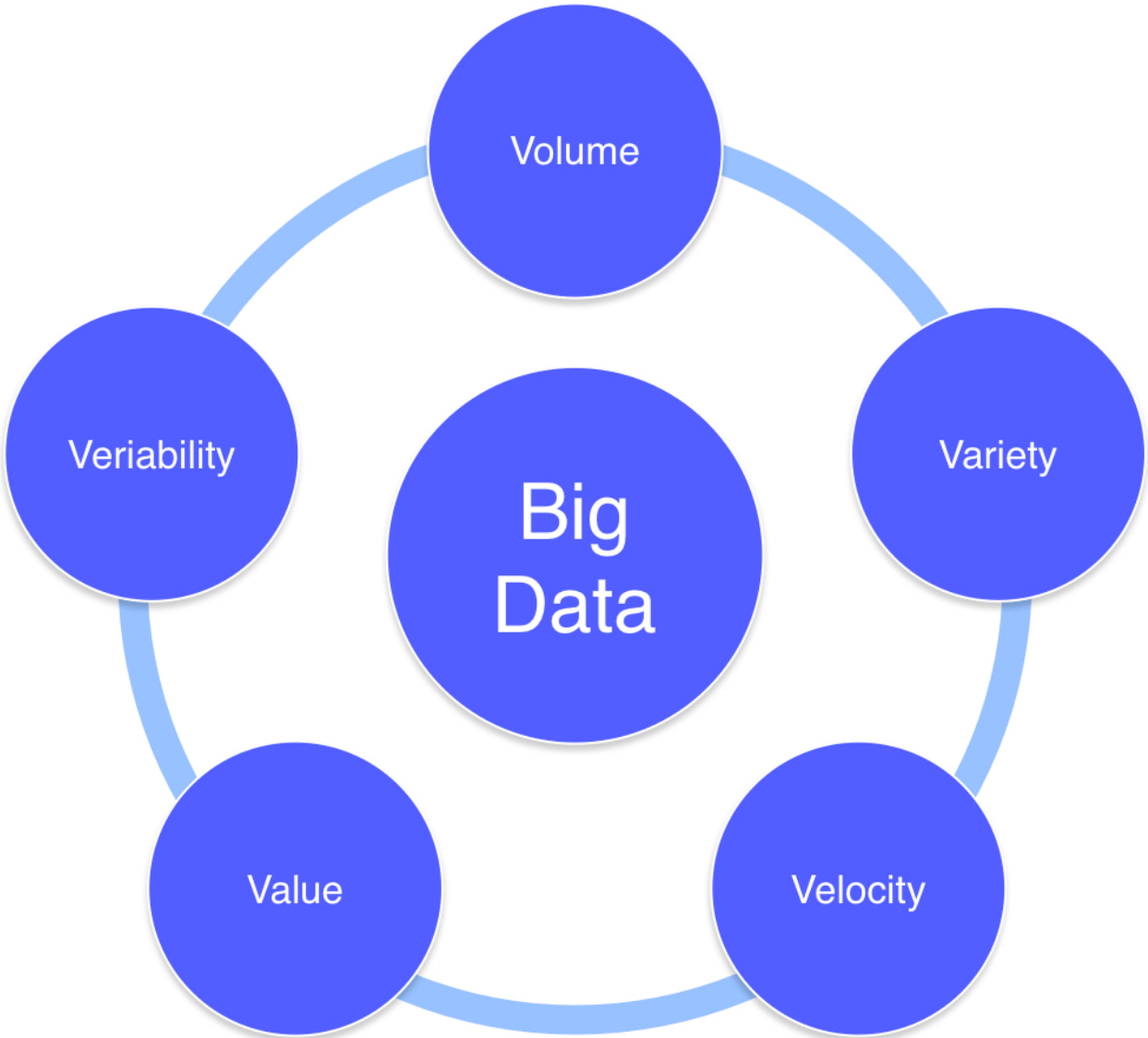
- Data
  - Common devices and specialized devices
  - New data sources producing torrent of data
- Cloud Computing
  - Computing anywhere and anytime
  - On-demand Computing

# Big data is not just "Big"

## Combining data from different sources

- Machines
  - real-time data, sensor data, trackers, streaming, etc.
- People
  - social media, personal documents, etc.
- Organizations
  - structured knowledge bases, data-warehouses, etc.

# Big Data: the 5 Vs



# MODERN DATA SCIENTIST

Data Scientist, the sexiest job of the 21th century, requires a mixture of multidisciplinary skills ranging from an intersection of mathematics, statistics, computer science, communication and business. Finding a data scientist is hard. Finding people who understand who a data scientist is, is equally hard. So here is a little cheat sheet on who the modern data scientist really is.

## MATH & STATISTICS

- Machine learning
- Statistical modeling
- Experiment design
- Bayesian inference
- Supervised learning: decision trees, random forests, logistic regression
- Unsupervised learning: clustering, dimensionality reduction
- Optimization: gradient descent and variants

## PROGRAMMING & DATABASE

- Computer science fundamentals
- Scripting language e.g. Python
- Statistical computing packages, e.g., R
- Databases: SQL and NoSQL
- Relational algebra
- Parallel databases and parallel query processing
- MapReduce concepts
- Hadoop and Hive/Pig
- Custom reducers
- Experience with xaaS like AWS

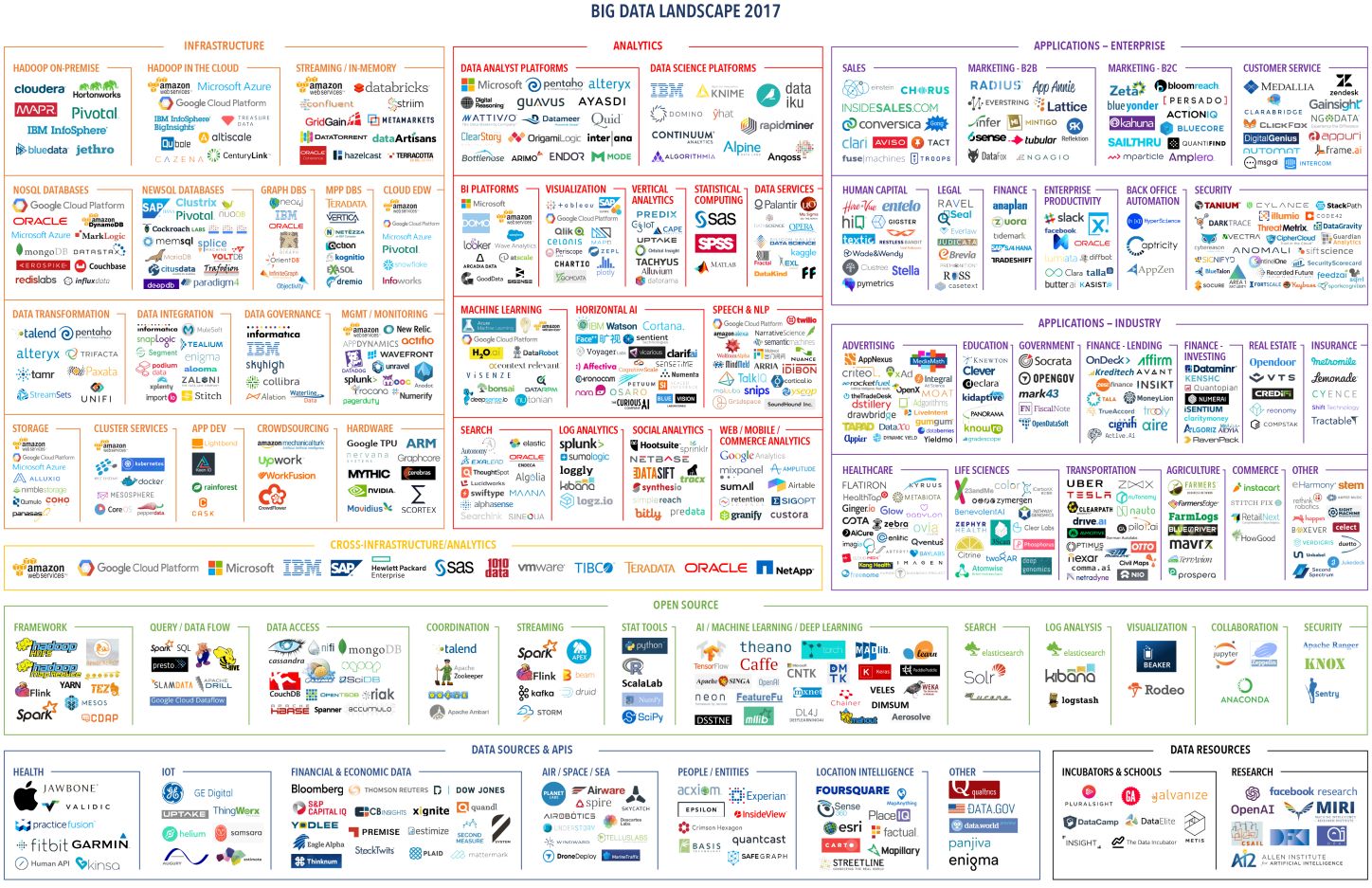
## DOMAIN KNOWLEDGE & SOFT SKILLS

- Passionate about the business
- Curious about data
- Influence without authority
- Hacker mindset
- Problem solver
- Strategic, proactive, creative, innovative and collaborative

## COMMUNICATION & VISUALIZATION

- Able to engage with senior management
- Story telling skills
- Translate data-driven insights into decisions and actions
- Visual art design
- R packages like ggplot or lattice
- Knowledge of any of visualization tools e.g. Flare, D3.js, Tableau

# Technologies




V2 – Last updated 5/3/2017

 mattturck.com/bigdata2017

FIRSTMARK EARLY STAGE VENTURE CAPITAL

# Bakery

# Baker - Chef

# Baker - Chef


# Ingredients

# Stockage Utils

# Utils


# Recipes


CIAMBELLONE SALATO
EMPANADA AL FORNO
(con ripieno di carne)
PREPARAZIONE
PREPARAZIONE


Galettes à la noix de coco
La pâte :
- 250 gr de beurre ramolli
- 1 verre de sucre glace
- 3 œufs
- 1 paquet de levure chimique
- 1 C. à café de vanille
- 1 verre de maïzena
- Farine (SIM)
La décoration :
- Fruits confits
- Noix de coco colorée
- Confiture
1- Préparer les ingrédients nécessaires.
2- Dans un récipient, mélanger le beurre, le sucre, la vanille, la maïzena et les œufs, ajouter la levure et progressivement la farine jusqu'à obtention d'une pâte malléable.
3- Façonner des boules de pâte puis tremper l'emporte-pièce dans la farine.
4- Bien appuyer sur chaque boule pour obtenir la forme de rose. Décorer le centre avec un morceau de fruit confit puis enfourner.
5- Après cuisson, coller les gâteaux deux à deux avec de la confiture.
6- Badigeonner les bordures de confiture.
7- Enrober-les de noix de coco colorée.
plaisir au partage


gâteau invisible
POMME CHOCO-CARAMEL

# Procedure


ALDEGHERI
ELECTRIC OVEN


gâteau invisible
POMME CHOCO-CARAMEL

# Procedure


Galettes à la noix de coco
La pâte :
- 250 gr de beurre ramolli
- 1 verre de sucre glace
- 3 œufs
- 1 paquet de levure chimique
- 1 C. à café de vanille
- 1 verre de maïzena
- Farine (SIM)
La décoration :
- Fruits confits
- Noix de coco colorée
- Confiture
1- Préparer les ingrédients nécessaires.
2- Dans un récipient, mélanger le beurre, le sucre, la vanille, la maïzena et les œufs, ajouter la levure et progressivement la farine jusqu'à obtention d'une pâte malléable.
3- Façonner des boules de pâte puis tremper l'emporte-pièce dans la farine.
4- Bien appuyer sur chaque boule pour obtenir la forme de rose. Décorer le centre avec un morceau de fruit confit puis enfourner.
5- Après cuisson, coller les gâteaux deux à deux avec de la confiture.
6- Badigeonner les bordures de confiture.
7- Enrober-les de noix de coco colorée.

# Procedure


# In our context

- Data Science
- Data
- Data Scientist
- Algorithms: K-means, PageRank, etc.
- Databases
- Map-Reduce
- Hadoop, Spark
- Produce Value

# Plan

# Scalability

Fix the terminology and the scenarios:

- Scale up
- Scale out

# Single Server

# Single Server

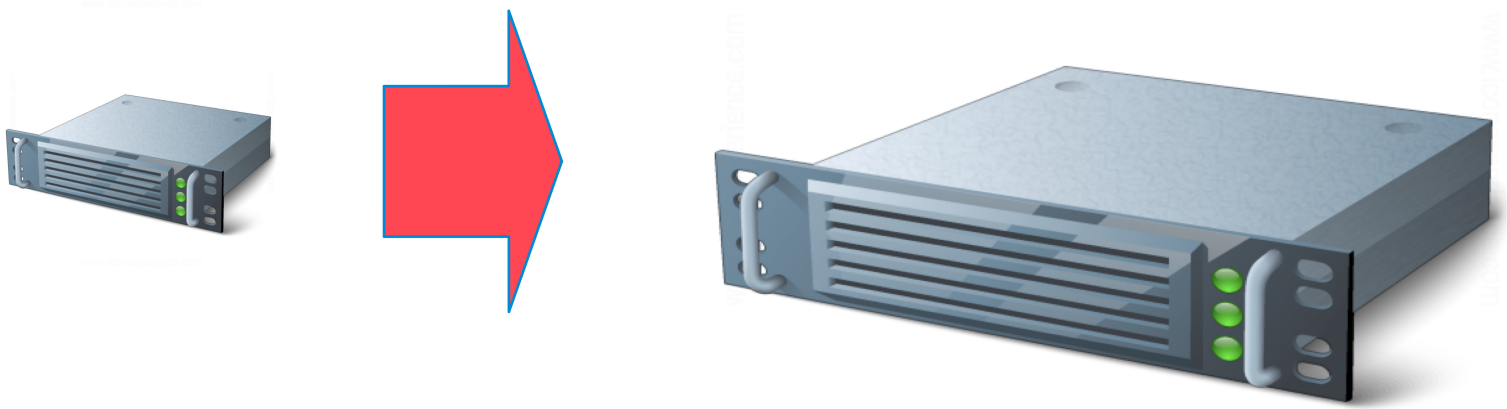
How is data organized?

## No distribution

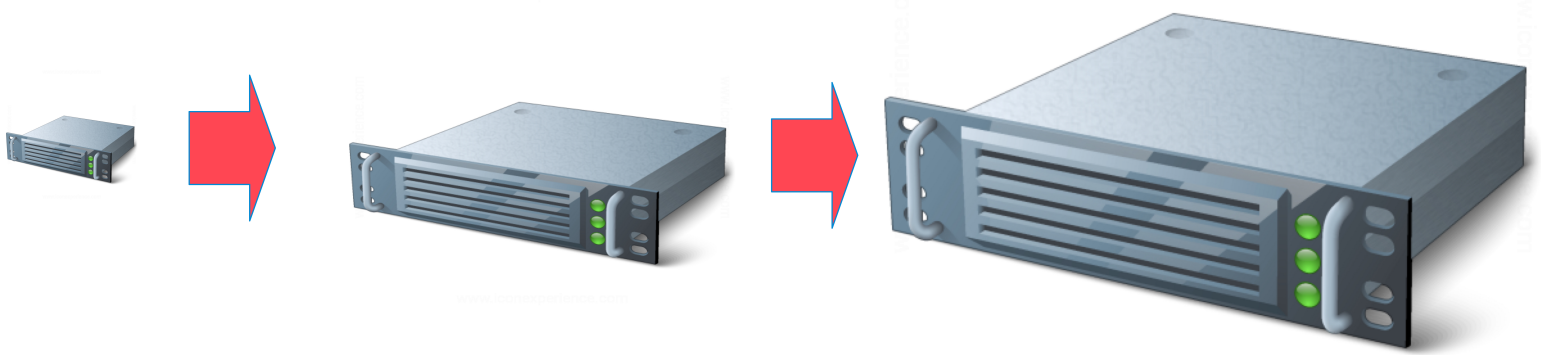
Data is on a single machine

# Scale up

# Scale up



# Scale up



# Single Server - Scalability

How do we achieve scalability?

## Scenario - Single Server

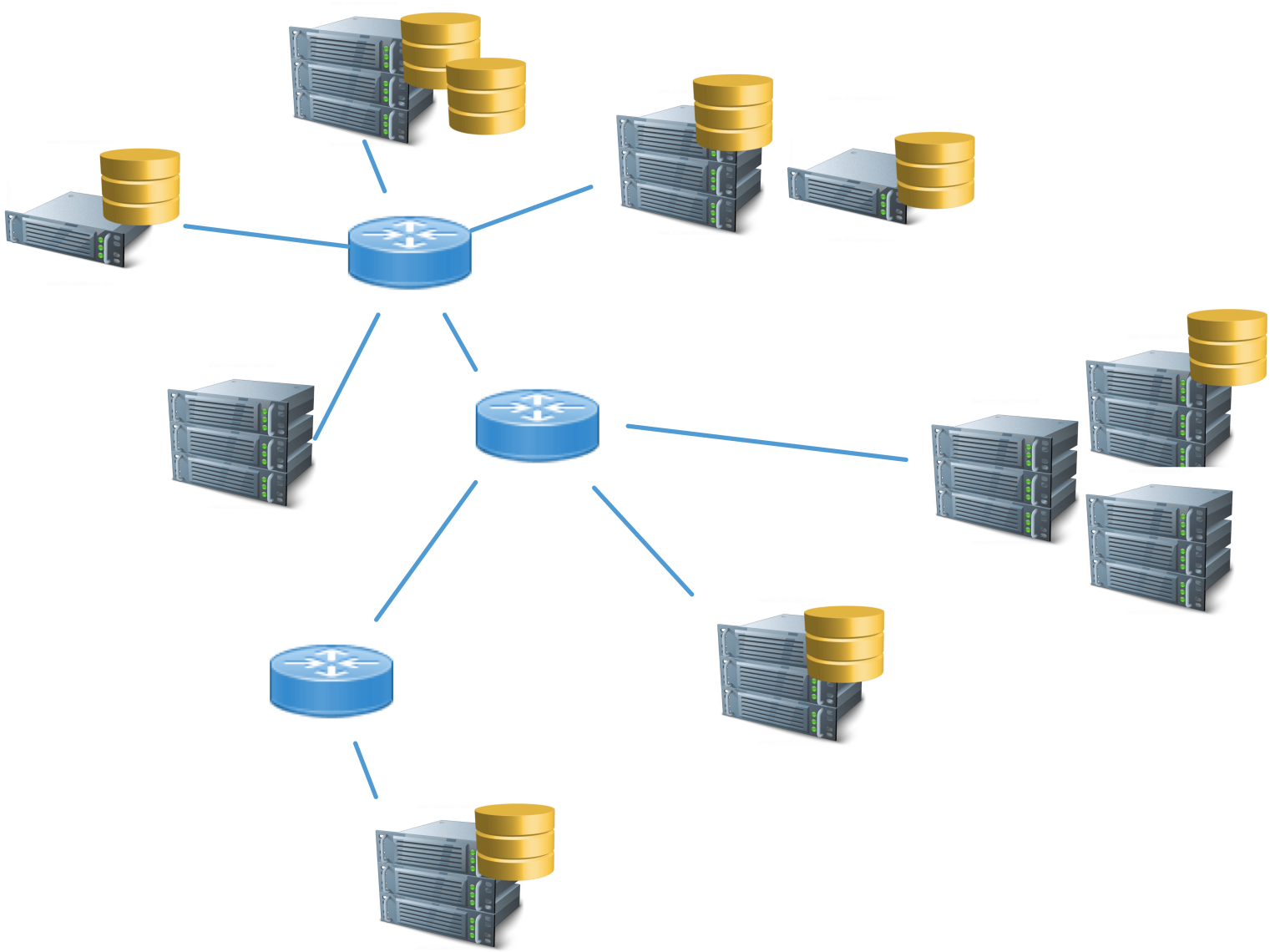
Methodology - Scale up:

- Add new computation capability if needed and/or if possible

The single machine handles all the reads and writes to data

- Hint for the future: fits well with graph databases

# Multiple servers



# Multiple servers

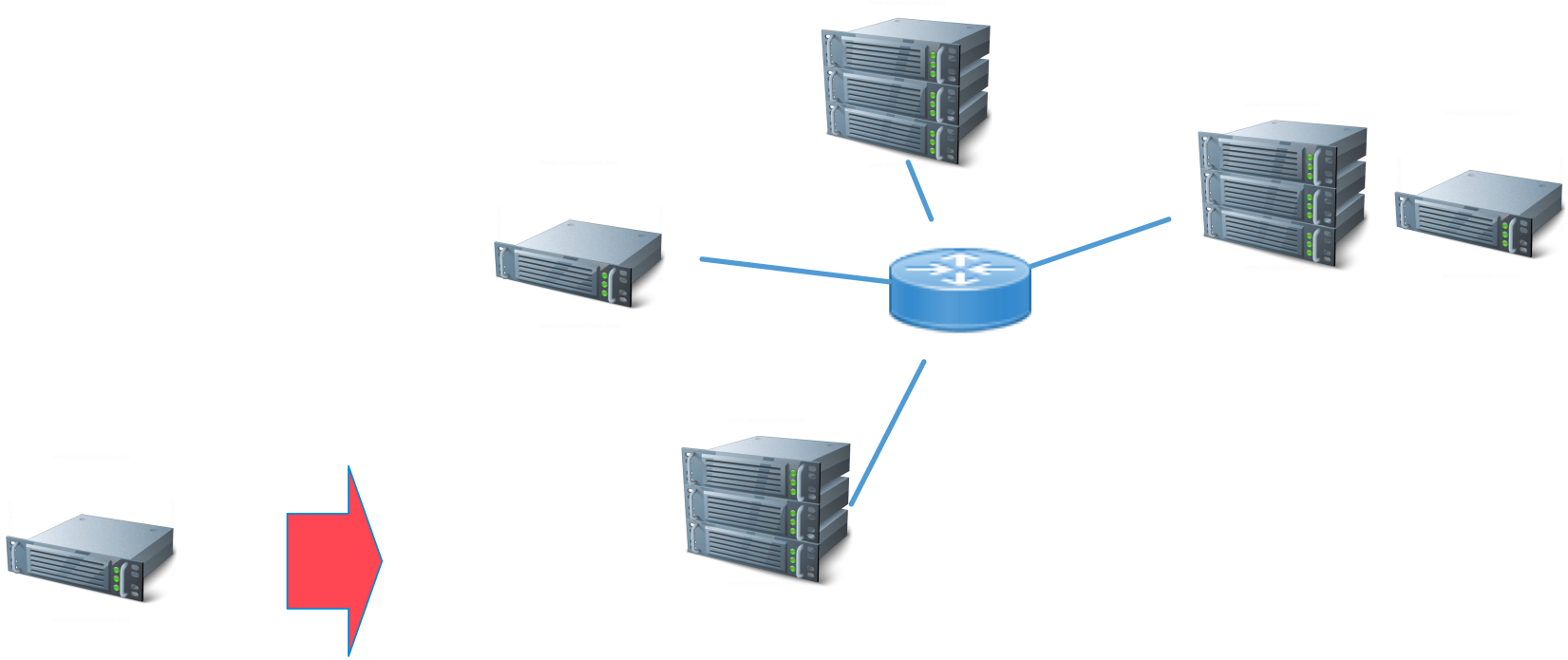
How is data organized?

## Data is distributed

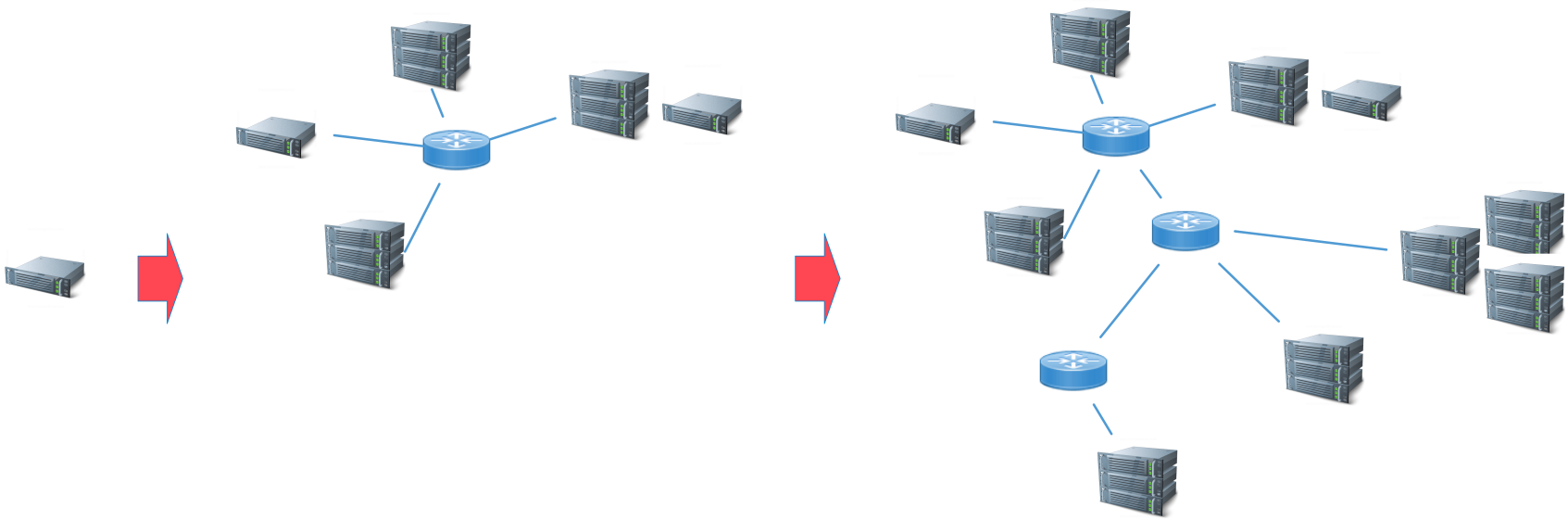
Data is on multiple nodes of a cluster

# Scale Out

# Scale Out



# Scale Out



# Multiple Servers - Scalability

How do we achieve scalability?

## Scenario - Multiple Servers

Methodology - Scale out:

- Run the database on a cluster of servers
- Add new nodes when needed

## Related issues

1. How to distribute (shard) data across the nodes
2. How and if replicate data
3. How to assign read/write operations
    - Focus on read/write operations

# Plan

# Data Distribution

- Data distribution models:
  - Single server
  - Multiple servers
- Orthogonal aspects of data distribution:
  - Sharding: different data on different nodes
  - Replication: the same data copied over multiple nodes

# Sharding

Different parts of data into different nodes and each node does its own reads and writes

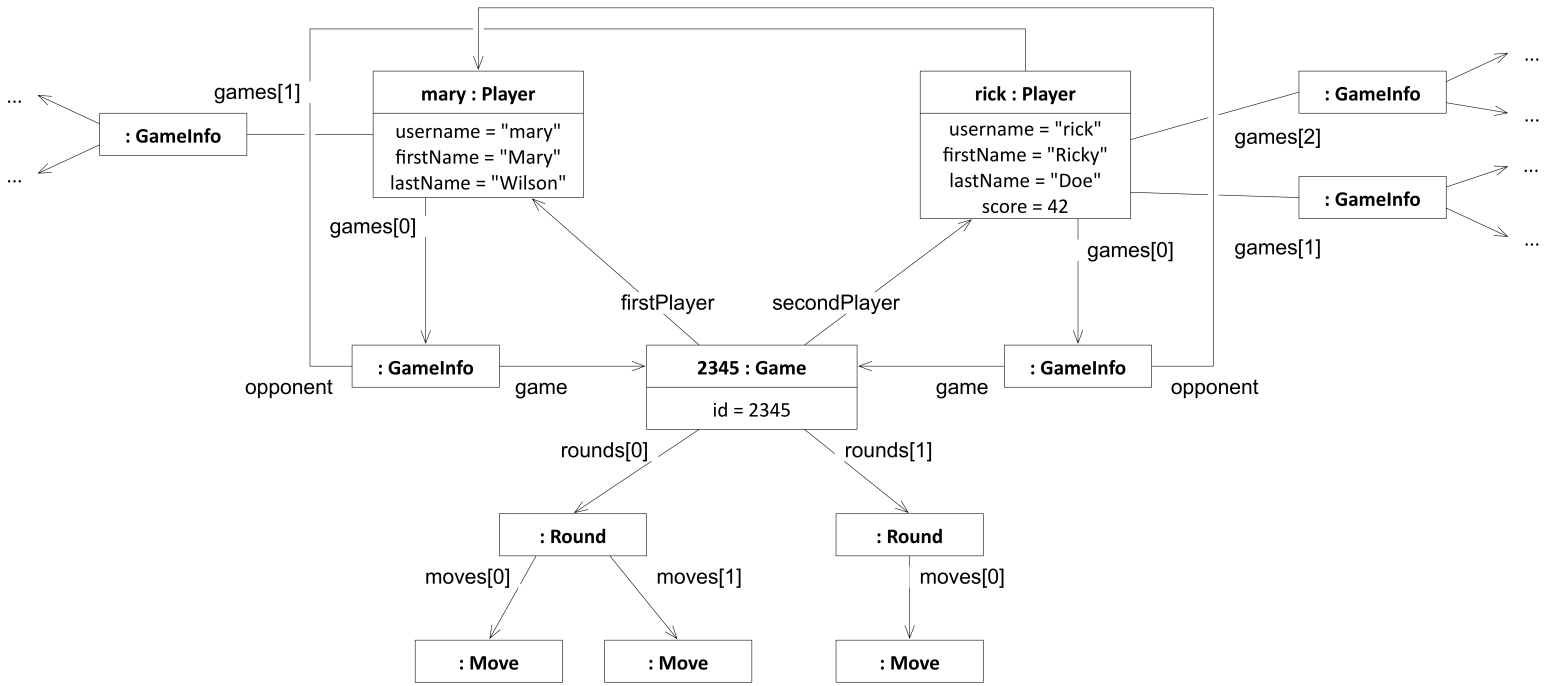
- Horizontal scalability
- Ideal case: different users all talking to different server nodes
- Data accessed together on the same node
  - aggregate data!
  - How?
  - This is a challenge!!!

- Pros: it can improve both reads and writes
- Cons: clusters use less reliable machines - resilience decreases

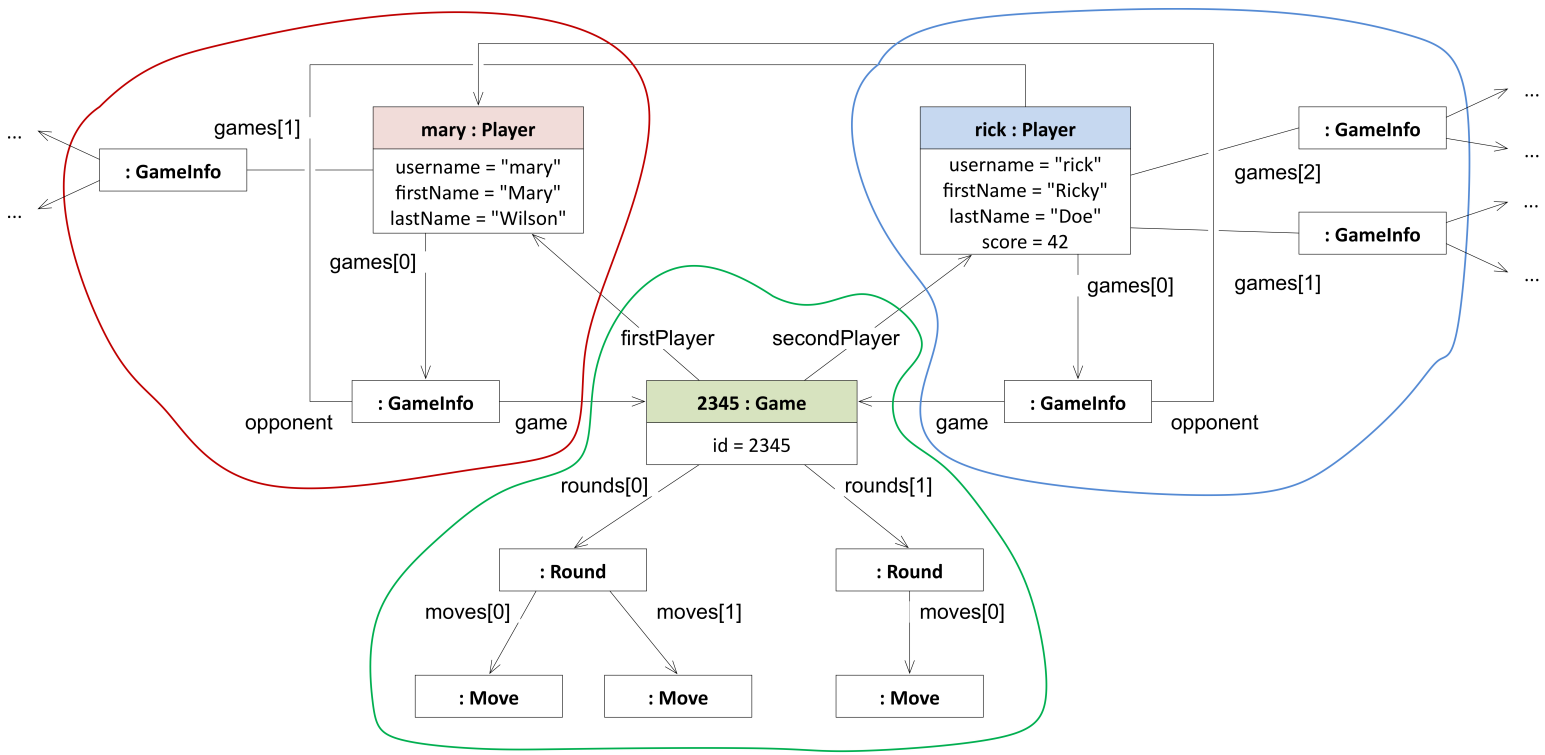
# Aggregates

- Data as units that have a complex structure
  - More structure than just a set of tuples
    - A complex record with fields inside
    - A record with a composite access structure
- Aggregates in Domain Driven Design [2]:
  - A collection of related objects that we treat as a unit
  - A unit for data manipulation and management of consistency
- Advantages of aggregates:
  - easier for programmer to work with
  - easier for database systems to handle operations on a cluster

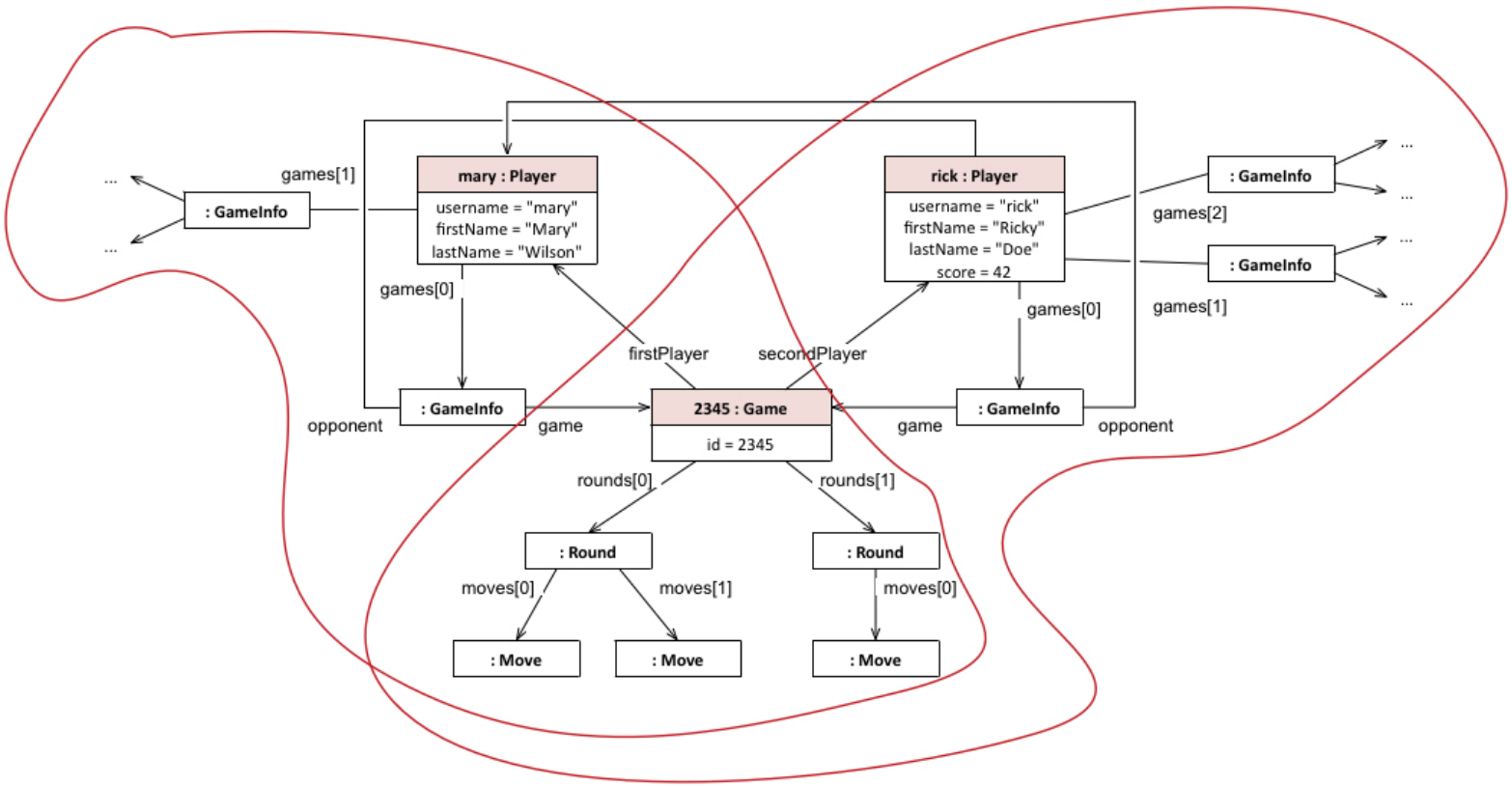
# Aggregates [1]



# Aggregates


mary : Player
username = "mary"
firstName = "Mary"
lastName = "Wilson"
rick : Player
username = "rick"
firstName = "Ricky"
lastName = "Doe"
score = 42
2345 : Game
id = 2345
: GameInfo
: GameInfo
: GameInfo
: GameInfo
: GameInfo
: Round
: Round
: Move
: Move
: Move
games[1]
games[0]
games[0]
games[2]
games[1]
firstPlayer
secondPlayer
opponent
game
game
opponent
rounds[0]
rounds[1]
moves[0]
moves[1]
moves[0]

# Aggregates


mary : Player
username = "mary"
firstName = "Mary"
lastName = "Wilson"
rick : Player
username = "rick"
firstName = "Ricky"
lastName = "Doe"
score = 42
2345 : Game
id = 2345
: GameInfo
: GameInfo
: GameInfo
: GameInfo
: GameInfo
: Round
: Round
: Move
: Move
: Move
games[1]
games[0]
games[0]
games[2]
games[1]
firstPlayer
secondPlayer
opponent
game
game
opponent
rounds[0]
rounds[1]
moves[0]
moves[1]
moves[0]

# Design Strategy

- No universal strategy for designing aggregates boundaries
- The design depends on data manipulation
  - Access on a single round
  - Access on a player with all his games and rounds
- Context specific:
  - Some applications will run on a data representation
  - Some others on another data representation
- Focus on the Data Access Unit
  - very good for clusters
  - not easy to change the data interaction strategy

# Sharding

Main rules of sharding:

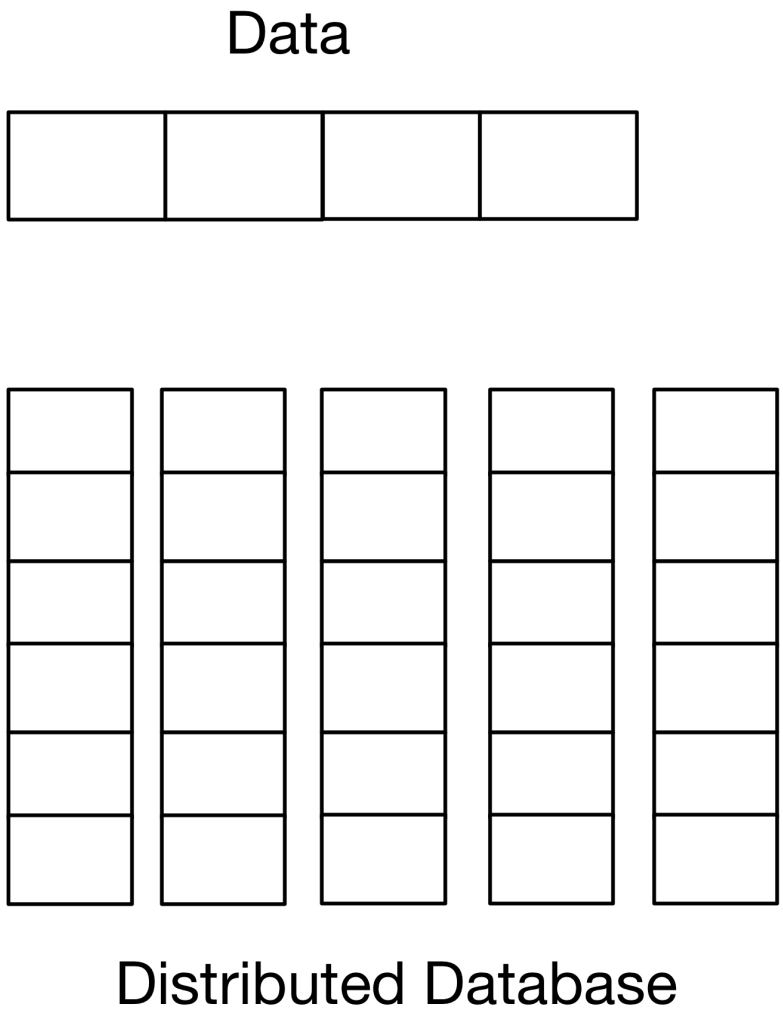
1. Place data close to where it's accessed
   - Games played by Californian players: data in the western US data center
2. Try to keep the load even
   - All nodes should get equal amounts of the load
3. Put together aggregates that may be read in sequence
   - Rounds of the same game in the same node

# Horizontal scalability

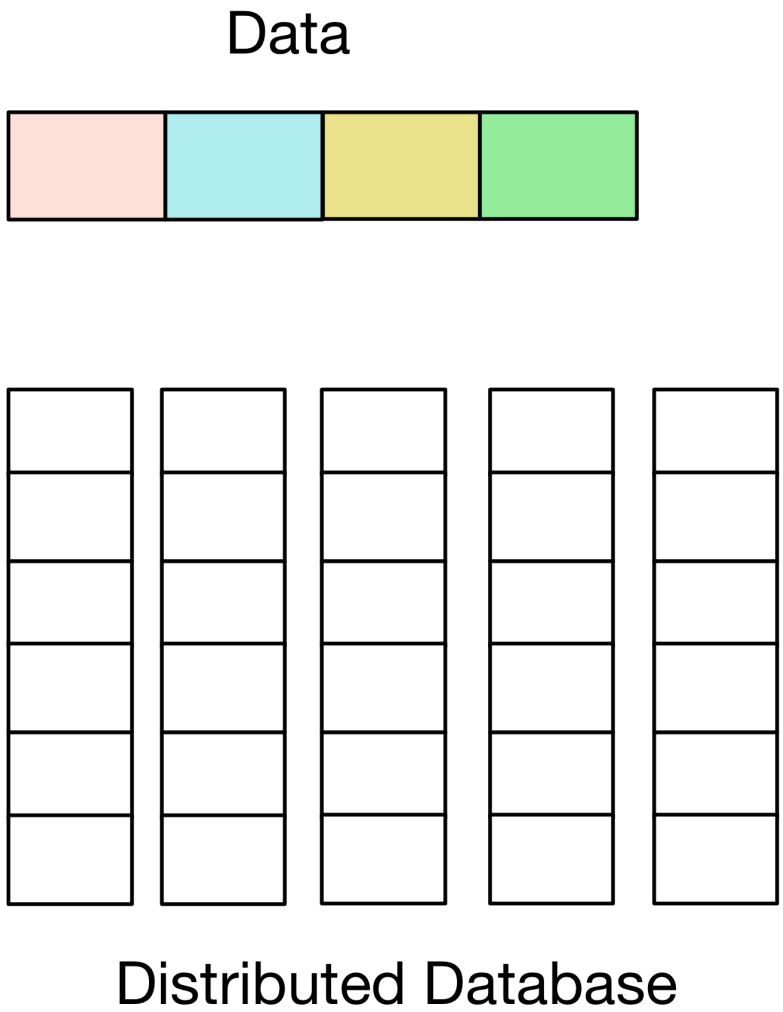
Build your database on a cluster and distribute data across the cluster

How?

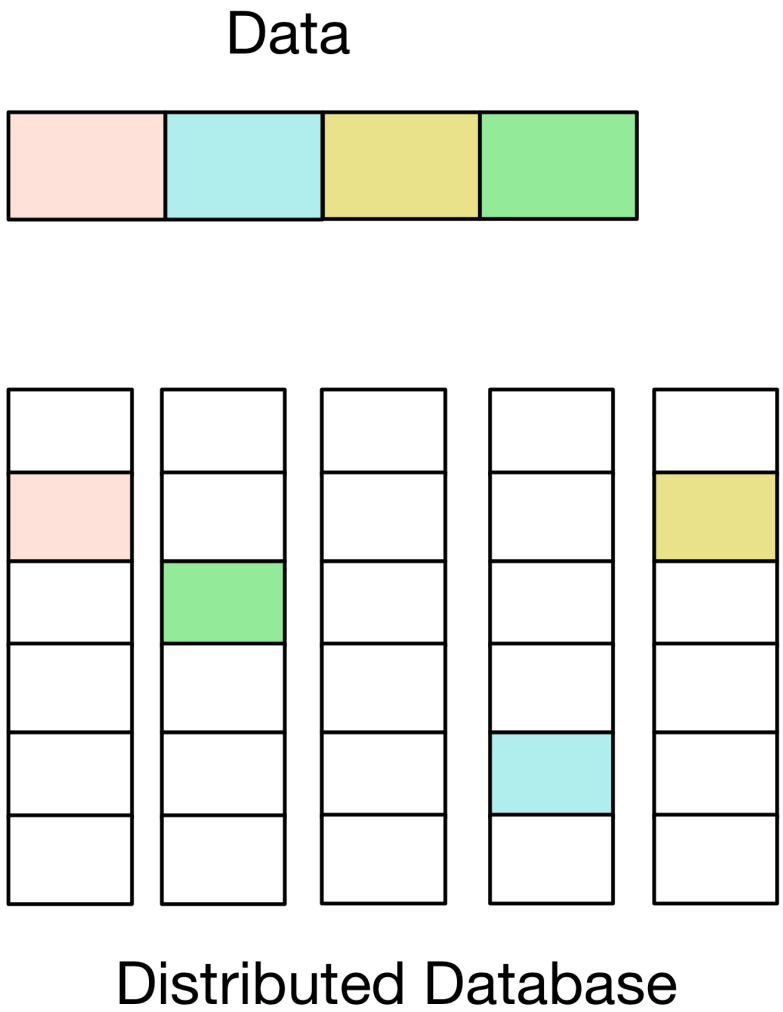
# Horizontal scalability



# Horizontal scalability



# Horizontal scalability

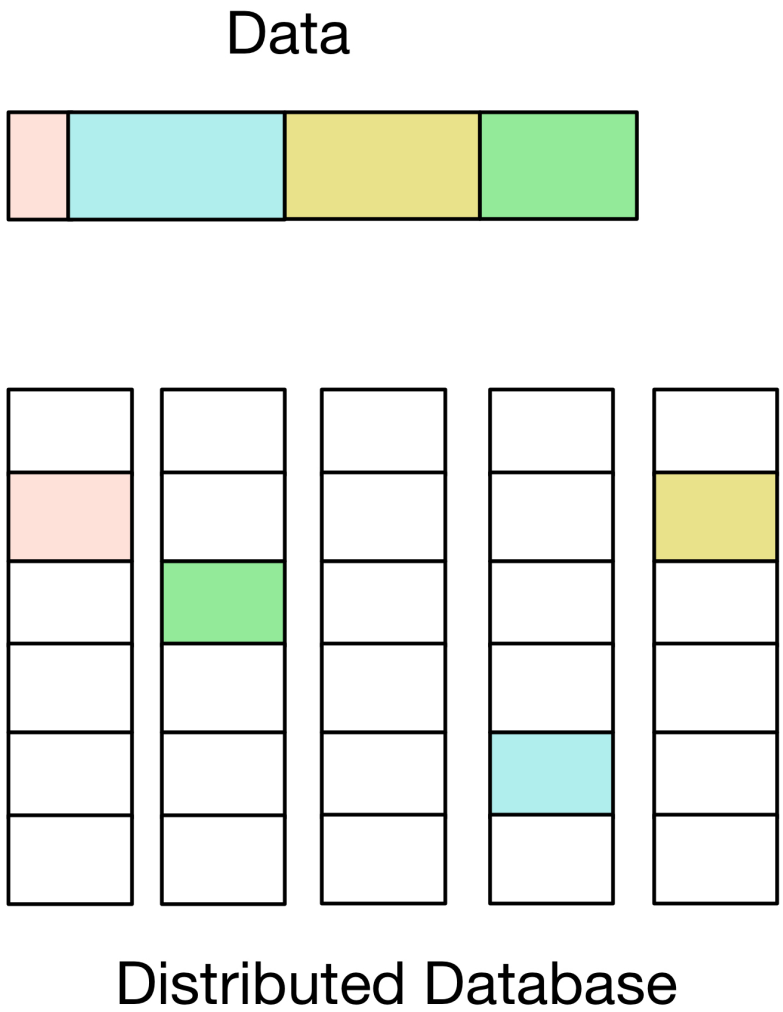


# Horizontal scalability

- How to divide data:
  - think about the aggregates
  - think about how to store aggregates together

Sometimes it is necessary to rethink aggregates

# Horizontal scalability



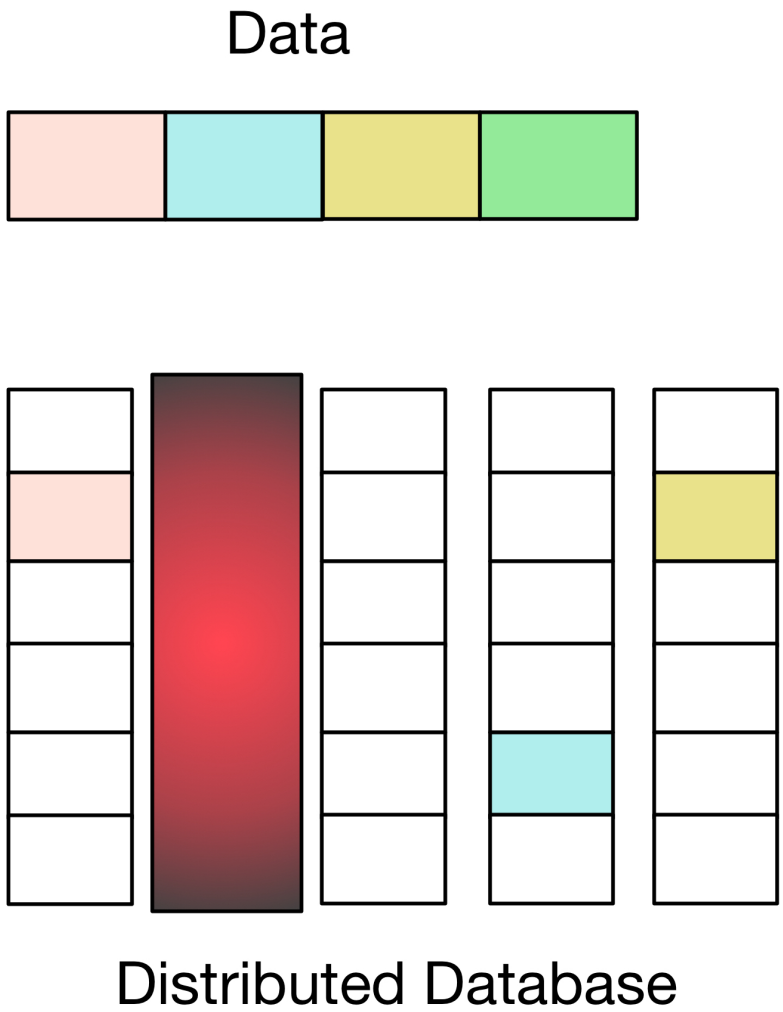
# More on Distribution

- Sharding is particularly valuable for performance
- Sharding does little for applications that have a lot of writes

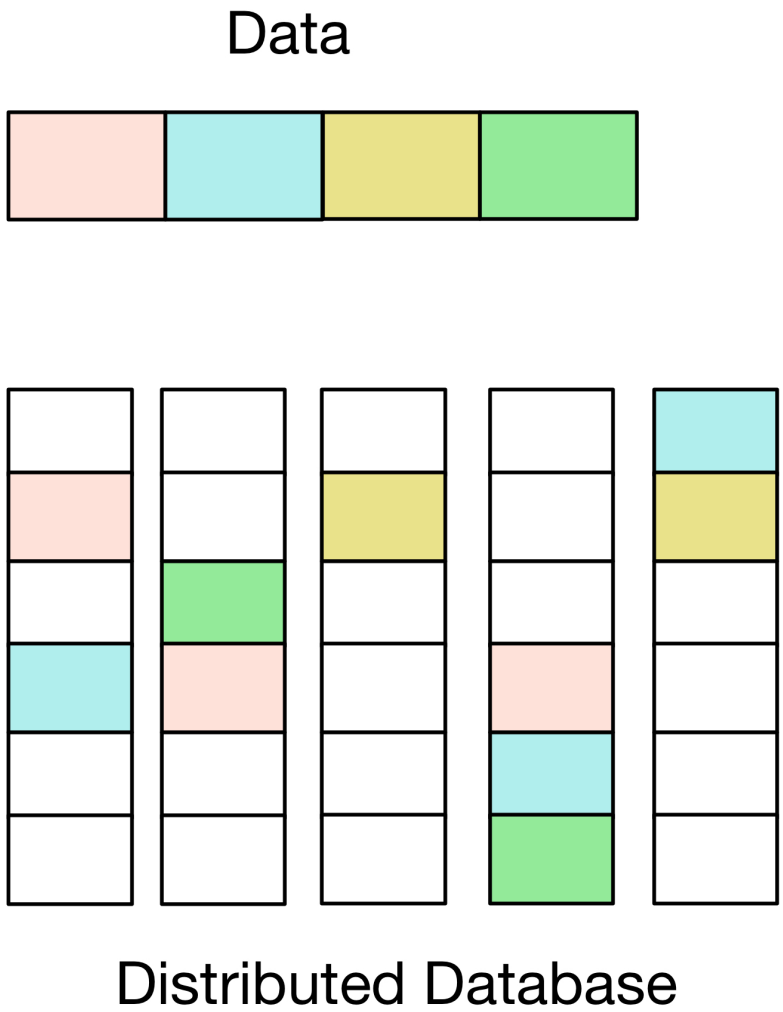
# Availability

- A node failure makes pieces of data unavailable
  - Maybe not for all the users
- It is like having a single server
- Moreover clusters usually try to use less reliable machines

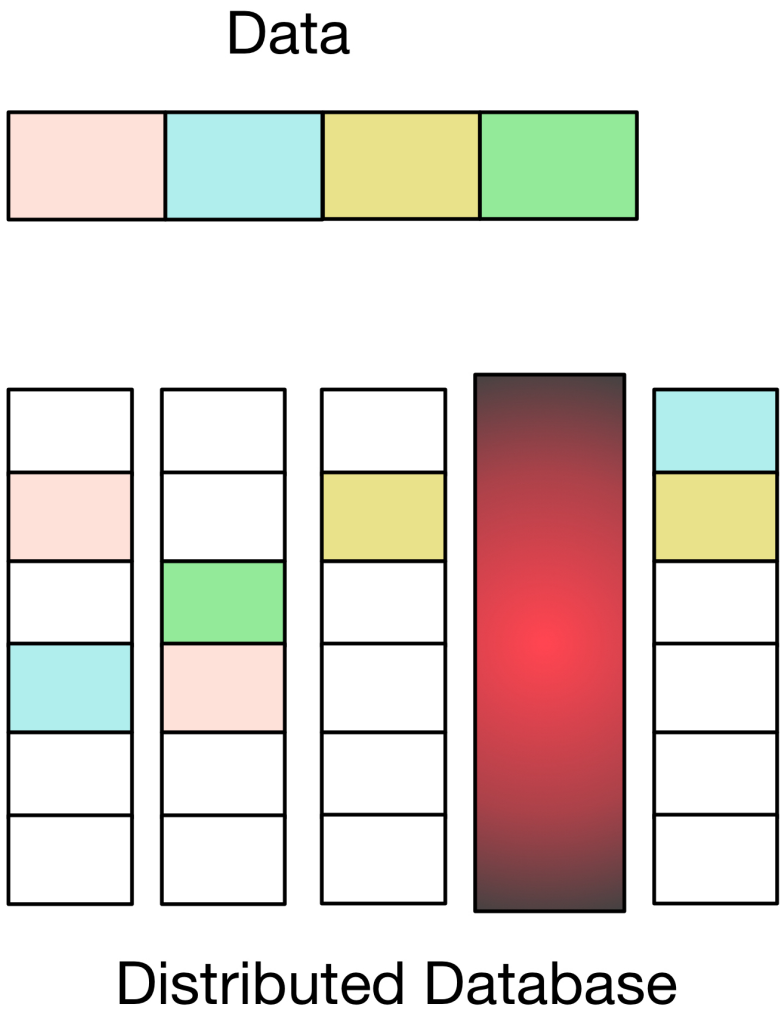
# Availability



# Availability - Data Replication



# Availability - Data Replication



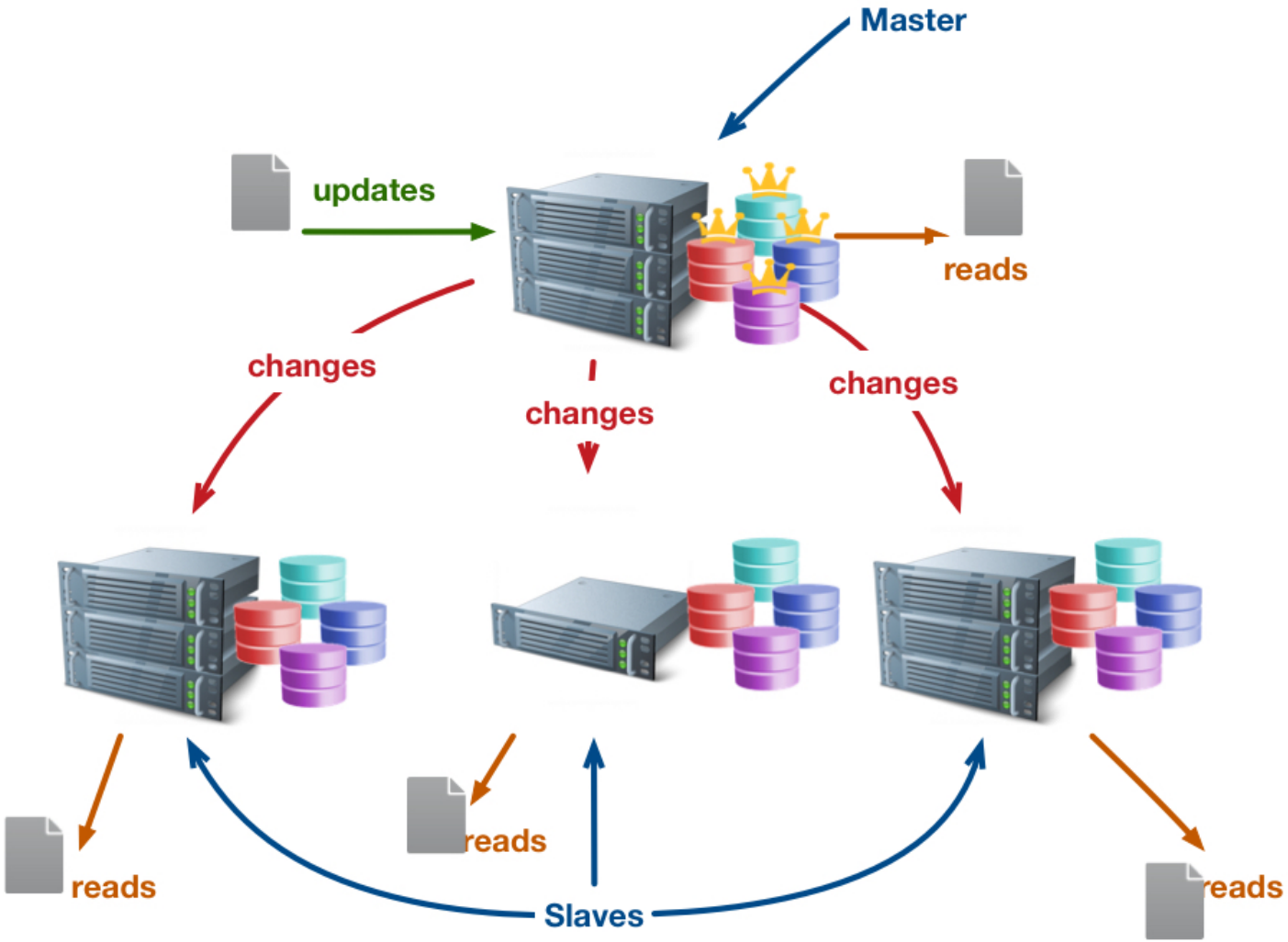
# Data Replication

Two forms of data replication:

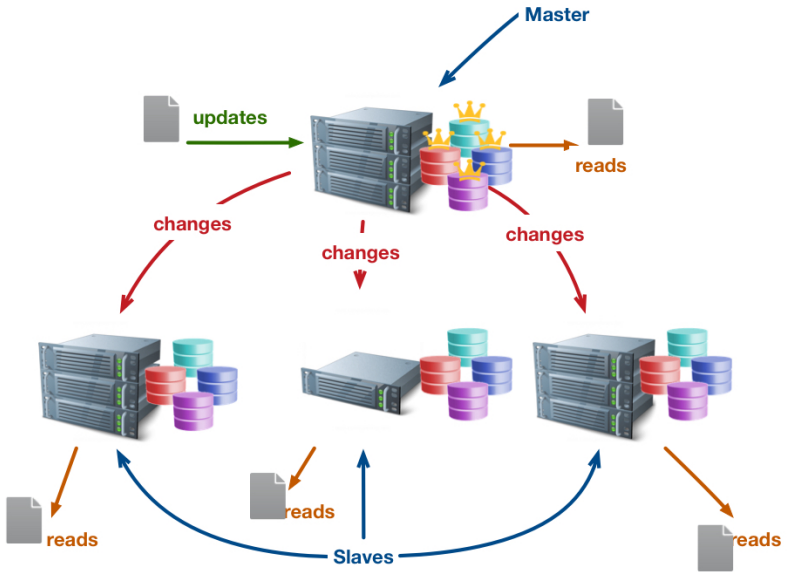
- Master-slave
- Peer to peer

# Master-Slave Replication

# Master-Slave Replication



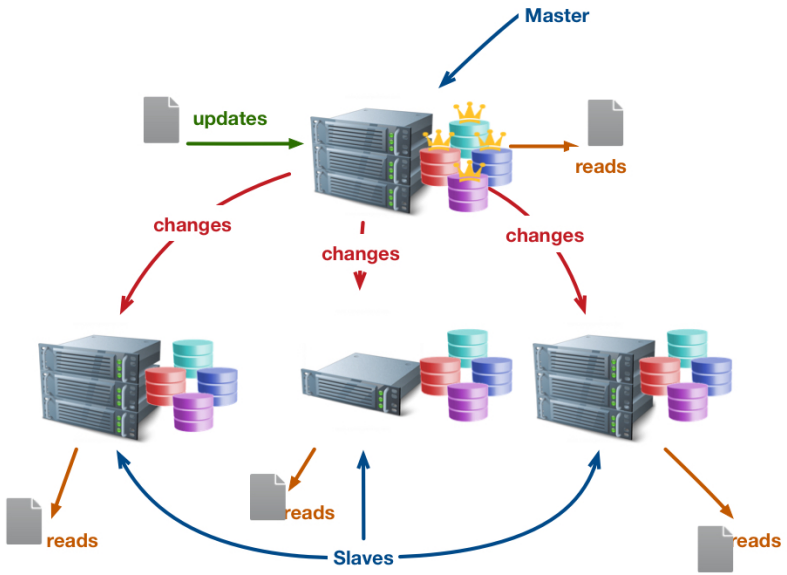
# Master-Slave Replication




## Master:

- is the authoritative source for the data
- is responsible for processing any data update
- can be appointed manually or automatically

# Master-Slave Replication




## Slaves:

- A replication process synchronizes the slaves with the master
- After a failure of the master, a slave can be appointed as new master very quickly

# Master-slave analysis

## Pros

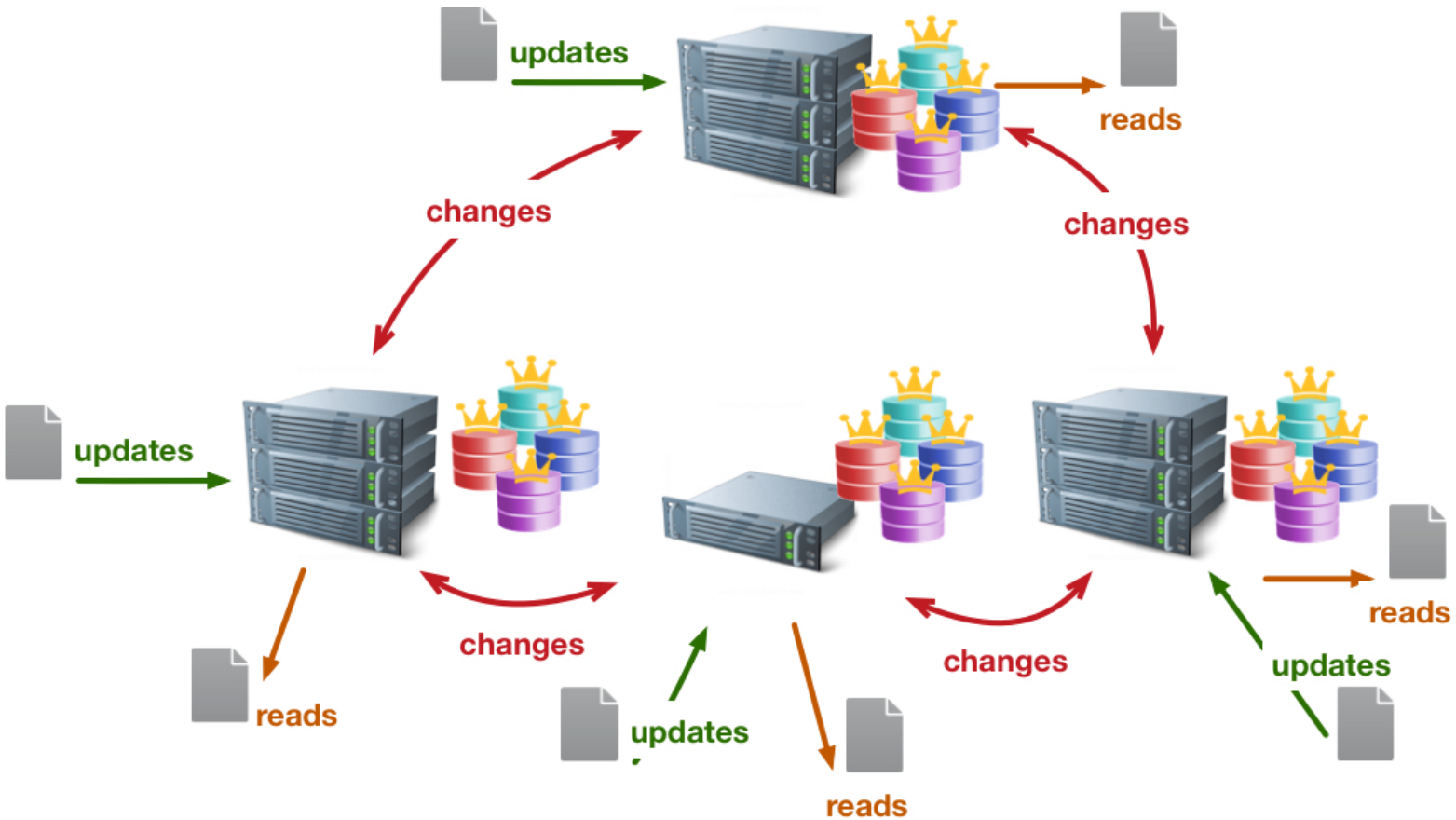
- More read requests:
  - Add more slave nodes
  - Ensure that all read requests are routed to the slaves
- Should the master fail, the slaves can still handle read requests
- Good for datasets with a read-intensive dataset

## Cons

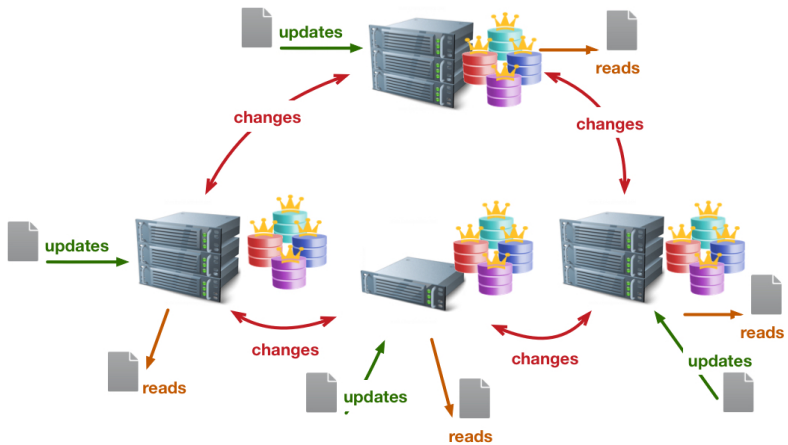
- The master is a bottleneck
  - Limited by its ability to process updates and to pass those updates on
  - Its failure does eliminate the ability to handle writes until:
    - The master is restored or
    - A new master is appointed
- Inconsistency due to slow propagation of changes to the slaves
- Bad for datasets with heavy write traffic

# Peer-to-Peer Replication

# Peer-to-Peer Replication



# Peer-to-Peer Replication




- All the replicas have equal weight: they can all accept writes
- The loss of any of them doesn't prevent access to the data store

# Peer-to-Peer analysis

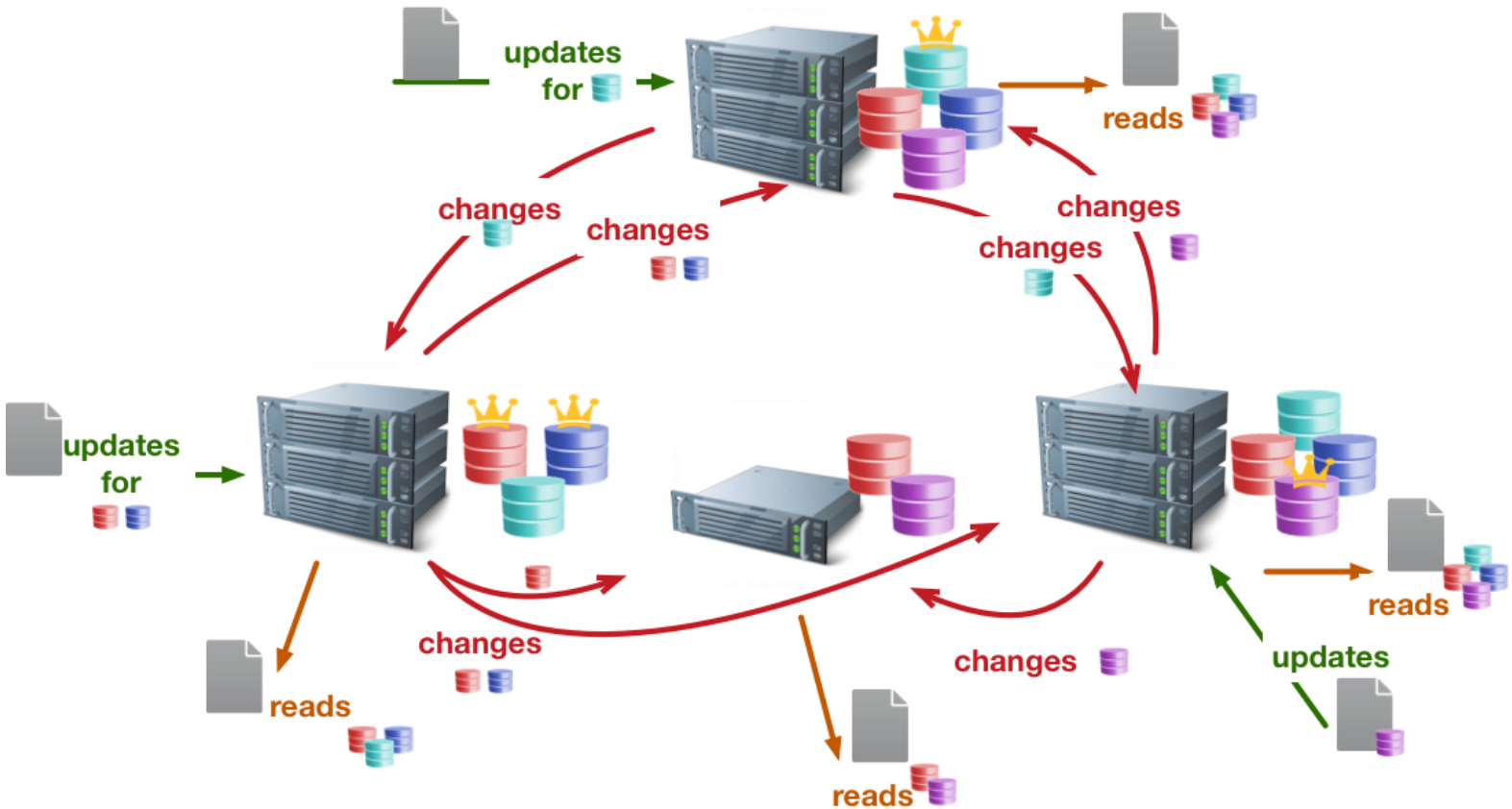
## Pros

- Ride over node failures without losing access to data
- Easy to add nodes to improve performance

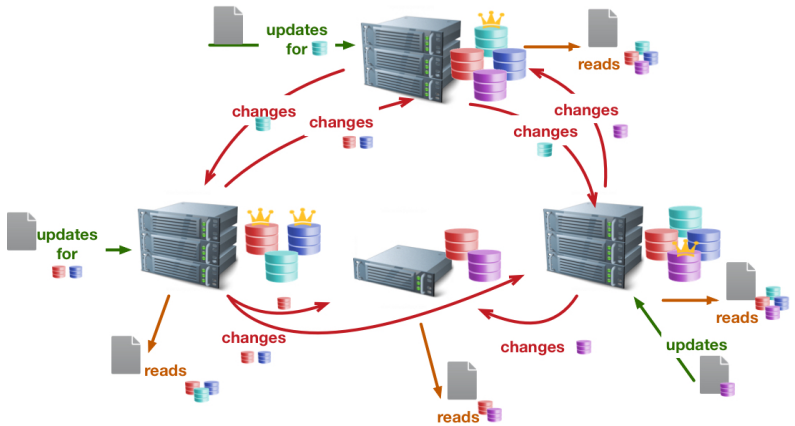
## Cons

- Inconsistency
  - Slow propagation of changes to copies on different nodes
    - Inconsistencies on read lead to problems but are relatively transient
  - Two people can update different copies of the same record stored on different nodes at the same time:
    - a write-write conflict
    - Inconsistent writes are forever

# Sharding + Replication: Master-Slave



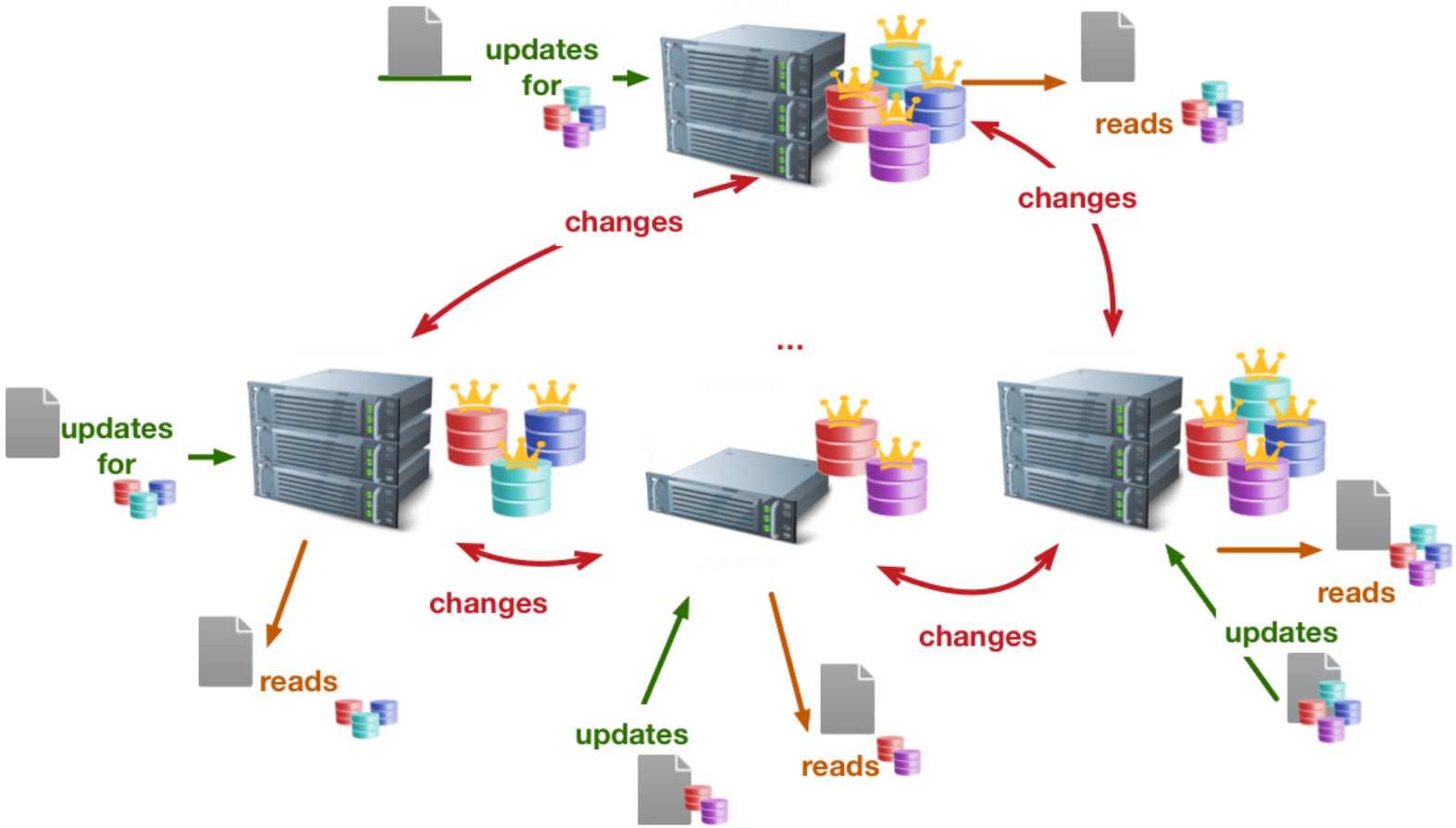
# Sharding + Replication: Master-Slave



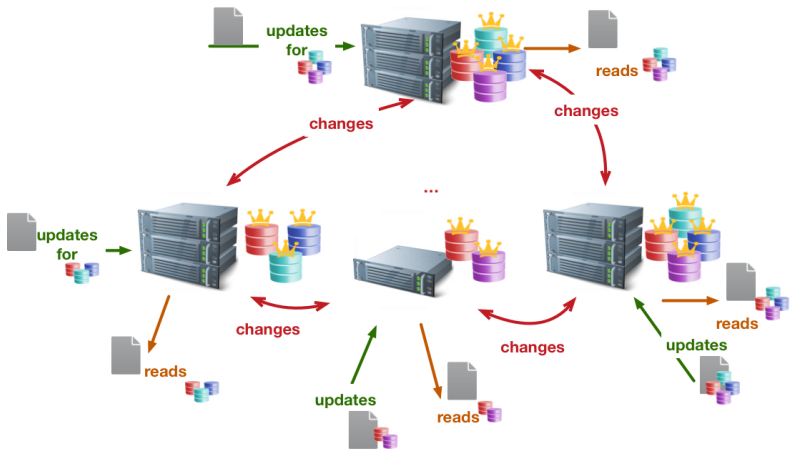

## Master-Slave

- We have multiple masters, but each data only has a single master
- Two schemes:
  - A node can be a master for some data and a slave for some others
  - Nodes are dedicated for master or slave duties

# Sharding + Replication: Peer-to-Peer



# Sharding + Replication: Peer-to-Peer




## Peer-to-Peer

- Usually each shard is replicated on three nodes
- A common strategy for column-family databases

# Summary

There are two styles of distributing data:

## Sharding

- Distributes different data across multiple servers
  - Each server acts as the single source for a subset of data

# Summary

There are two styles of distributing data:

## Sharding

- Distributes different data across multiple servers
  - Each server acts as the single source for a subset of data

## Replication

- Copies data across multiple servers
  - Each bit of data can be found in multiple places

# Summary

There are two styles of distributing data:

## Sharding

- Distributes different data across multiple servers
  - Each server acts as the single source for a subset of data

## Replication

- Copies data across multiple servers
  - Each bit of data can be found in multiple places

A system may use either or both techniques

# Summary

Data replication comes in two forms:

## Master-slave replication

- makes one node the authoritative copy (master)
- the master handles writes
- slaves synchronize with the master and may handle reads

# Summary

Data replication comes in two forms:

## Master-slave replication

- makes one node the authoritative copy (master)
- the master handles writes
- slaves synchronize with the master and may handle reads

## Peer-to-peer replication

- allows writes to any node
- the nodes coordinate to synchronize their copies of the data

# Summary

## Analysis

- Master-slave replication reduces the chance of update conflicts
- Peer-to-Peer replication avoids loading all writes onto a single point of failure

# References I


Francesca Bugiotti, Luca Cabibbo, Paolo Atzeni, and Riccardo Torlone.
Database design for nosql systems.
In *Conceptual Modeling - 33rd International Conference, ER 2014, Atlanta, GA, USA, October 27-29, 2014. Proceedings*, pages 223–231, 2014.

Eric Evans.
*Domain-Driven Design: Tacking Complexity In the Heart of Software*.
Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.