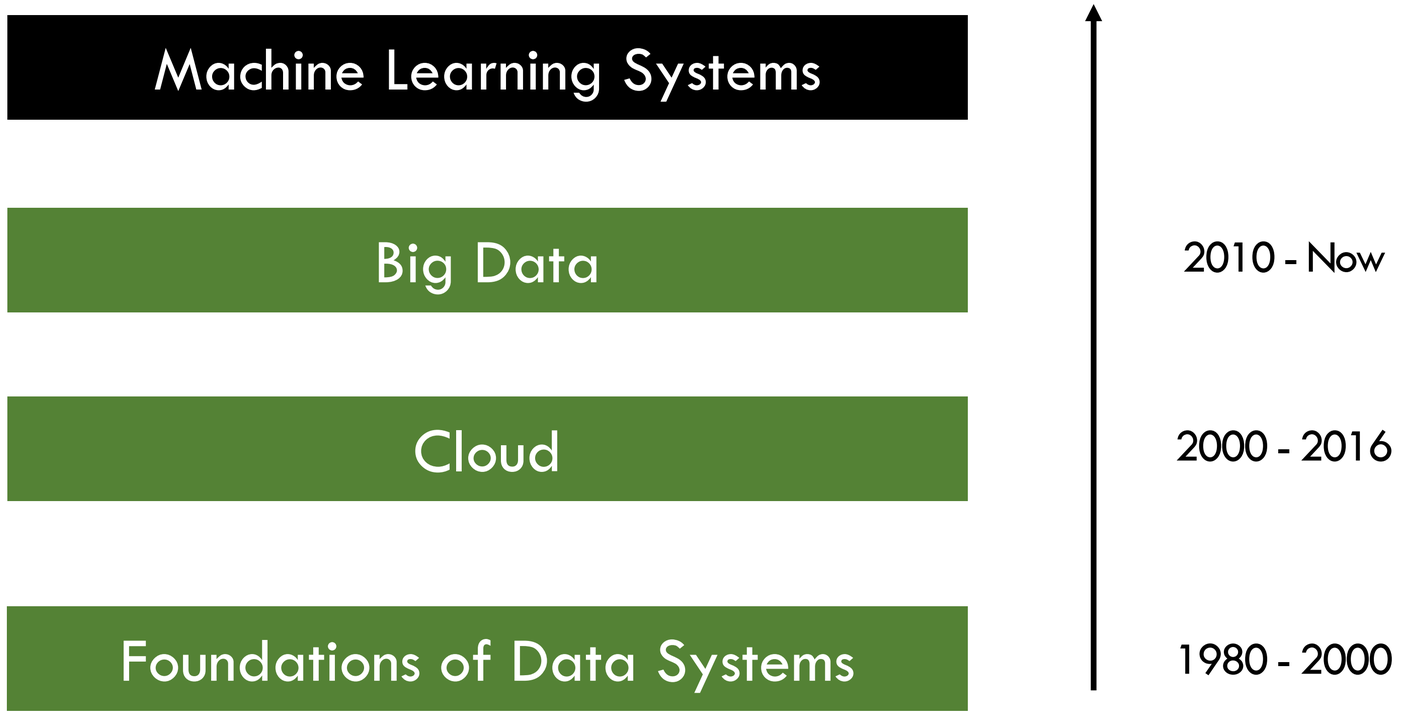
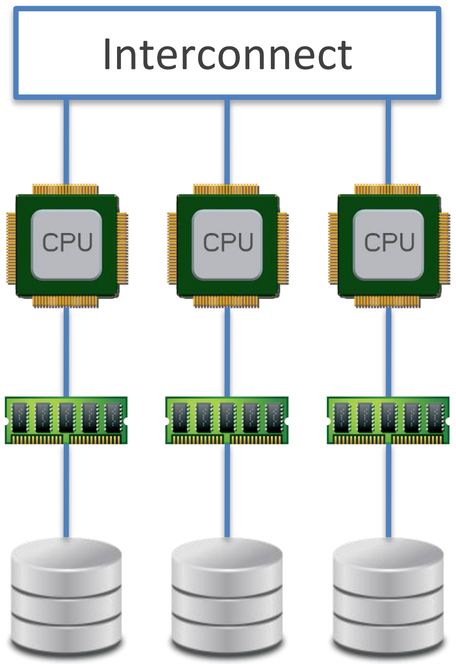
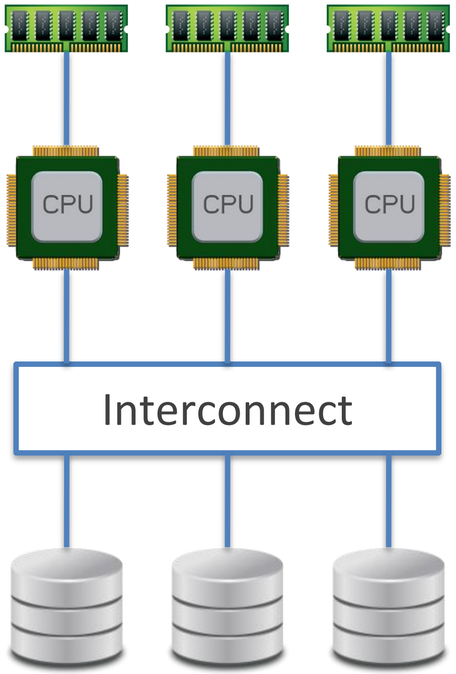
Where We Are


Machine Learning Systems
Big Data
Cloud
Foundations of Data Systems
2010 - Now
2000 - 2016
1980 - 2000

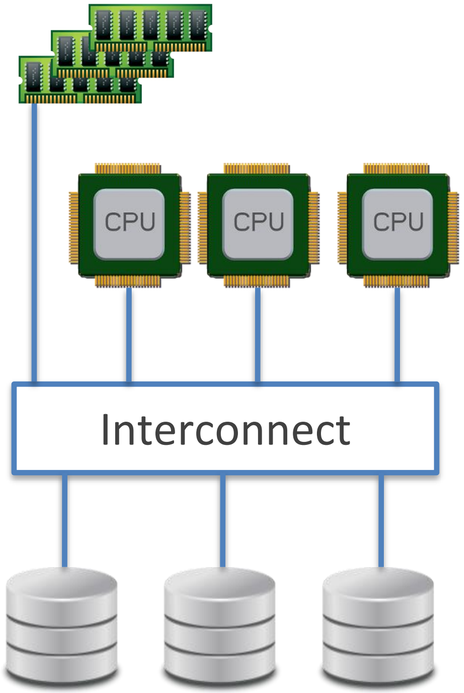
3 Paradigms of Multi-Node Parallelism Implementations



Shared-Nothing Parallelism

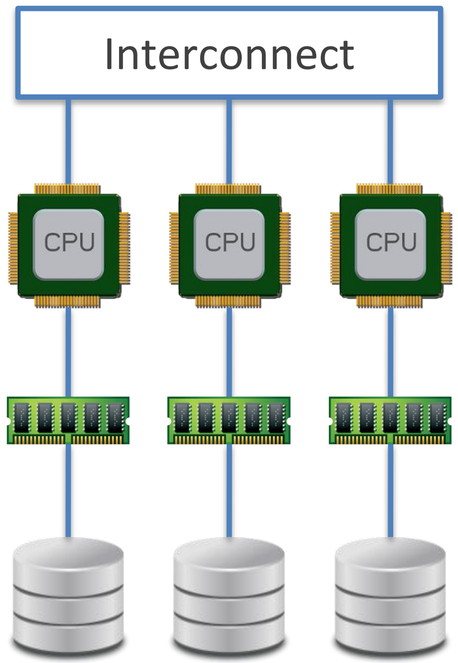


Shared-Disk Parallelism



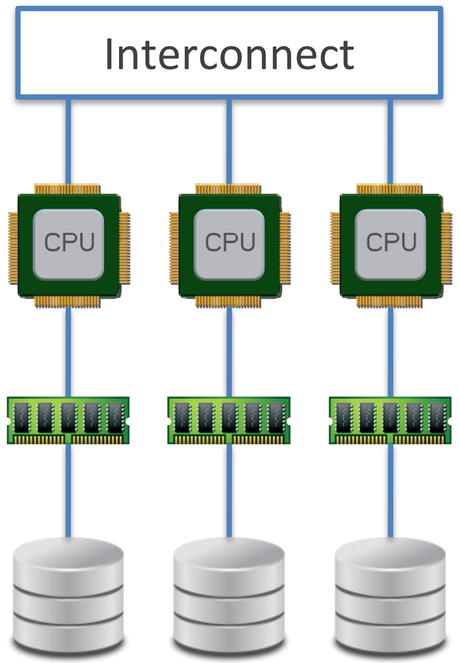
Shared-Memory Parallelism

Shared-Nothing Parallelism (horizontal scaling)



- Most popular approach!
- Each node uses its CPUs, RAM, and disks independently.
  - Any coordination, software level, through a conventional network.
- The vanilla (and most complex!) distributed systems
  - Consistency
  - Communication
  - Coordination

Shared-Nothing Parallelism (horizontal scaling)



- Advantage:
  - Performance
  - Cost
- Disadvantage
  - Complexity.
  - Involves many constraints and trade-offs.
- Database cannot magically hide all these from you.

Metrics to Evaluate Distributed Big Data Systems

- Scalability
  - Data volume
  - Read/Write/Compute load
- Consistency and correctness
  - Read / Write sees consistency data
  - Compute produce correct results
- Fault tolerance / high availability
  - When one fails, another can take over.
- Latency
  - Distribute machines worldwide.
  - Reduce network latency.

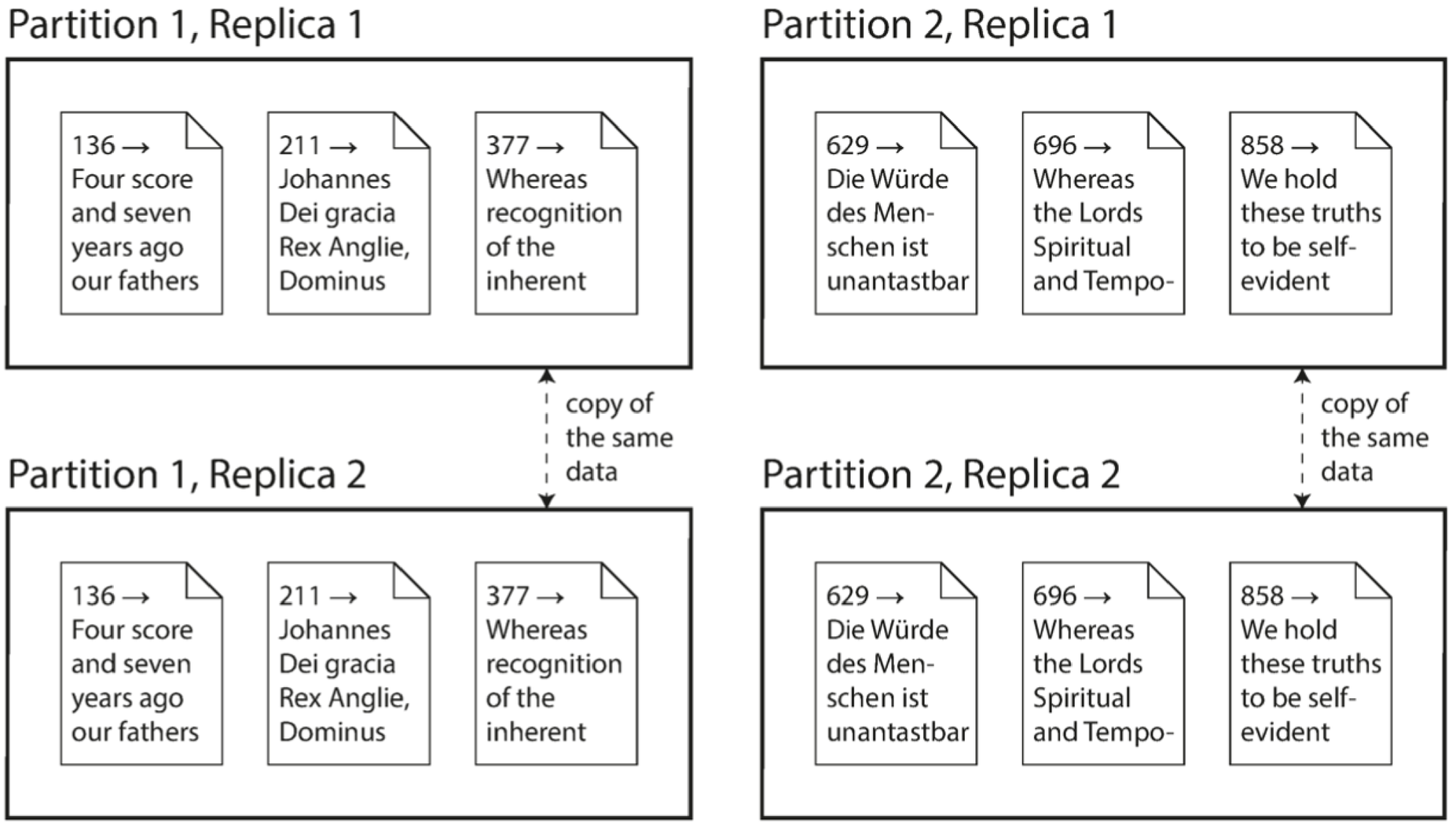
Problems Distributed Systems Need to Solve

- Communication (covered)
- How to Distribute Data?
- How to Distribute Compute?
- How to Coordinate/Synchronize?

Problems Distributed Systems Need to Solve

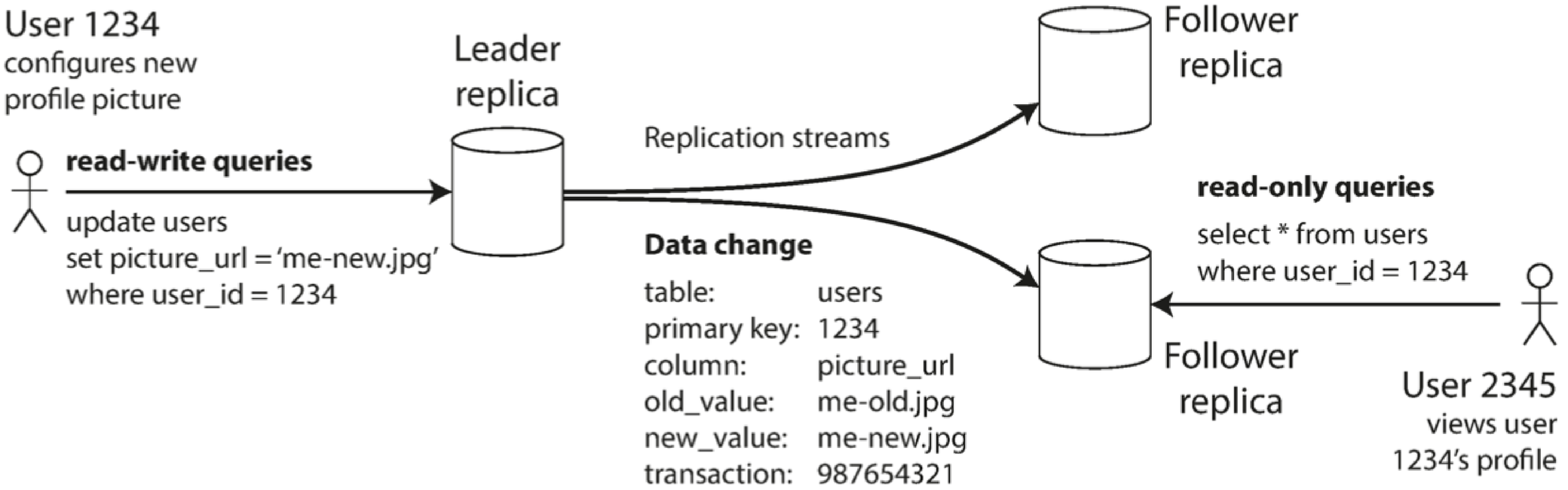
- **How to Distribute Data?**
  - **Replicate / Partition the data**
- How to Distribute Compute?
  - Batch Processing / Steaming processing
- How to Coordinate/Synchronize?
  - Distributed decision making and consensus

Replication versus Partitioning



Challenge: How to handle changes to replicated data?

Leaders and Followers

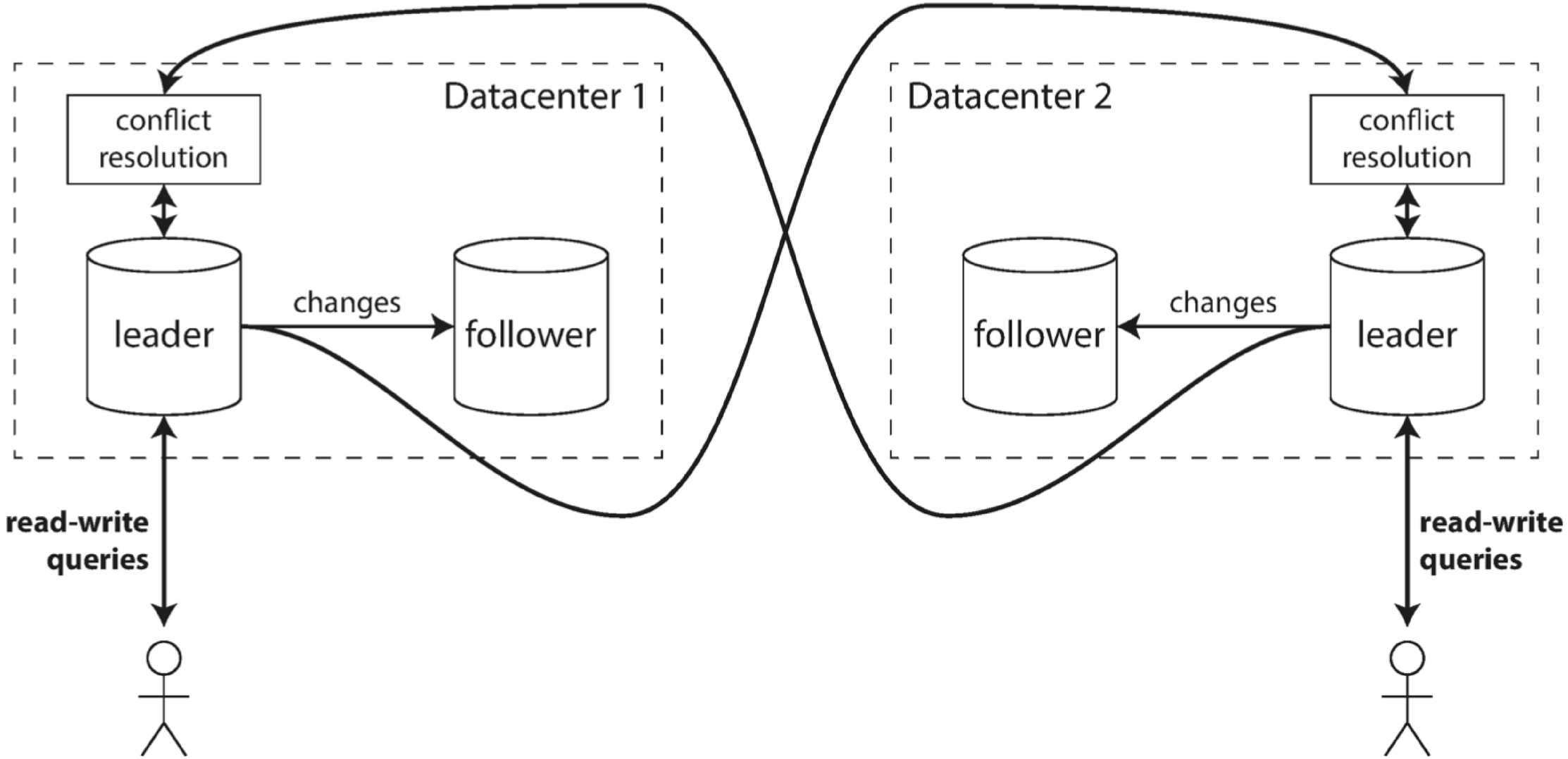


- Clients send requests to the leader to write to the db. The leader saves locally
- The leader then sends the data change to all of its followers.
- Clients read data from the leader or followers.
- One of the big ideas in CS (leader-follower). Distributed msg brokers (e.g., Kafka).

Single-leader replication

- Advantages:
  - Simplicity (easy-to-understand)
  - Conflicts across nodes (consistency)
- Disadvantages:
  - Single point of failure

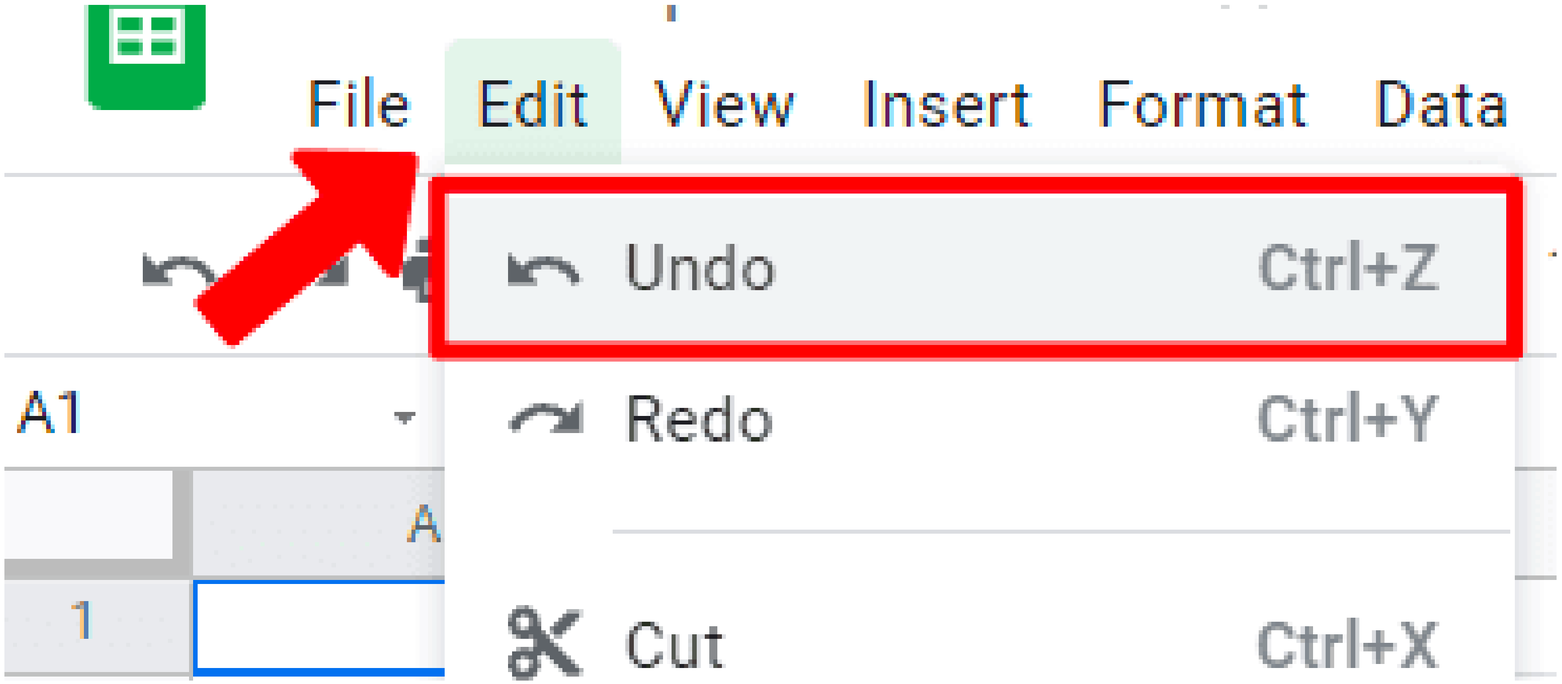
Multi-leader replication (Multi-datacenter operation)



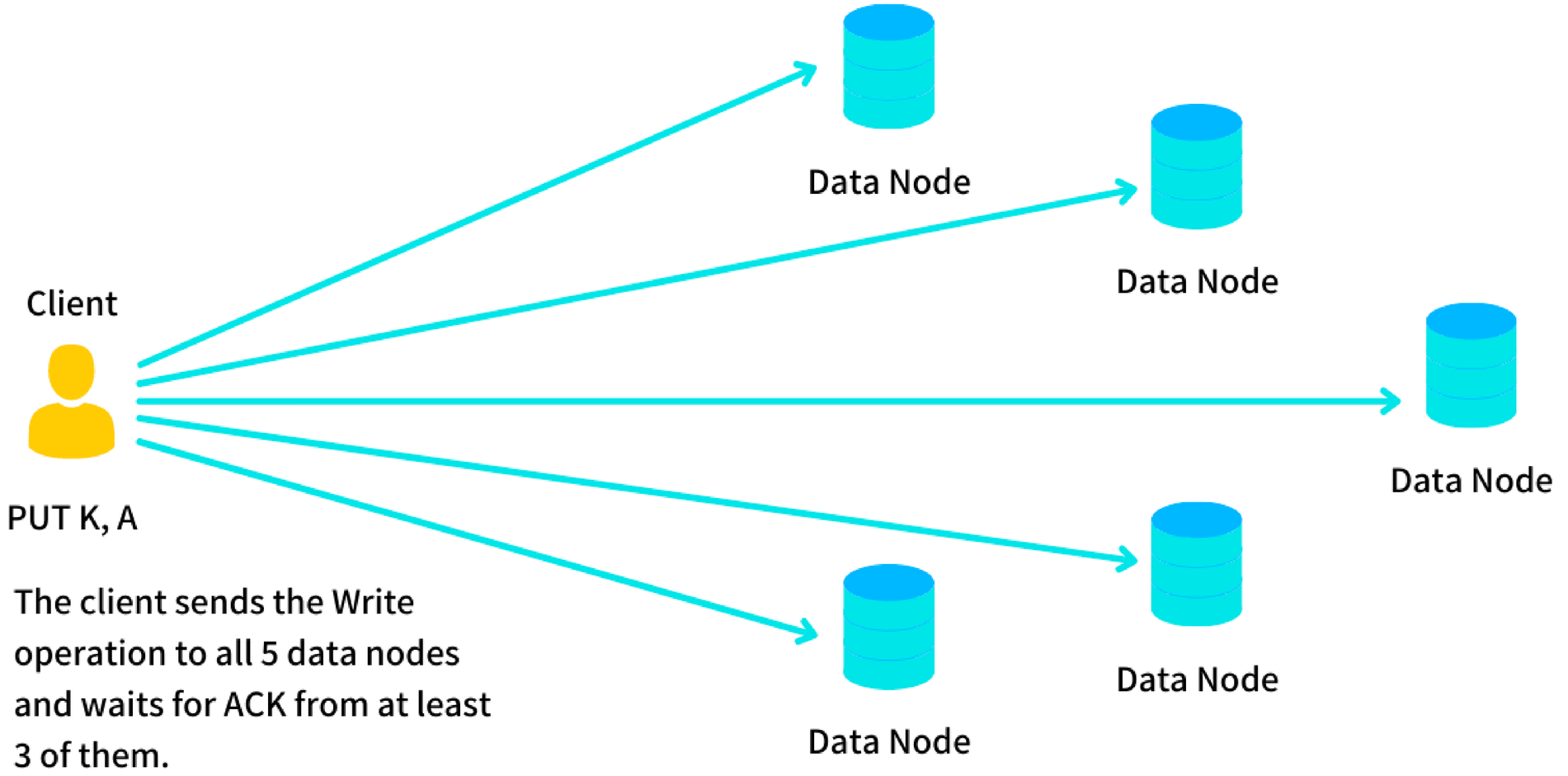
Multi-leader replication

- Advantages:
  - Performance (i.e., network latency).
  - Tolerance of data center outages.
  - Tolerance of network problems.
    - Asynchronous syncing.
- Disadvantages:
  - Potential write conflicts.
    - The same data may be concurrently modified in two different data centers.
- Only use it if necessary.

Google Doc Undo-Redo



Leaderless replication



Challenge: How to handle changes to replicated data?

- Three main approaches:
  - Single-leader replication
  - Multi-leader replication
  - Leaderless replication
- Tradeoffs
  - Simplicity (easy-to-understand)
  - Conflicts across nodes (consistency)
  - Faulty nodes
  - Network interruption
  - Latency spikes

Replication (assuming a small dataset)

- Scalability
  - Data volume
  - Read load
  - Write load
- Fault tolerance | high availability
  - When one fails, another can take over.
- Latency
  - Distribute machines worldwide.
  - Reduce network latency.

Replication (assuming a small dataset)

- Scalability
  - Data volume
  - Read load
  - Write load
- Fault tolerance | high availability
  - When one fails, another can take over.
- Latency
  - Distribute machines worldwide.
  - Reduce network latency.

Partitioning (what if the dataset is too big for a single machine?)

Bigtable: A distributed storage system for structured data [PDF] acm.org

F Chang, J Dean, S Ghemawat, WC Hsieh… - ACM Transactions on …, 2008 - dl.acm.org

… Despite these varied demands, **Bigtable** has … **Bigtable**, which gives clients dynamic control over data layout and format, and we describe the design and implementation of **Bigtable**…

☆ Save 99 Cite Cited by 7813 Related articles All 229 versions

[PDF] The **Google file system** [PDF] acm.org

S Ghemawat, H Gobioff, ST Leung - … symposium on Operating **systems** …, 2003 - dl.acm.org

… the **Google File** System, a scalable distributed **file system** for … goals as previous distributed **file systems**, our design has … departure from some earlier **file system** assumptions. This has led …

☆ Save 99 Cite Cited by 10064 Related articles All 305 versions

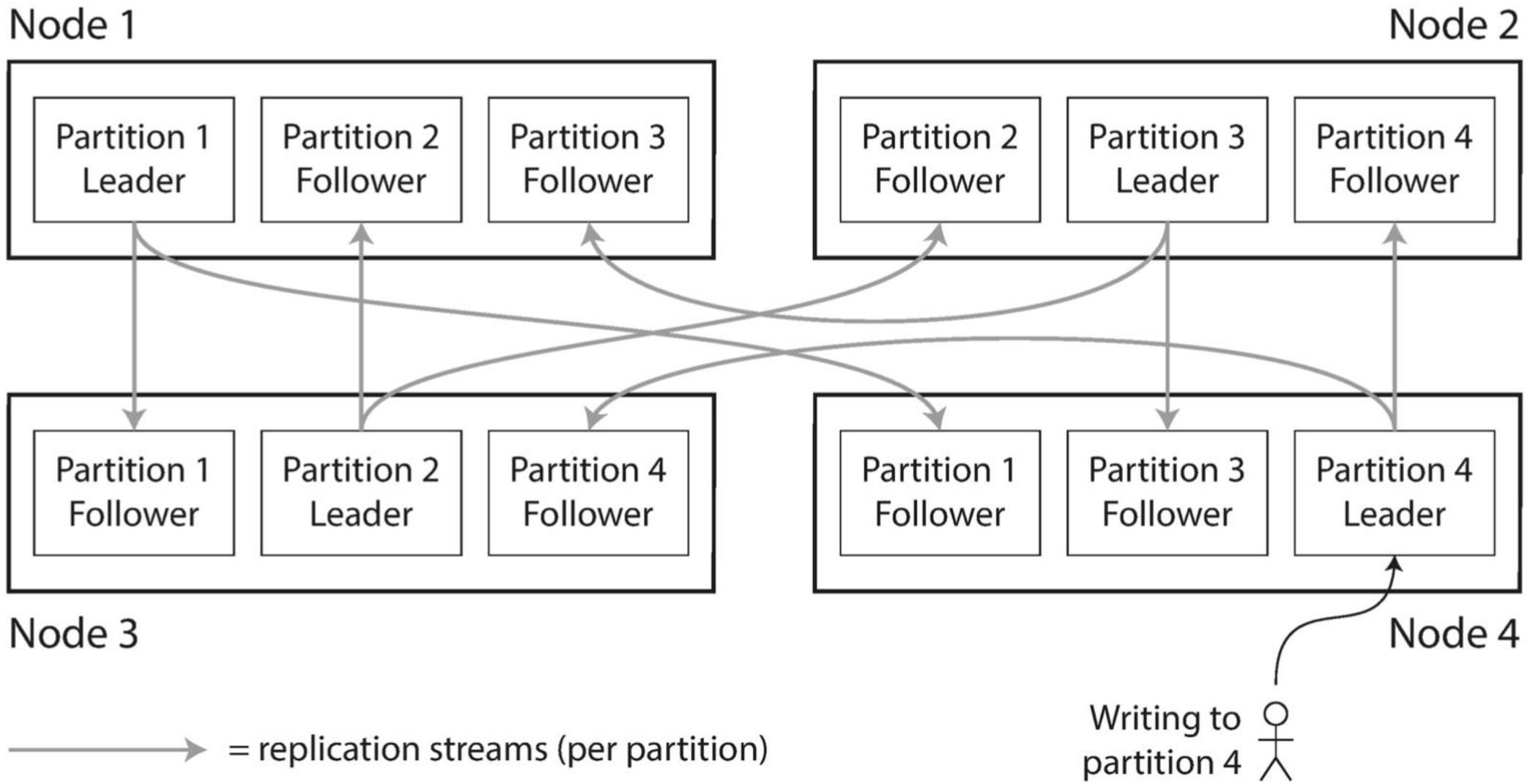
MapReduce: simplified data processing on large clusters [PDF] acm.org

J Dean, S Ghemawat - Communications of the ACM, 2008 - dl.acm.org

… pairs, and then applying a **reduce** operation to all the values … with user-specified **map** and **reduce** operations allows us to … implementation of the **Map Reduce** interface tailored towards …

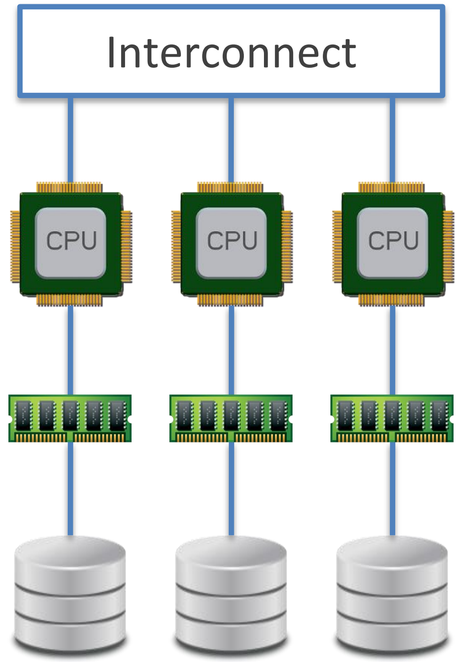
☆ Save 99 Cite Cited by 22507 Related articles All 96 versions

Combining replication and partitioning



Intuitions behind partitioning (why?)

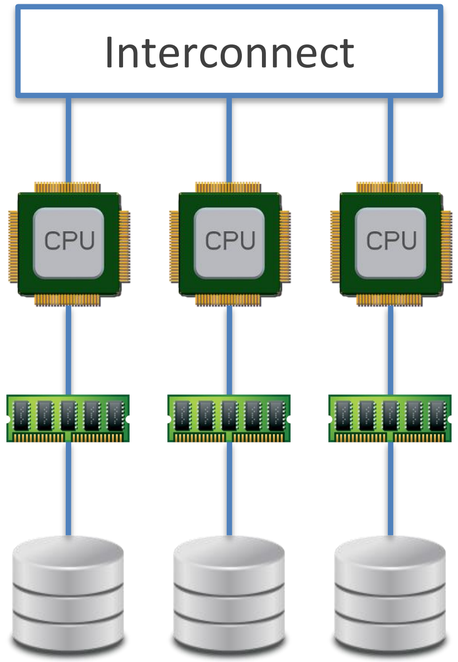
- A large dataset can be distributed across many disks.
- The query load can be distributed across many processors.



Shared-Nothing Parallelism

Partitioning challenges

- How to partition and how to index?
- How to add or remove nodes?
- How to route the requests and execute queries?

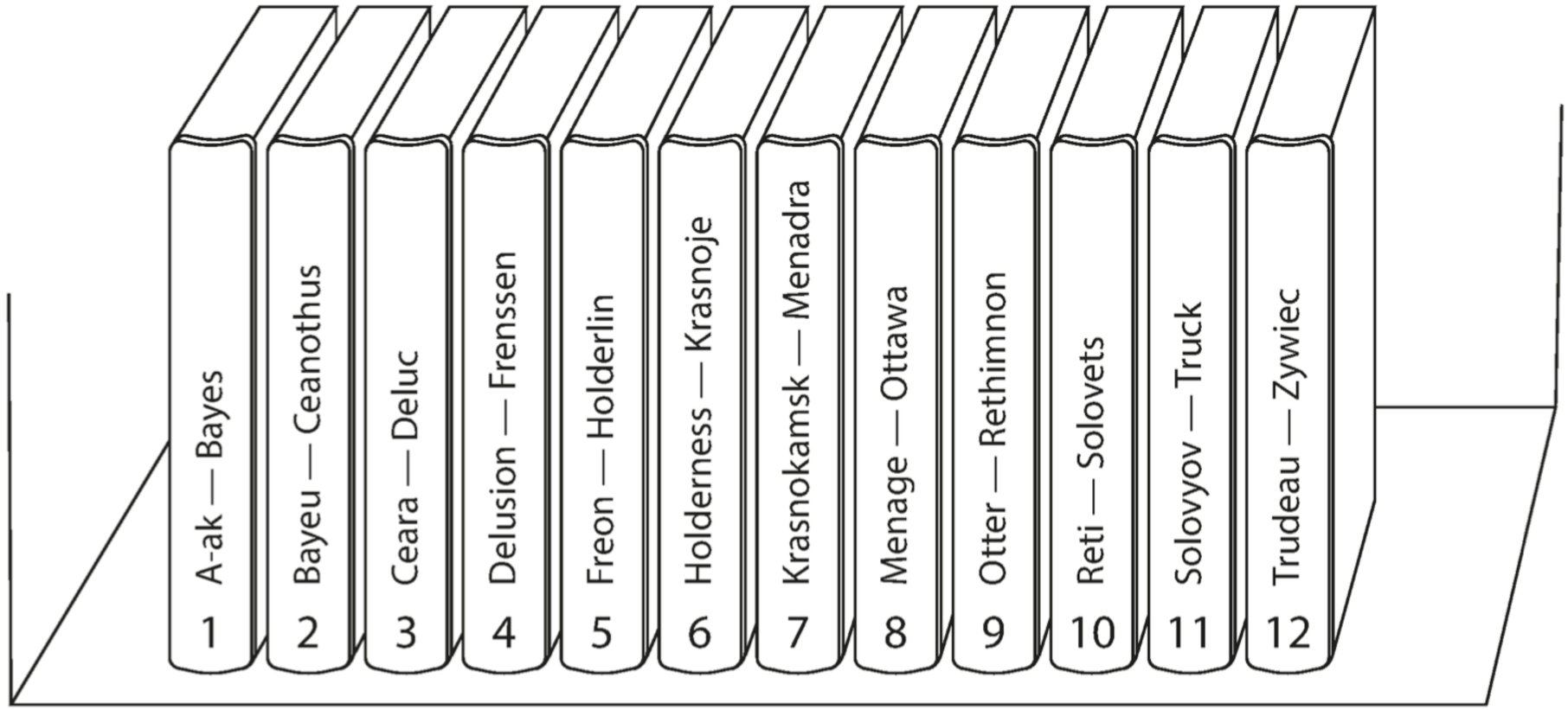


Shared-Nothing Parallelism

How to partition?

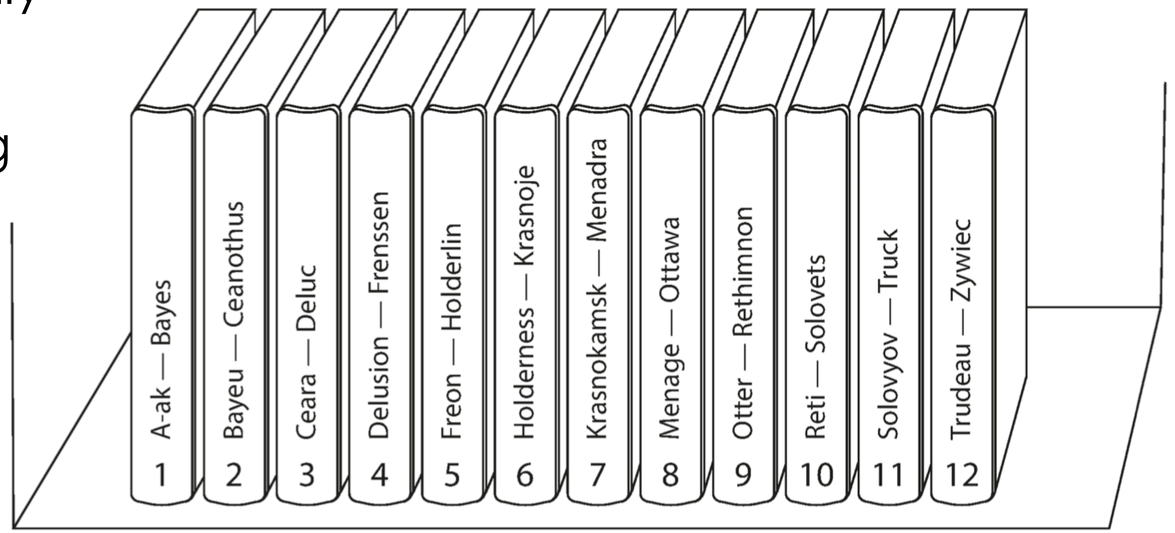
- Ideally:
  - Spread the data and the query load evenly across nodes.
- Reality:
  - Hot spot: a partition with disproportionately high load.
- Straw-man solution:
  - Distribute randomly.
  - Problem:
    - When you read a data, you do not know which node you should request. You have to request all the nodes.

Partition solution #1: by Key Range

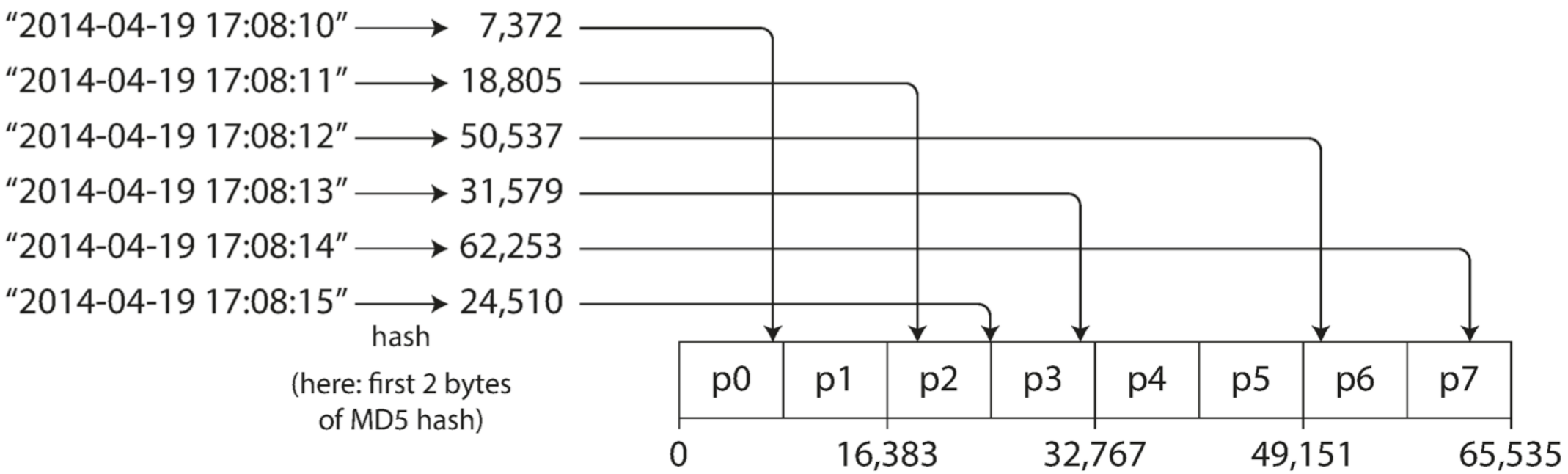


Partition solution #1: by Key Range

- Advantage: can do range queries
- Problems:
  - The ranges of keys are not necessarily evenly spaced.
    - Volume 12 contains words starting with T - Z.
  - Manual process. Require domain expertise.
  - Hard to rebalance.
  - Hot spot issues.
    - One letter is popular.
- Common keys: names, titles, dates.



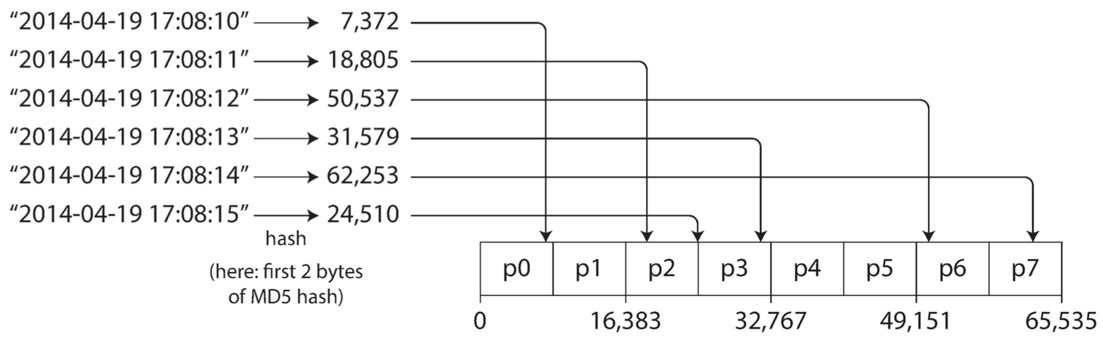
Partition solution #2: by Hash of Key



Recall bloom filter.

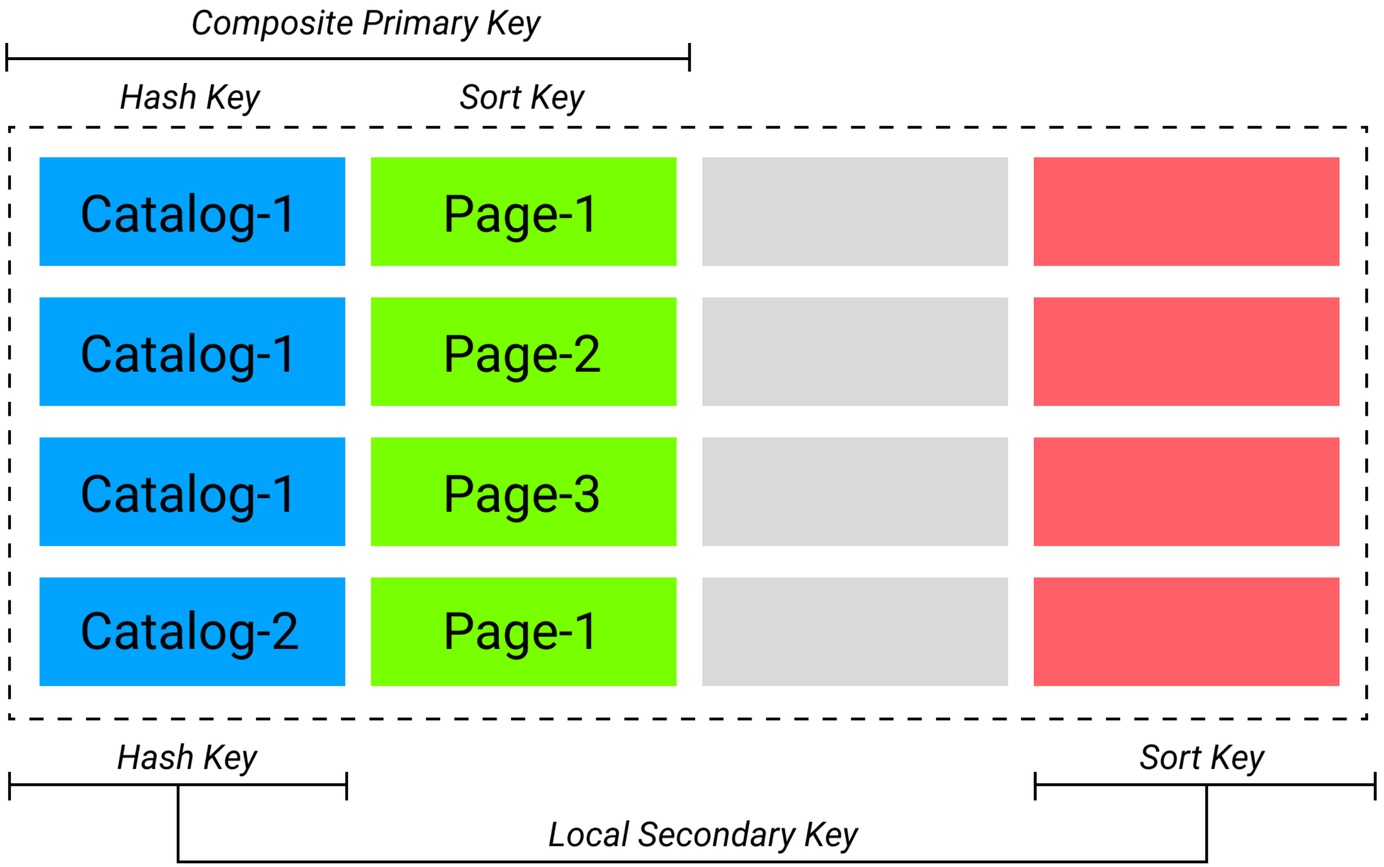
Partition solution #2: by Hash of Key

- Advantages:
  - Automatic.
  - Easy to balance.
- Problems:
  - Do not support efficient range queries.



Recall bloom filter.

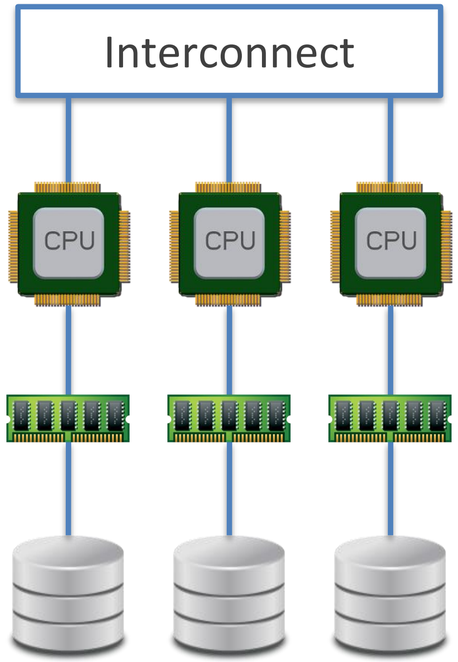
Partition solution #3: Hash of Key + Key Range



user_id update_timestamp

Partitioning challenges

- How to partition and how to index?
- How to add or remove nodes?
- How to route the requests and execute queries?

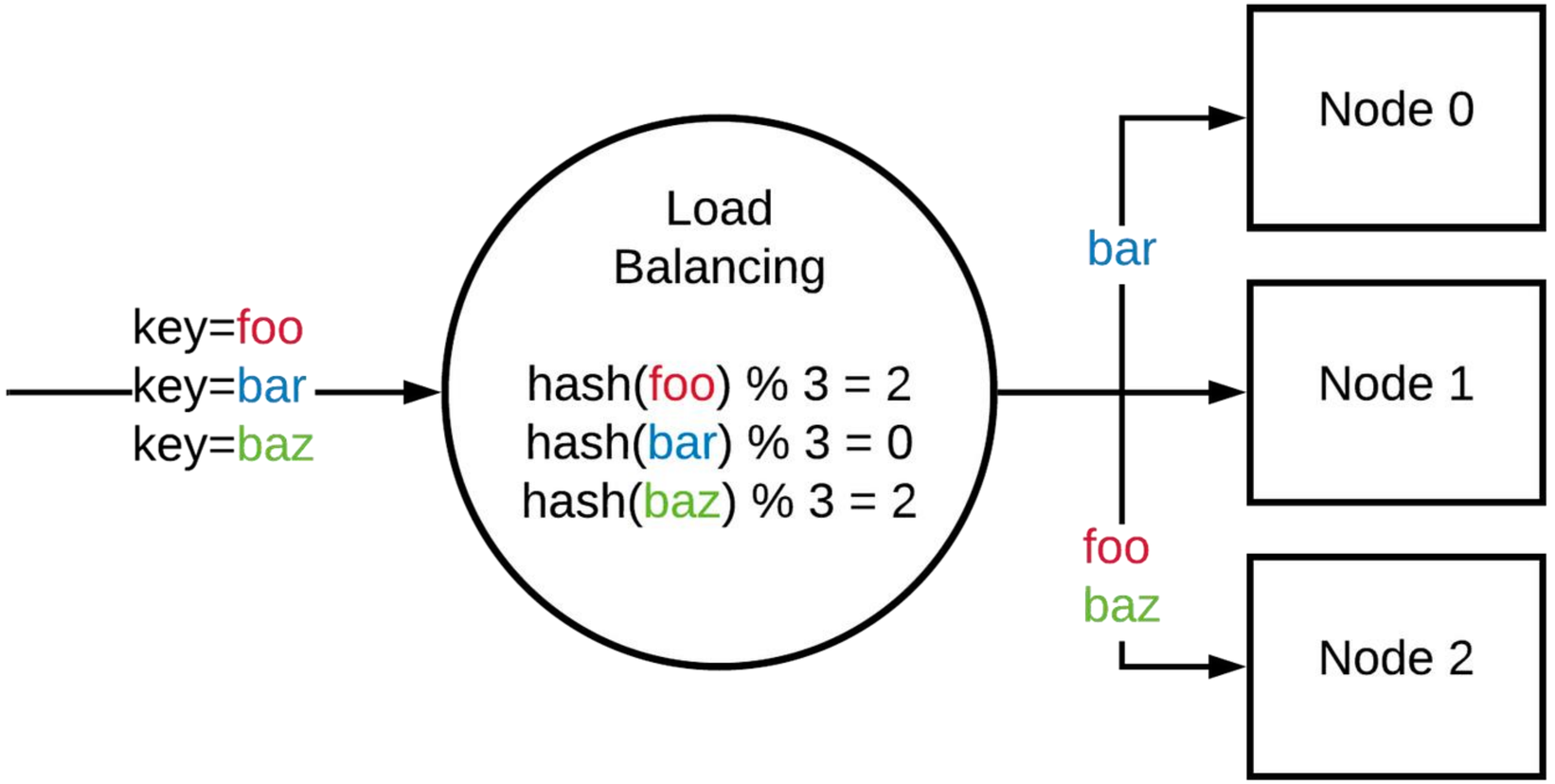


Shared-Nothing Parallelism

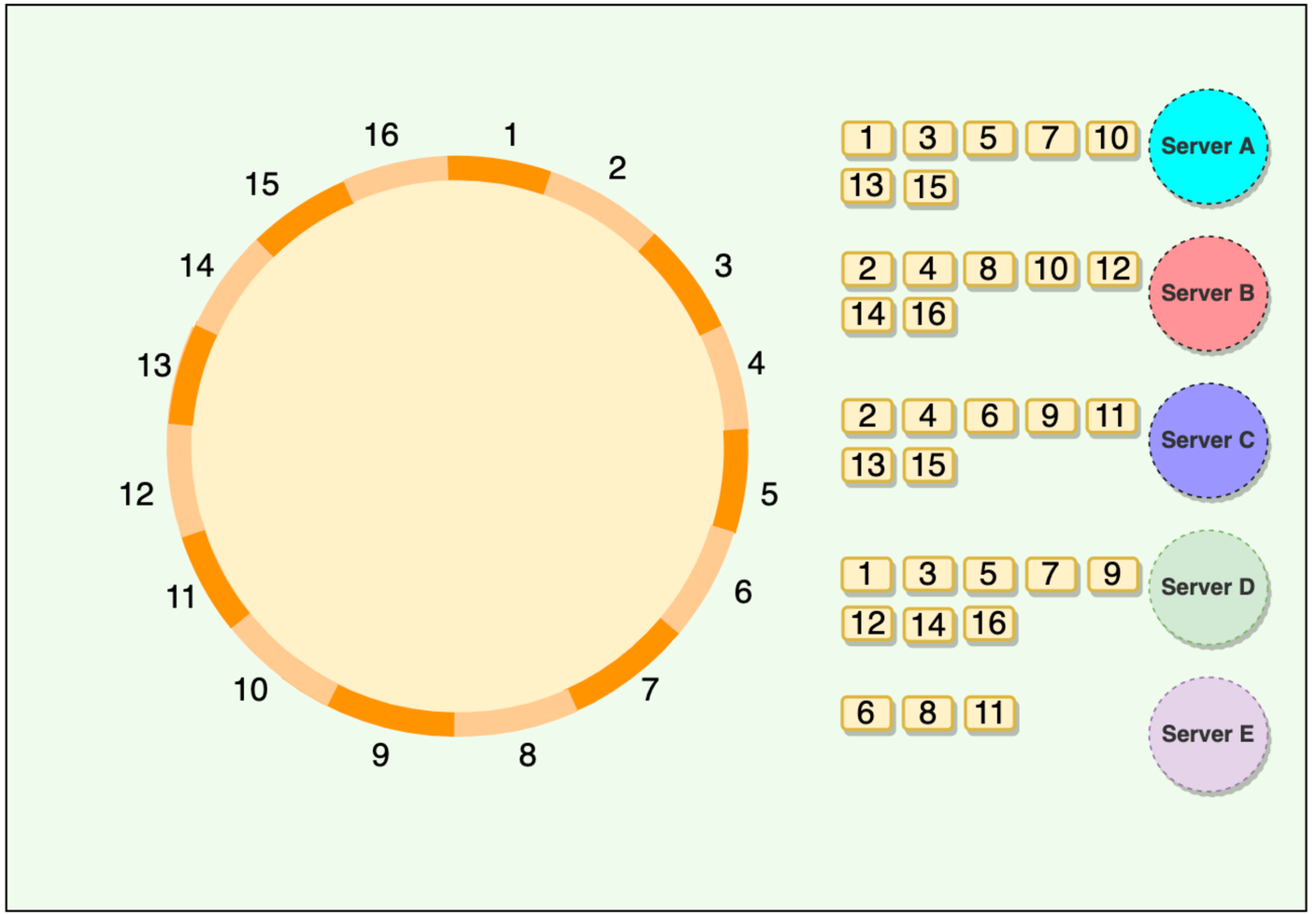
Rebalance

- Move the load from one node in a cluster to another
  - The query throughput increases → more CPUs
  - The dataset size increases → more disks and RAM
  - Machine failure
- Rebalancing goals
  - Share the load fairly after rebalancing.
  - Continue accepting reads and writes while rebalancing.
  - Minimize data moving (i.e., network and disk I/O load)

Strawman solution: Hash mod N



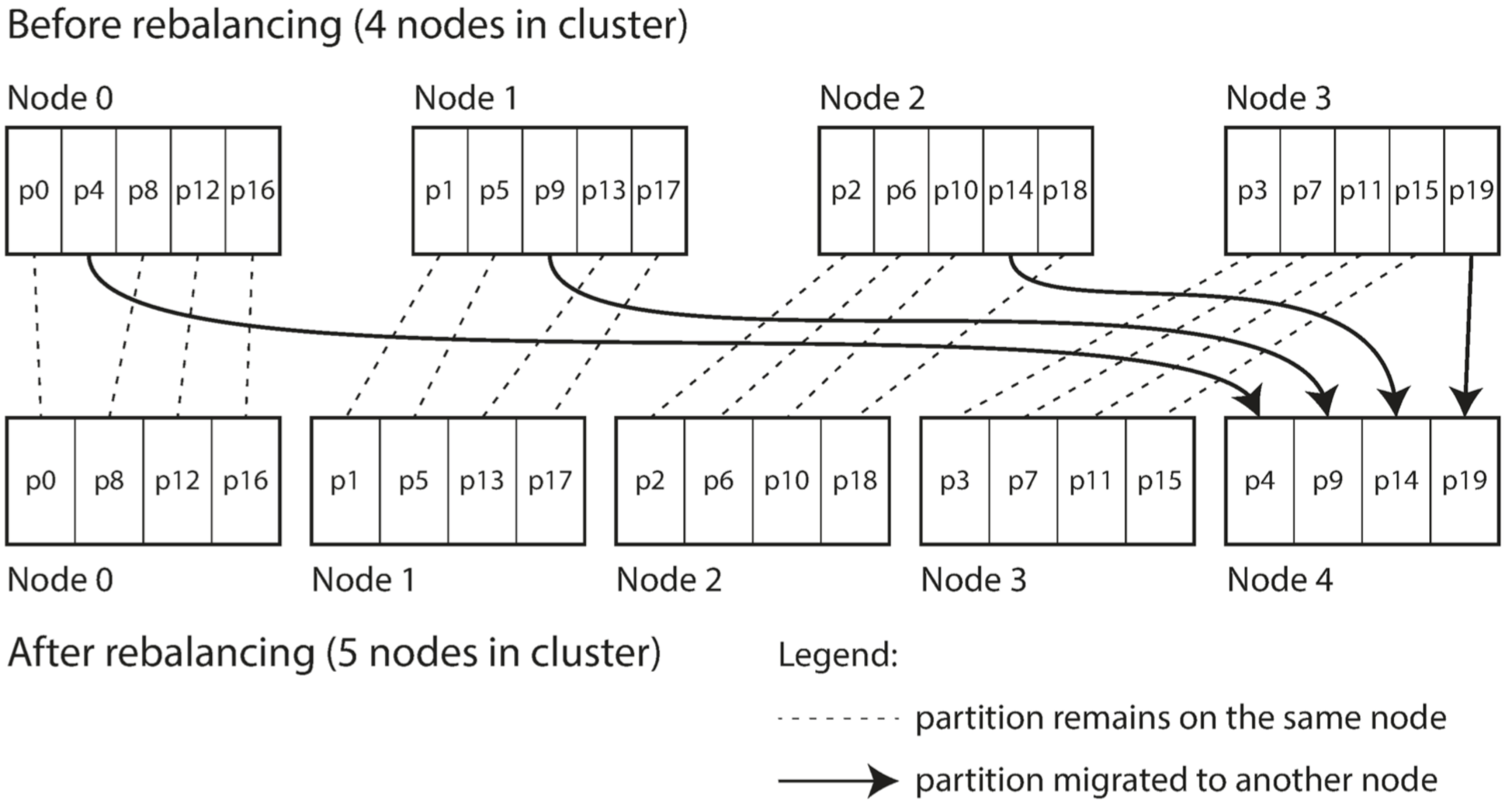
Consistent hashing ring



Mapping Vnodes to physical nodes on a Consistent Hashing ring

Learn more in https://www.youtube.com/watch?v=UF9Iqmg94tk

Rebalancing solution #1: Fixed number of partitions



Rebalancing solution #1: Fixed number of partitions

- The total number of partitions is fixed.
- The # of nodes can be adaptive for different machines.
- Partitions should have similar sizes => why?
  - Easier for management.
- Each partition grows proportionally to the total amount of data.
- Challenges:
  - Need to choose the right number of partitions.
    - Too high → too much overhead
    - Too low → Migration will be very expensive.
  - Wrong index system.
    - Very unbalanced distributions.
  - Only work with hash partitioned database. => Why?

Rebalancing solution #2: Dynamic partitions

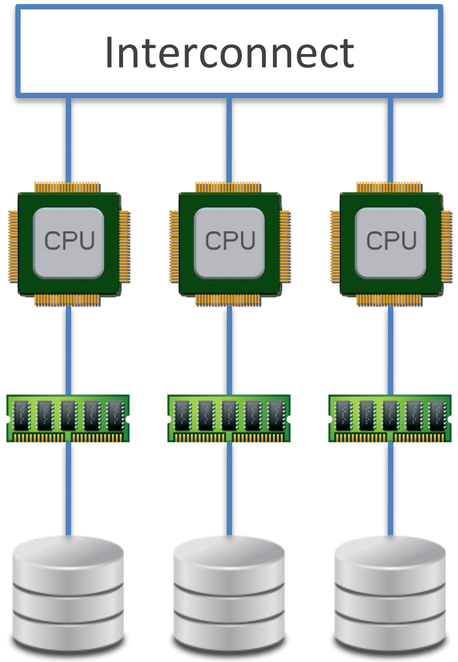
- Similar idea as the B-tree.
- Split ← When a partition grows to exceed a configured size (e.g., 10 GB).
- Merge ← When lots of data is deleted and a partition shrinks.
- One node → multiple partitions.
- One partition → one node.
- A caveat:
  - By default, start → an empty database → one partition → only one node.
  - Some systems allow pre-splitting.
- Dynamic partition works for both key range and hash partitioned data.

Automatic or Manual Rebalancing

- Mostly manual → Why?
- Rebalancing is very expensive!
- Some suggestive interfaces exist.

Partitioning challenges

- How to partition and how to index?
- How to add or remove nodes?
- How to route the requests and execute queries?



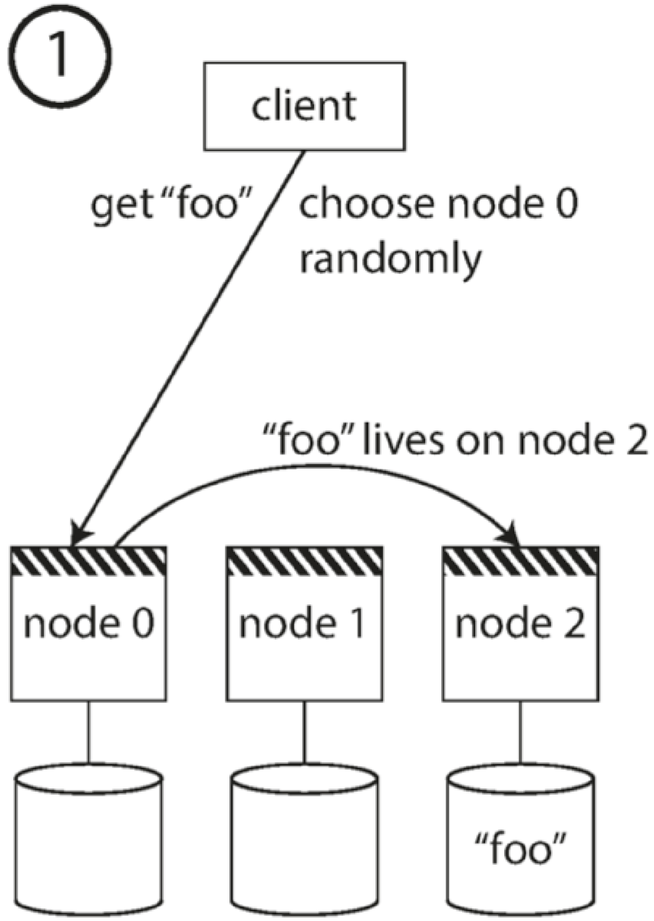
Shared-Nothing Parallelism

Two questions

- Which node to connect to?
- Where to maintain the knowledge of rebalanced results?

Routing paradigm #1

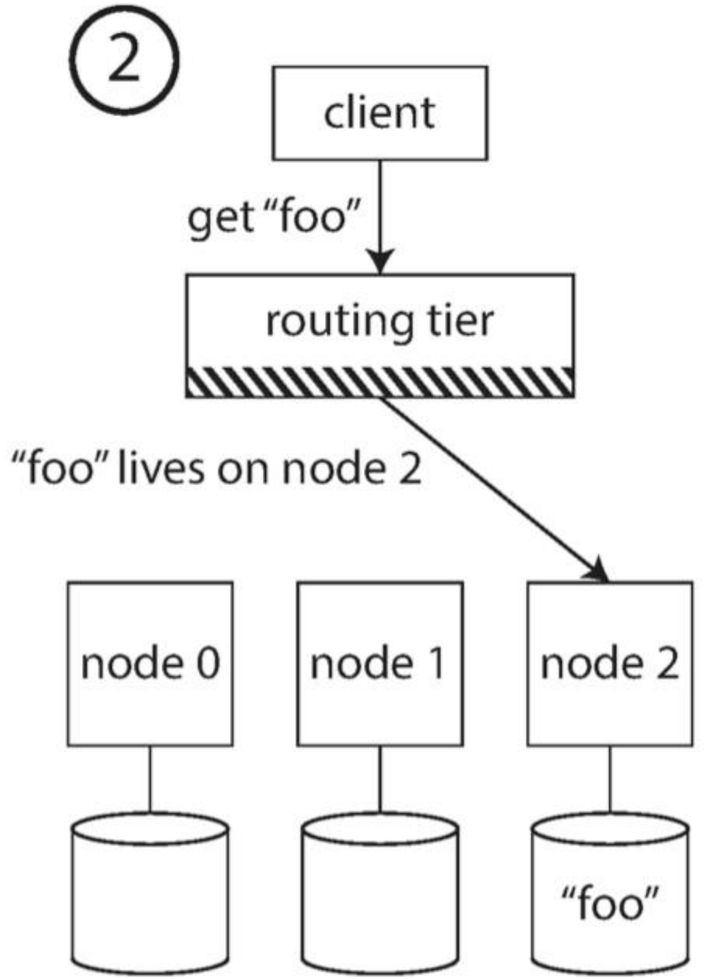
- Contact any node (e.g., a round-robin load balancer),
- If the node has the data copy, respond.
- If not, forward, receives the reply, and passes the reply along to the client.



= the knowledge of which partition is assigned to which node

Routing paradigm #2

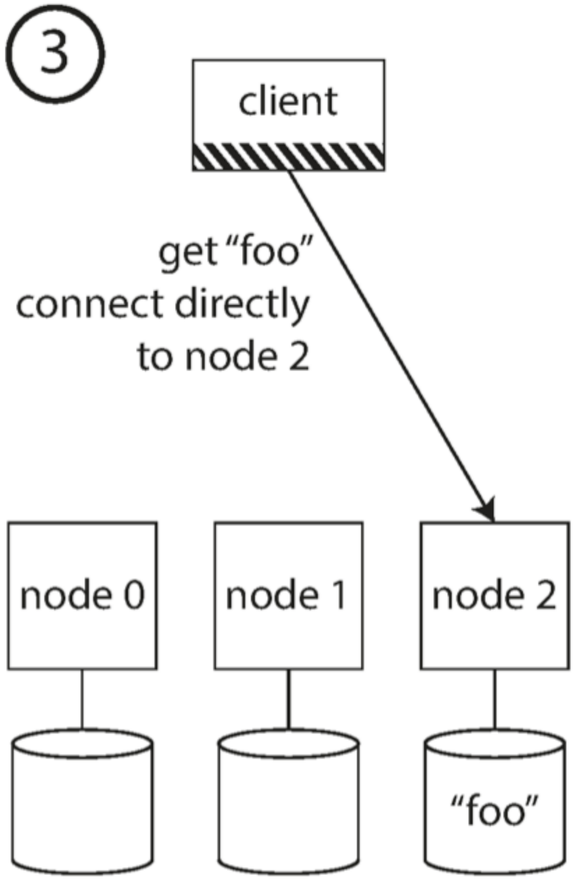
- Send all requests to a routing tier first.
- The routing tier forward all the requests.



= the knowledge of which partition is assigned to which node

Routing paradigm #3

- The client is aware of the partitioning and the assignment of partitions to nodes.
- No intermediary.

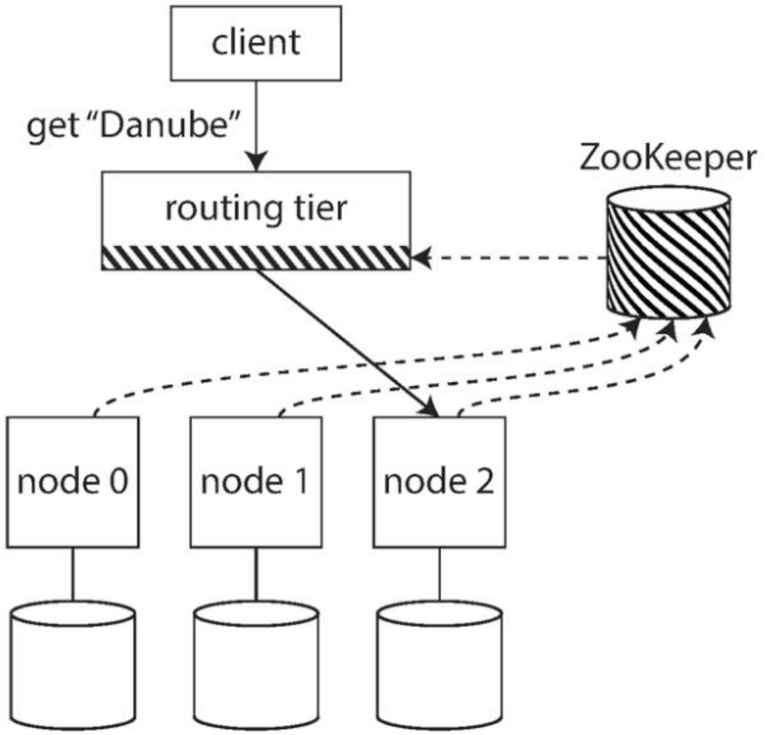


= the knowledge of which partition is assigned to which node

Two questions

- Which node to connect to?
- Where to maintain the knowledge of rebalanced results?

ZooKeeper

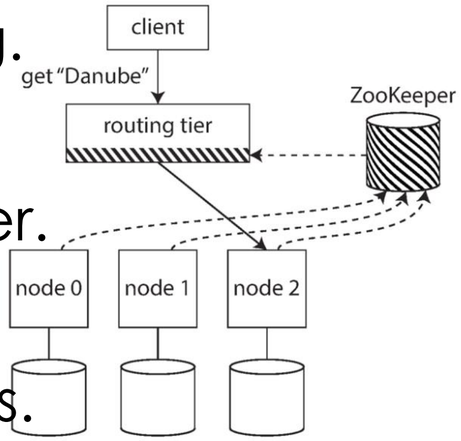


Key range	Partition	Node	IP address
A-ak — Bayes	partition 0	node 0	10.20.30.100
Bayeu — Ceanothus	partition 1	node 1	10.20.30.101
Ceara — Deluc	partition 2	node 2	10.20.30.102
Delusion — Frenssen	partition 3	node 0	10.20.30.100
Freon — Holderlin	partition 4	node 1	10.20.30.101
Holderness — Krasnoje	partition 5	node 2	10.20.30.102
Krasnokamsk — Menadra	partition 6	node 0	10.20.30.100
Menage — Ottawa	partition 7	node 1	10.20.30.101
Otter — Rethimnon	partition 8	node 2	10.20.30.102
Reti — Solovets	partition 9	node 0	10.20.30.100
Solovyov — Truck	partition 10	node 1	10.20.30.101
Trudeau — Zywiec	partition 11	node 2	10.20.30.102

= the knowledge of which partition is assigned to which node

ZooKeeper

- Each node registers itself in ZooKeeper.
- ZooKeeper maintains the mapping.
- Other actors (different in three paradigms) subscribe to ZooKeeper.
- Whenever the partition mapping changes, ZooKeeper notifies actors.



Key range	Partition	Node	IP address
A-ak — Bayes	partition 0	node 0	10.20.30.100
Bayeu — Ceanothus	partition 1	node 1	10.20.30.101
Ceara — Deluc	partition 2	node 2	10.20.30.102
Delusion — Frenssen	partition 3	node 0	10.20.30.100
Freon — Holderlin	partition 4	node 1	10.20.30.101
Holderness — Krasnoje	partition 5	node 2	10.20.30.102
Krasnokamsk — Menadra	partition 6	node 0	10.20.30.100
Menage — Ottawa	partition 7	node 1	10.20.30.101
Otter — Rethimnon	partition 8	node 2	10.20.30.102
Reti — Solovets	partition 9	node 0	10.20.30.100
Solovyov — Truck	partition 10	node 1	10.20.30.101
Trudeau — Zywiec	partition 11	node 2	10.20.30.102

= the knowledge of which partition is assigned to which node

Takeaway

- The benefits of Partitioning and Replication.
- The challenges of Partitioning and Replication.
- The tradeoffs of different strategies.
- Replication: single-leader, multiple-leader, leaderless
- Partition: Key range, hash, hybrid.
  - Partition rebalancing strategies: fixed, dynamic
  - Partition routing, ZooKeeper