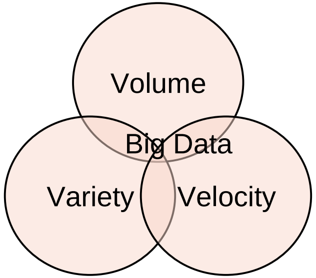
PRINCIPLES OF DATA ORGANISATION

Distributed Data

BIG DATA

WHAT IS BIG DATA?



- No standard definition
- First occurrence of the term: High Performance Computing (HPC)

Gartner: *"**Big Data**" is high **v**olume, high **v**elocity, and/or high **v**ariety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization.*

3 (4, 5)
Vs

http://www.gartner.com/

WHO IS Gartner ?

- Information technology research and advisory company
- Founded in 1979 by Gideon Gartner
- HQ in Stanford, Connecticut, USA
  - > 5,300 employees
  - > 12,400 client organizations
- Provides: competitive analysis reports, industry overviews, market trend data, product evaluation reports, …

http://www.ibmbigdatahub.com/

WHAT IS BIG DATA?

Mobile devices
(tracking all objects all the time)

Social media and networks
(all of us are generating data)

Scientific instruments
(collecting all sorts of data)

Sensor technology and networks
(measuring all kinds of data)

IBM: *Depending on the industry and organization, **Big Data** encompasses information from internal and external sources such as transactions, social media, enterprise content, sensors, and mobile devices.*
Companies can leverage data to adapt their products and services to better meet customer needs, optimize operations and infrastructure, and find new sources of revenue.

FACEBOOK BY THE NUMBERS: STATS, DEMOGRAPHICS & FUN FACTS (LAST UPDATE: APRIL 2020)

- 2.5 billion monthly active users
- 5 billion comments are left on Facebook pages monthly
- 55 million status updates are made every day
- Every 60 seconds
  - 317,000 status updates
  - 147,000 photos uploaded
  - 54,000 shared links

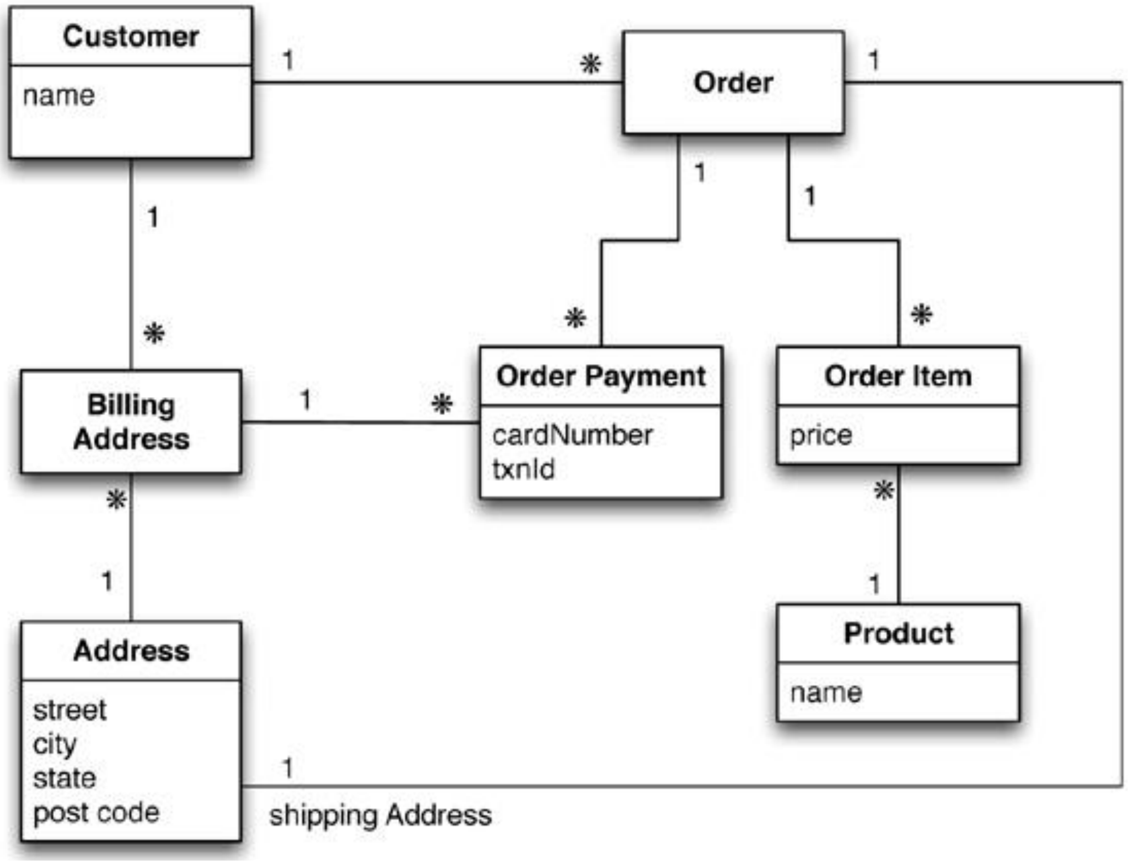
https://www.omnicoreagency.com/facebook-statistics/

AGGREGATES

- Data model = the model by which the database organizes data
- Aggregate
  - A data unit with a complex structure
    - Not just a set of tuples like in RDBMS
  - Domain-Driven Design: "an aggregate is a collection of related objects that we wish to treat as a unit"
    - A unit for data manipulation and management of consistency

AGGREGATE-IGNORANT APPROACHES

- There is no universal strategy how to draw aggregate boundaries
  - Depends on how we manipulate the data
- Relational databases are aggregate-ignorant
  - It is not a bad thing, it is a feature
  - Allows to easily look at the data in different ways
  - Better choice when we do not have a primary structure for manipulating data



Customer
name
Order
Billing Address
Order Payment
cardNumber
txnId
Order Item
price
Product
name
Address
street
city
state
post code
shipping Address
1
*

Customer	
Id	Name
1	Martin

Orders		
Id	CustomerId	ShippingAddressId
99	1	77

Product	
Id	Name
27	NoSQL Distilled

BillingAddress		
Id	CustomerId	AddressId
55	1	77

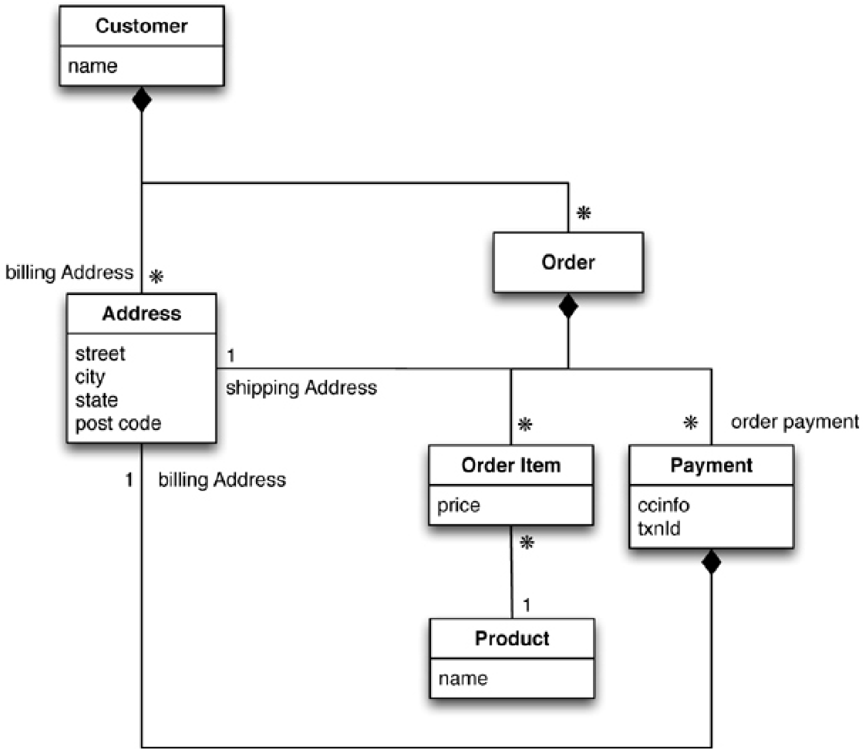
OrderItem			
Id	OrderId	ProductId	Price
100	99	27	32.45

Address	
Id	City
77	Chicago

OrderPayment				
Id	OrderId	CardNumber	BillingAddressId	txnId
33	99	1000-1000	55	abelif879rft

AGGREGATE-ORIENTED APPROACHES

- Aggregate orientation
  - Aggregates give the information about which bits of data will be manipulated together
    - Which should live on the same node
  - Helps greatly with running on a cluster
    - We need to minimize the number of nodes we need to query when we are gathering data
- Consequence for transactions
  - NoSQL (non-relational) databases support atomic manipulation of a single aggregate at a time




```
// in customers
{
"customer": {
"id": 1,
"name": "Martin",
"billingAddress": [{"city": "Chicago"}],
"orders": [
  {
    "id":99,
    "customerId":1,
    "orderItems":[
    {
    "productId":27,
    "price": 32.45,
    "productName": "NoSQL Distilled"
    }
  ],
  "shippingAddress":[{"city":"Chicago"}]
  "orderPayment":[
    {
    "ccinfo":"1000-1000-1000-1000",
    "txnId":"abelif879rft",
    "billingAddress": {"city": "Chicago"}
    }],
  }]
}
}
```

BASIC PRINCIPLES

MAIN PROBLEM: SCALABILITY

Vertical Scaling (scaling up)

- Traditional choice has been in favour of strong consistency
  - System architects have in the past gone in favour of scaling up (vertical scaling)
    - Involves larger and more powerful machines
- Works in many cases but…
- Vendor lock-in
  - Not everyone makes large and powerful machines
    - Who do, often use proprietary formats
  - Makes a customer dependent on a vendor for products and services
    - Unable to use another vendor

Horizontal Scaling (scaling out)

- Systems are distributed across multiple machines/nodes (horizontal scaling)
  - Commodity machines (cost effective)
  - Often surpasses scalability of vertical approach
- But…
- Fallacies of distributed computing:
  - The network is reliable
  - Latency is zero
  - Bandwidth is infinite
  - The network is secure
  - Topology does not change
  - There is one administrator
  - Transport cost is zero
  - The network is homogeneous

DISTRIBUTION MODELS

- Scaling out = running the database on a cluster of servers
- Two orthogonal techniques to data distribution:
  - Replication – takes the same data and copies it over multiple nodes
    - Master-slave or peer-to-peer
  - Sharding – puts different data on different nodes
- We can use either or combine them

DISTRIBUTION MODELS

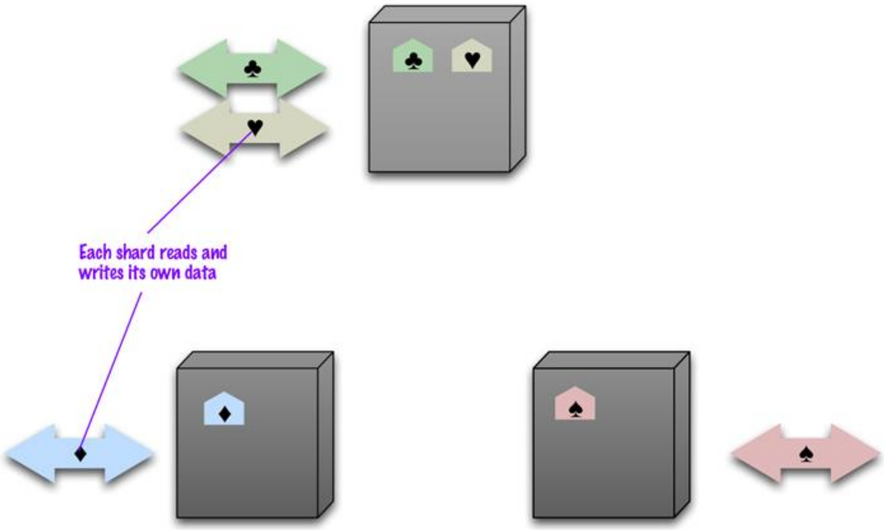
SINGLE SERVER

- No distribution at all
  - The database runs on a single machine
- It can make sense to use Big Data with a single-server distribution model
  - Graph databases
    - The graph is “almost” complete → it is difficult to distribute it

DISTRIBUTION MODELS

SHARDING

- Horizontal scalability → putting different parts of the data onto different servers
- Different people are accessing different parts of the dataset



DISTRIBUTION MODELS

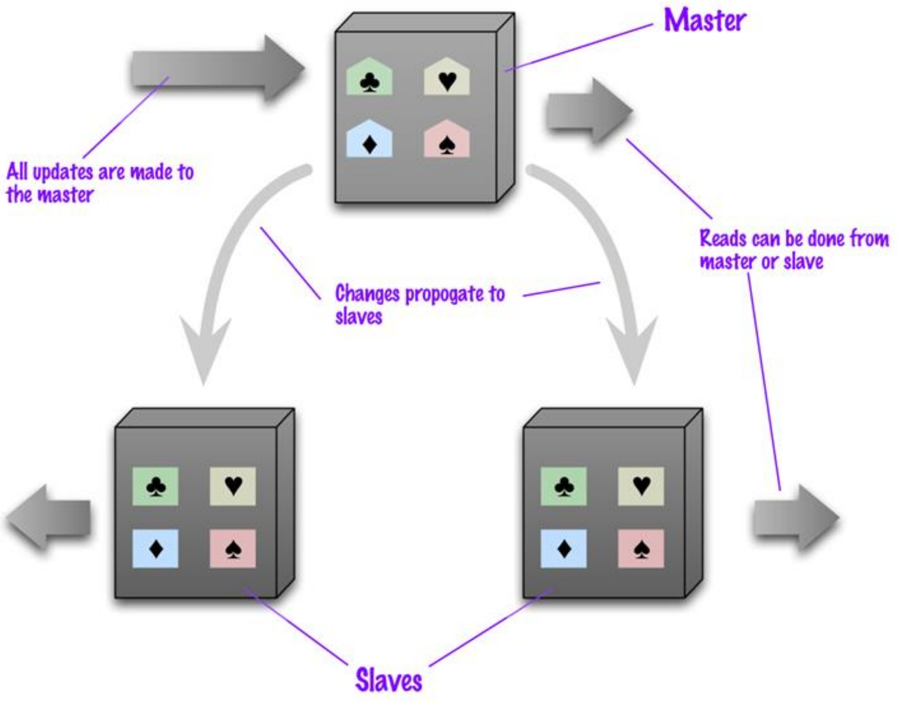
SHARDING – HOW TO?

- The ideal case is rare
- To get close to it, we have to ensure that data that is accessed together are stored together
- How to arrange the nodes:
  a. One user mostly gets data from a single server
  b. Based on a physical location
  c. Distribute across the nodes with equal amounts of the load
- Many distributed databases offer auto-sharding
- A node failure makes the shard's data unavailable
  - Sharding is often combined with replication

DISTRIBUTION MODELS

MASTER-SLAVE REPLICATION

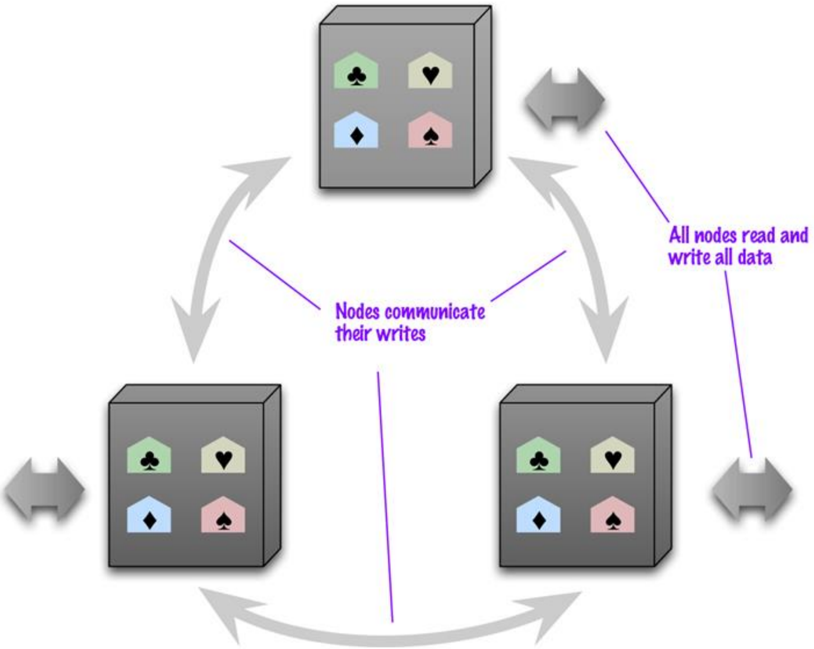
- We replicate data across multiple nodes
- One node is designed as primary (master), others as secondary (slaves)
- Master is responsible for processing any updates to that data



DISTRIBUTION MODELS

PEER-TO-PEER REPLICATION

- Problems of master-slave replication:
  - Does not help with scalability of writes
    - The master is still a bottleneck
  - Provides resilience against failure of a slave, but not of a master
- Peer-to-peer replication: no master
  - All the replicas have equal weight



DATA CENTER AT GOOGLE

“In each cluster's first year, it's typical that 1,000 individual machine failures will occur; thousands of hard drive failures will occur; one power distribution unit will fail, bringing down 500 to 1,000 machines for about 6 hours; 20 racks will fail, each time causing 40 to 80 machines to vanish from the network; 5 racks will "go wonky," with half their network packets missing in action; and the cluster will have to be rewired once, affecting 5 percent of the machines at any given moment over a 2-day span, Dean said. And there's about a 50 percent chance that the cluster will overheat, taking down most of the servers in less than 5 minutes and taking 1 to 2 days to recover.”

https://www.datacenterknowledge.com/archives/2008/05/30/failure-rates-in-google-data-centers

PROBLEM: REALLOCATION

- $h(x) = x \mod 12$
  - The addition or deletion of one machine changes it to x mod 13 or x mod 11
  - Remapping of everything to maintain consistency.
    - Infeasible when n is changing all the time

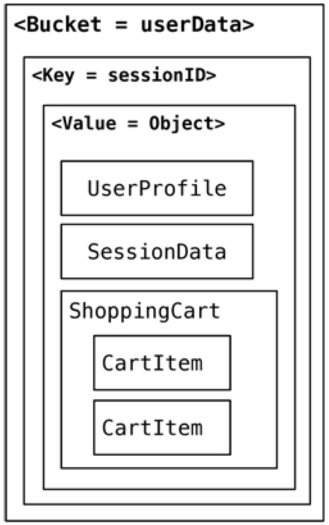
- Solution: Consistent Hashing
  - 1997
  - David Karger et al. (MIT)

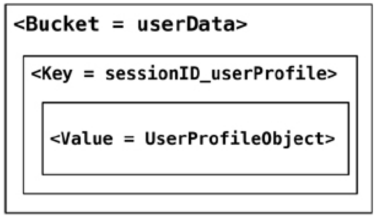
riak

DATA DISTRIBUTION

riak DATABASE

- Key-value database
- A table in RDBMS with two columns, such as ID and NAME
  - ID column being the key
  - NAME column storing the value (a BLOB)
- Basic operations:
  - Get the value for the key
  - Put a value for a key
  - Delete a key from the data store



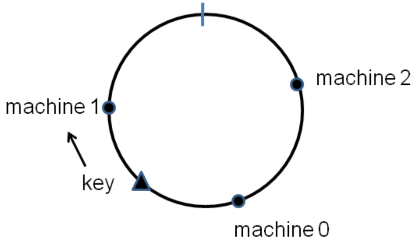


DATA DISTRIBUTION

REPLICATION IN RIAK

- No master node
  - Each node is fully capable of serving any client request
- Uses consistent hashing to distribute data around the cluster
  - Minimizes reshuffling of keys when a hash-table data structure is rebalanced
    - Slots are added/removed
  - Hash function maps the keys to a circle
  - Each node in the cluster is responsible for an interval of hashes (slot) in the circle
  - Only `k/n` keys need to be remapped on average
    - `k` = number of keys
    - `n` = number of intervals (slots)

  instead of almost all in most other hashing types



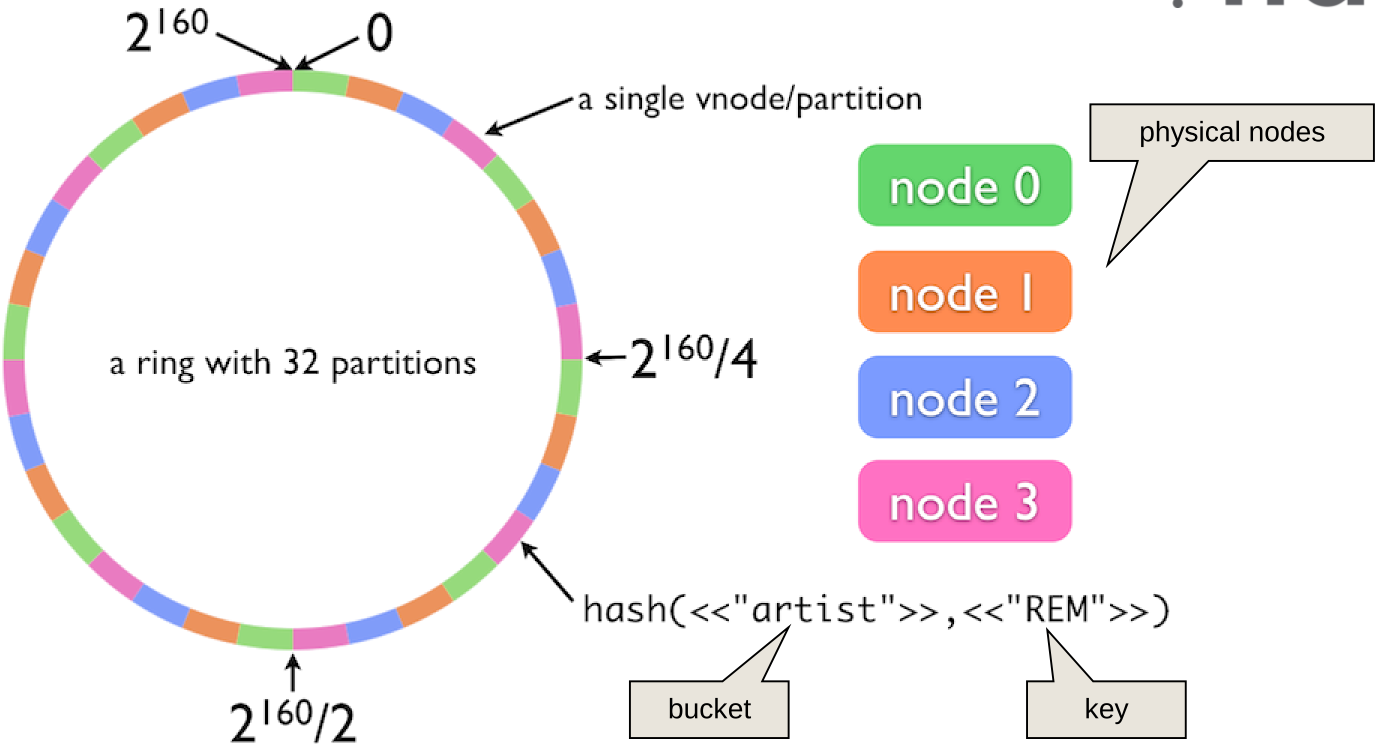
DATA DISTRIBUTION
REPLICATION IN RIAK

- Center of any cluster: 160-bit integer space (Riak ring) which is divided into equally-sized partitions
- Physical nodes run virtual nodes (vnodes)
  - vnode is responsible for storing a separate portion of the keys
    - They solve the problem of changing length of intervals
  - Each physical node in the cluster is responsible for:

    `1/(number of physical nodes)`

    of the ring
  - Number of vnodes on each node:

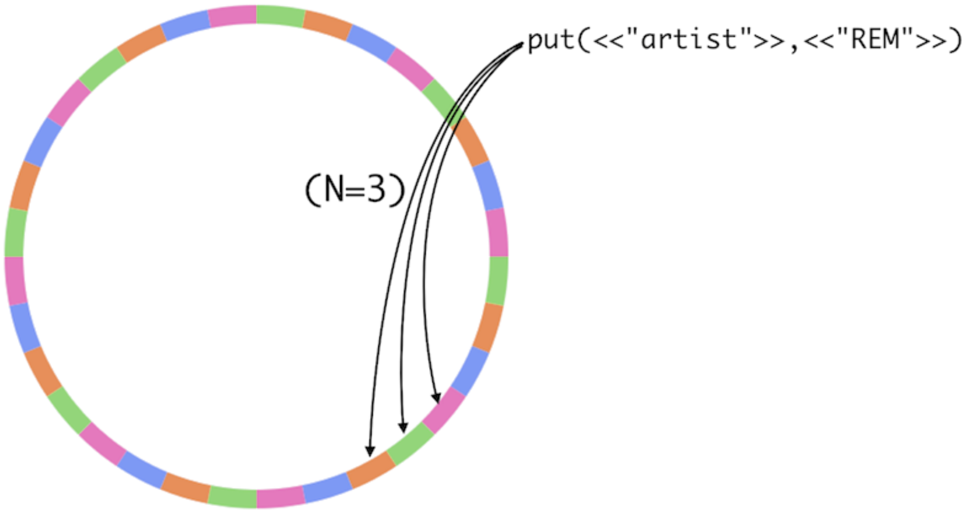
    `(number of partitions)/(number of physical nodes)`
- Nodes can be added and removed from the cluster dynamically
  - Riak will redistribute the data accordingly
- Example:
  - A ring with 32 partitions
  - 4 physical nodes
  - 8 vnodes per node


2^{160}
0
a single vnode/partition
a ring with 32 partitions
$2^{160}/4$
$2^{160}/2$
physical nodes
node 0
node 1
node 2
node 3
hash(<<"artist">>,<<"REM">>)
bucket
key

DATA DISTRIBUTION

REPLICATION IN RIAK

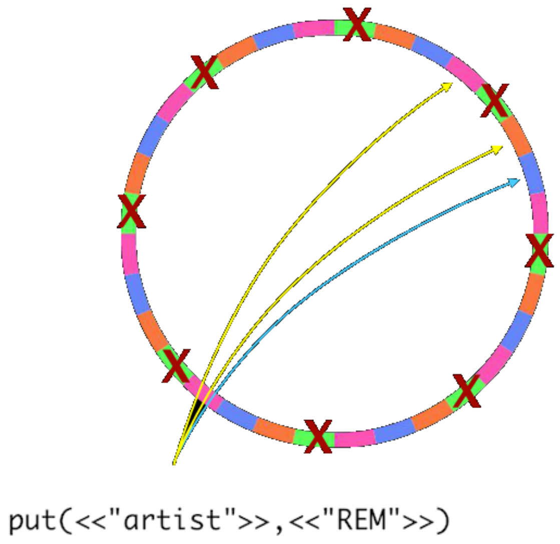
- Setting called N value
  - Default: N=3
- Riak objects inherit the N value from their bucket



DATA DISTRIBUTION

REPLICATION IN RIAK

- Riak's key feature: high availability
- Hinted handoff
  1. Node failure
  2. Neighboring nodes temporarily take over storage operations
  3. When the failed node returns, the updates received by the neighboring nodes are handed off to it



DATA DISTRIBUTION
SHARING INFORMATION IN RIAK

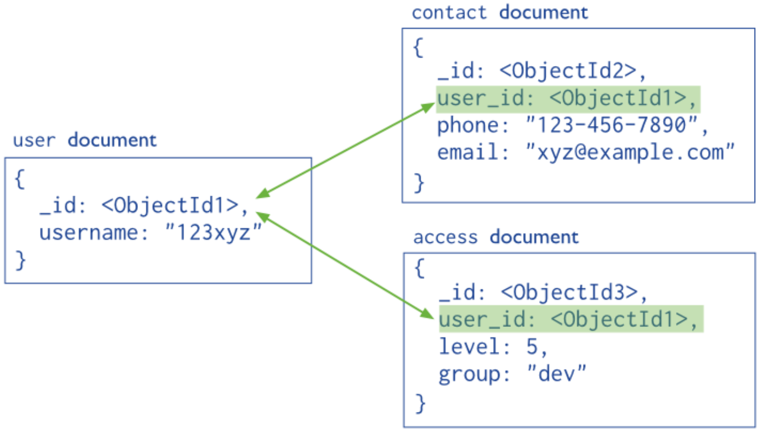
- Gossip protocol
  - Motivation: robust spread of information when people gossip
  - To share and communicate ring state and bucket properties around the cluster
  - Gossiping = sending an information to a randomly selected node
    - According to the acquired information it updates its knowledge about the cluster
  - Each node „gossips":
    - Whenever it changes its claim on the ring
      - Announces its change
    - Periodically sends its current view of the ring state
      - For the case a node missed previous updates

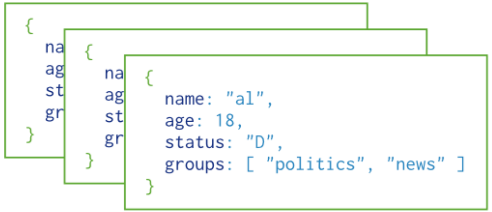
mongoDB®

DATA DISTRIBUTION

mongoDB DATABASE

- Document database
- Use JSON
- Stored as BSON
  - Binary representation of JSON
- Have maximum size: 16MB (in BSON)
  - Not to use too much RAM
  - GridFS tool divides larger files into fragments

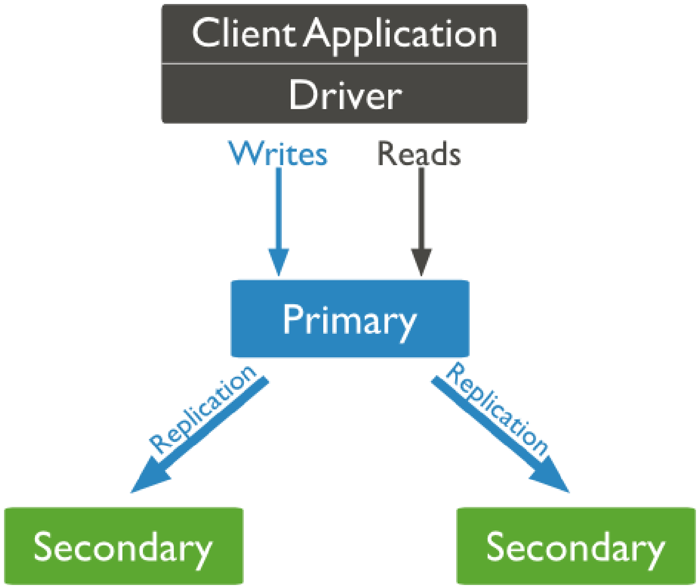




Collection

MONGODB
REPLICATION

- Master/slave replication
- Replica set = group of instances that host the same data set
  - **primary** (master) – receives all write operations
  - **secondaries** (slaves) – apply operations from the primary so that they have the same data set



MONGODB

REPLICATION STEPS

- **Write:**
  1. mongoDB applies write operations on the primary
  2. mongoDB records the operations to the primary's oplog
  3. Secondary members replicate oplog + apply the operations to their data sets

operation log

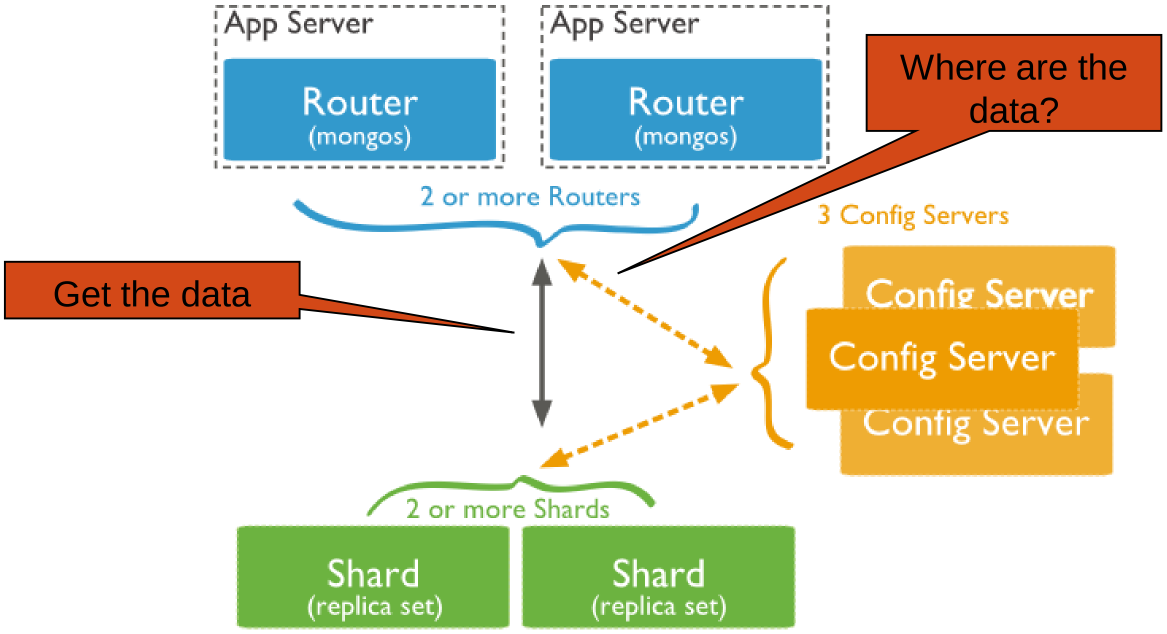
- **Read:** All members of the replica set can accept read operations
  - By default, an application directs its read operations to the primary member
    - Guaranties the latest version of a document
    - Decreases read throughput
  - Read preference mode can be set

MONGODB

SHARDING

- Supported through sharded clusters
- Consisting of:
  - Shards – store the data
    - Each shard is a replica set
      - For testing purposes can be a single node
  - Query routers – interface with client applications
    - Direct operations to the appropriate shard(s) + return the result to the user
    - More than one ⇒ to divide the client request load
  - Config servers – store the cluster's metadata
    - Mapping of the cluster's data set to the shards
    - Recommended number: 3

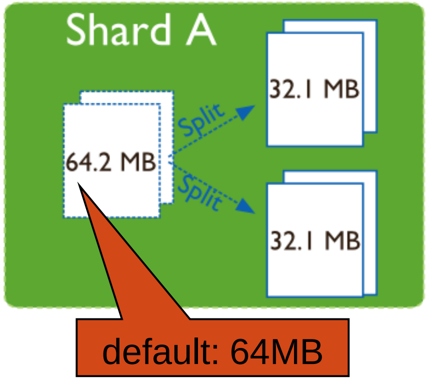
MONGODB
SHARDED CLUSTER



MongoDB

Data Partitioning

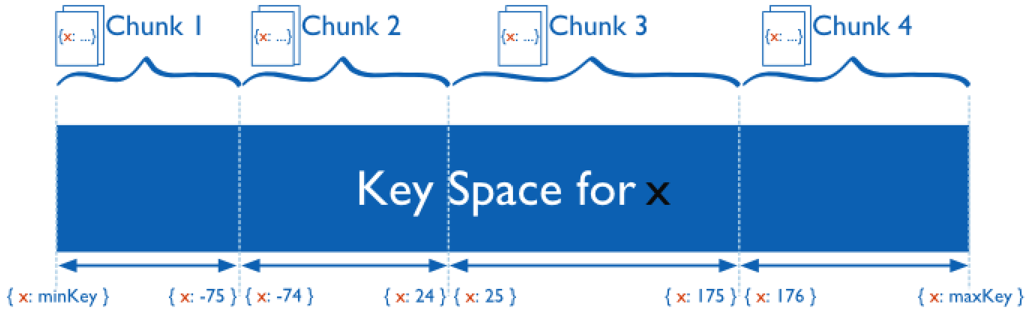
- Partitions a collection's data by the shard key
  - Indexed (possibly compound) field that exists in every document in the collection
    - Immutable
  - Divided into chunks distributed across shards
    - **Range-based partitioning**
    - **Hash-based partitioning**
  - When a chunk grows beyond the chunk size, it is split
    - Small chunks ⇒ more even distribution at the expense of more frequent migrations
    - Large chunks ⇒ fewer migrations



MONGODB

RANGE-BASED PARTITIONING

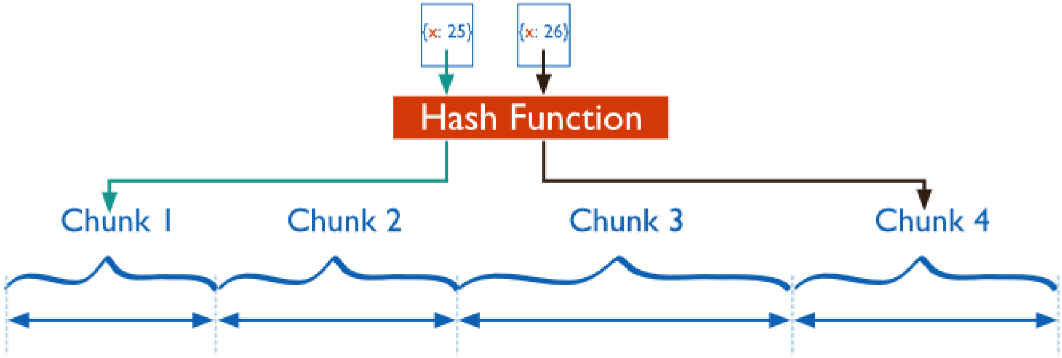
- Each value of the shard key falls at some point on line from negative infinity to positive infinity
- The line is partitioned into non-overlapping chunks
- Documents with "close" shard key values are likely to be in the same chunk
  - More efficient range queries
  - Can result in an uneven distribution of data



MONGODB

HASH-BASED PARTITIONING

- Computes a hash of a field's value
  - Hashes form chunks
- Ensures a more random distribution of a collection in the cluster
  - Documents with "close" shard key values are unlikely to be a part of the same chunk
  - A range query may need to target most/all shards



REFERENCES

- http://nosql-database.org/
- Pramod J. Sadalage – Martin Fowler: NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence
- Eric Redmond – Jim R. Wilson: Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement