Introduction to NoSQL

Lecture Plan

- Introductions
- What is NoSQL?
- Relational vs. NoSQL databases
- Aggregate data model
- Map-Reduce and Hadoop

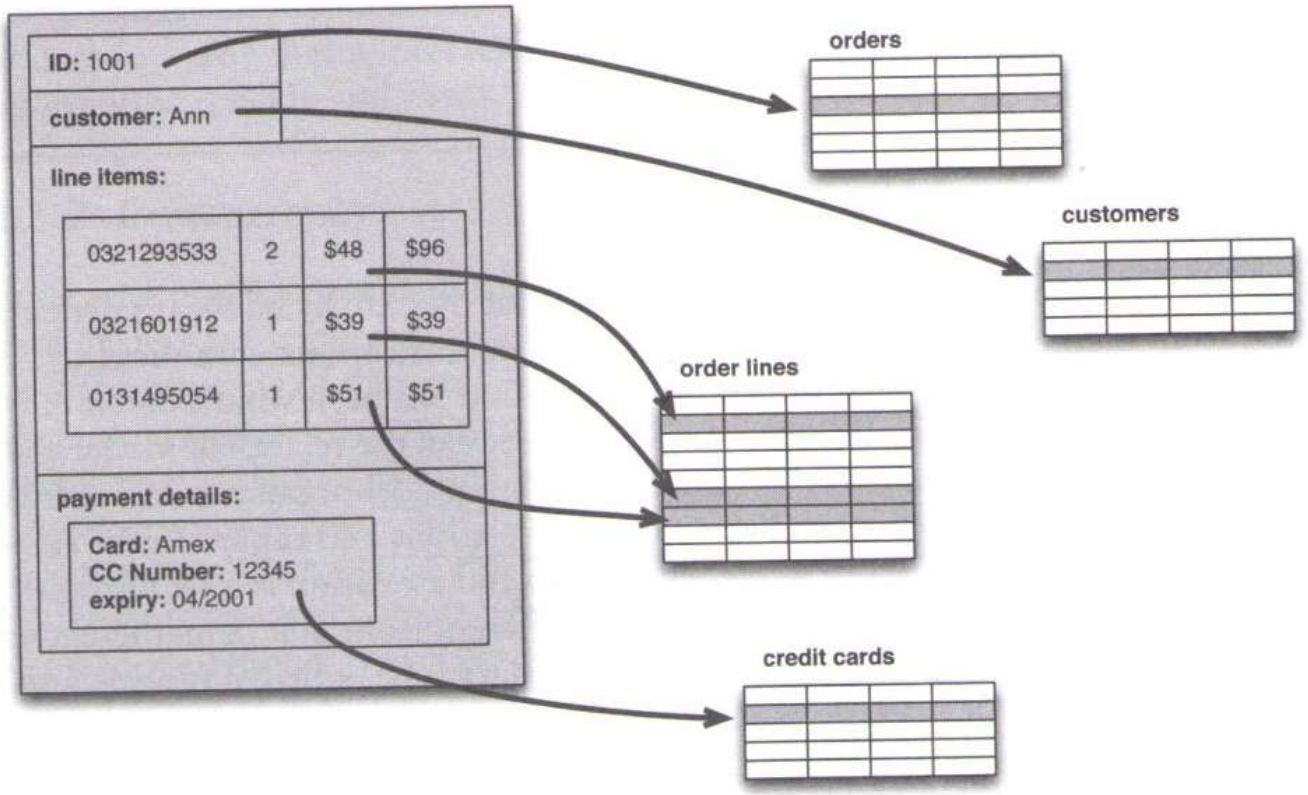
Relational databases: strengths

- **Persistence:** large amounts of data can be safely and securely kept on storage device(s)
  - ability to get small bits of information quickly and easily
- **Concurrency:** many applications may look at the same body of data at once, possibly modifying that data:
  - RDBs handle concurrency by controlling the access to their data through transactions
  - if an error occurs during the processing of changes, transactions can be rolled back
- **Integration:** several applications need to communicate and collaborate to solve a complex task:
  - concurrency control automatically handles multiple applications

Relational databases: weaknesses

- **Impedance mismatch:** difference between the relational model and in-memory data structures
  - RDBs organize data into structure of relations and tuples (tables and rows)
  - values in a relational tuple have to be simple (i.e. no structures, such as nested records or lists)
  - in-memory data structures can be more complex than simple relations
  - as a result, in-memory data structures need to be translated into a relational representation in order to be stored on disk

Relational data model



Relational databases: major weakness

- RDBs are designed to be run on a single machine
- **Sharding:** RDBs could be run as separate servers for different sets of data
  - sharding is controlled by an application, which keeps track of which RDB server to talk to for each bit of data
  - …but querying, referential integrity, transactions and consistency control across shards still need to be implemented

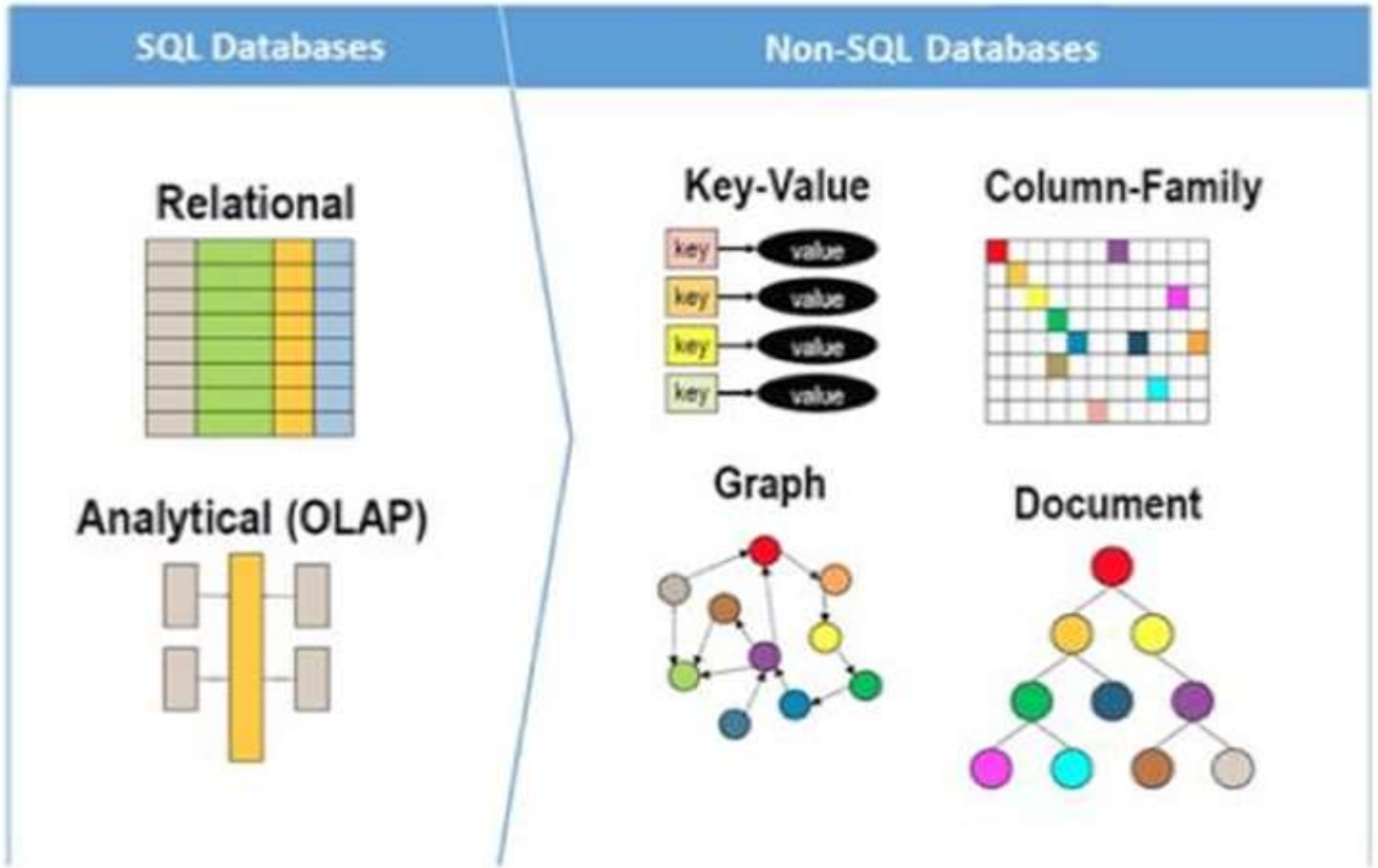
Why NoSQL?

- Relational DBMSs have been a successful technology for more than twenty years, since they provided reliable persistence, concurrency control and integration mechanisms
- RDBs are designed to run on a single machine and do not scale up horizontally
- However, the need to process large volumes of data led to a shift from scaling vertically to scaling horizontally on clusters
- **Cluster:** large number of commodity machines connected with a network

History of NoSQL

- Early efforts were focused on proprietary systems by Amazon and Google in 2000s:
  - BigTable from Google
  - Dynamo from Amazon
- The term “NoSQL” traces back to a meetup on June 11, 2009 in San Francisco, after which NoSQL DBMs have become an open-source phenomenon

NoSQL: “Not Only SQL”



Relational DBs

Name	Birthday	PERSONID
Jos The Boss	11-12-1985	1
Fritz von Braun	27-1-1978	2
Freddy Stark		3
Delphine Thewiseone	16-9-1986	4

Person info table: represents person specific information

PERSONID	HOBBYID
1	2
1	2
2	3
2	4
3	5
3	6
3	1

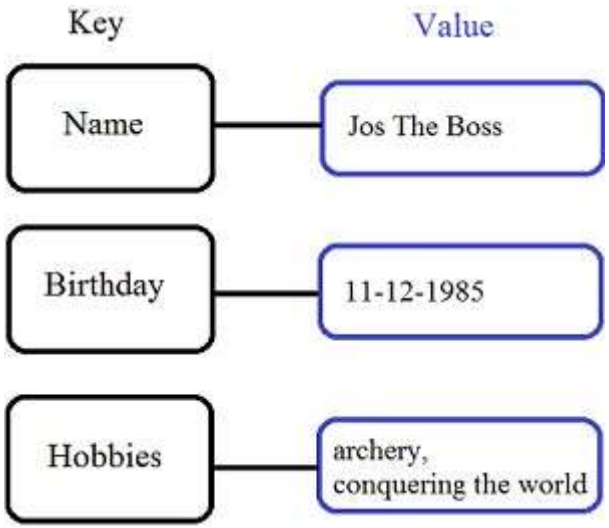
Person-Hobby linking table. This is necessary because of the many to many relationship between hobbies and persons.

HOBBYID	HobbyName	HobbyDescription
1	Archery	Shooting arrows from a bow
2	Conquering the world	looking for trouble with your neighboring countries
3	building things	also known as construction
4	surfing	catching waves on a plank
5	swordplay	fencing with swords
6	lollygagging	hanging around doing nothing

Hobby info table: represents hobby specific information

Relational databases strive towards **normalization** (making sure every piece of data is stored just once). Each table has unique identifiers (primary keys) that are used to model the relation between the entities (tables) hence "relational".

KEY-VALUE STORES



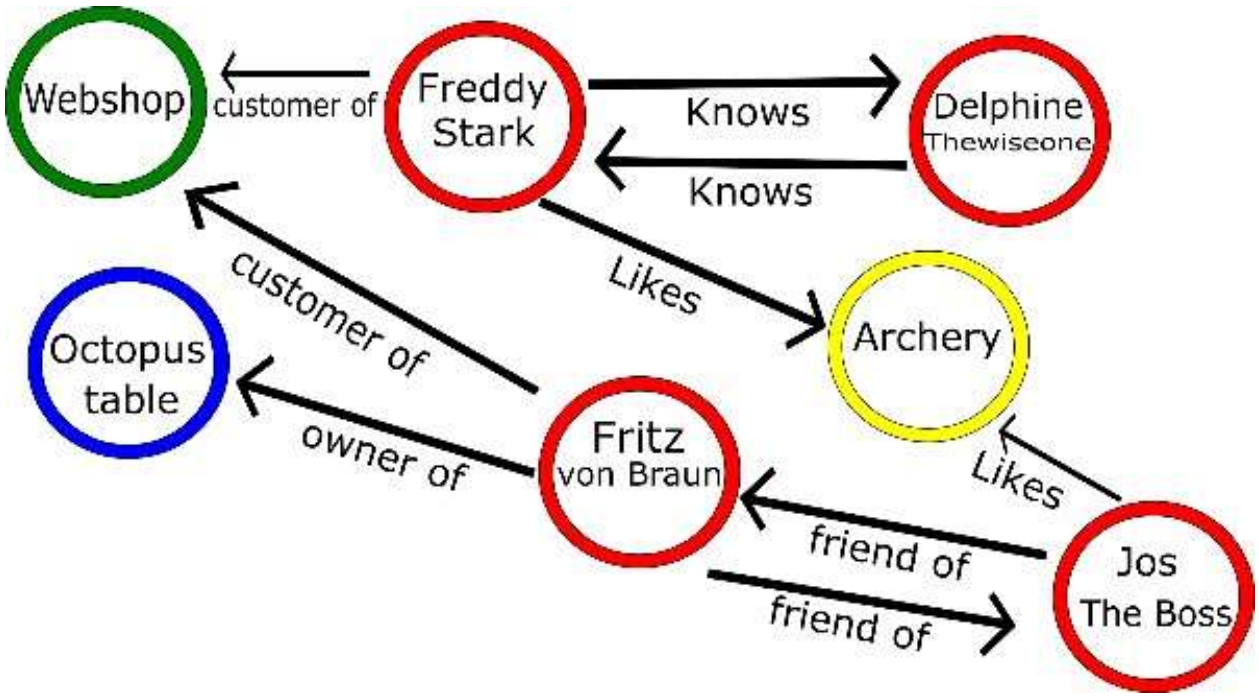

```
{"internal data":[{"entities":[
    {"customer":[
        {"id":1,"name":"Freddy"},
        {"id":2,"name":"Fritz"}
    ]},
    {"legal entities":[
        {"id":1,"company":"Maiton"}
    ]}]
},{"Products":[
    {"furniture":[
        {"id":1,"name":"Octopus Table","stock":1}
    ]
}]}]}
```

Document Stores

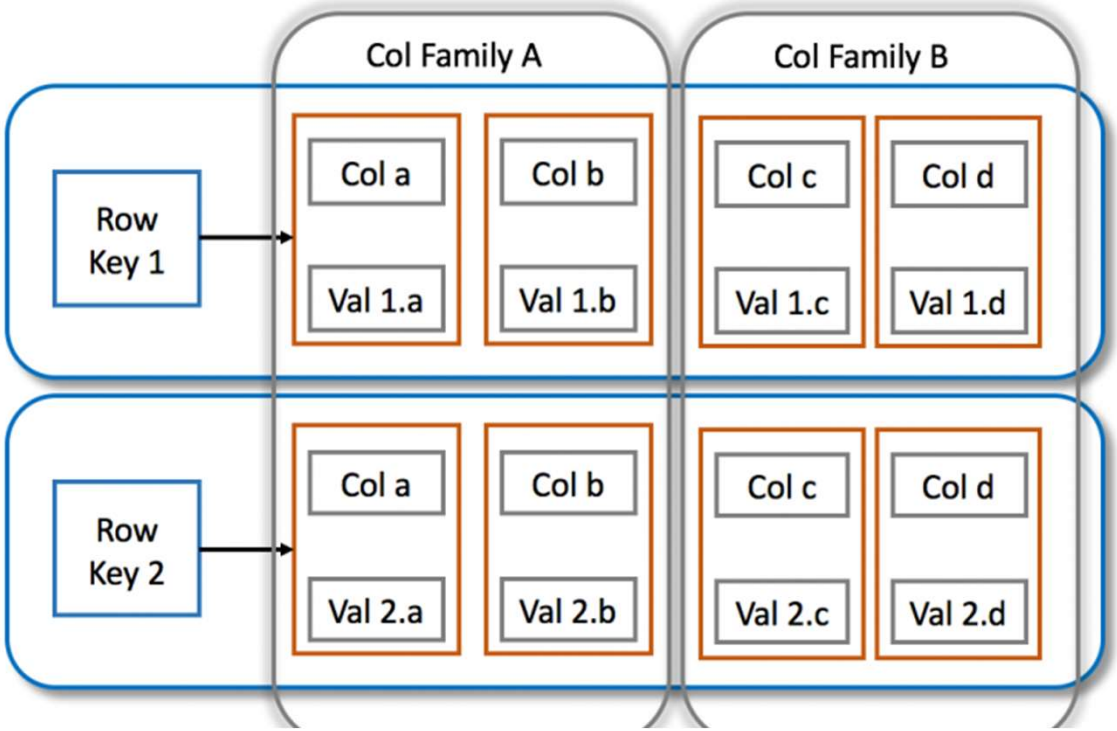

```
{
  "articles": [
    {
      "title": "title of the article",
      "articleID": 1,
      "body": "body of the article",
      "author": "Isaac Asimov",
      "comments": [
        {
            "username": "Fritz",
            "join date": "1/4/2014",
            "commentid": 1,
            "body": "this is a great article",
            "replies": [
                {
                    "username": "Freddy",
                    "join date": "11/12/2013",
                    "commentid": 2,
                    "body": "seriously? it's rubbish"
                }
            ]
        },
        {
            "username": "Stark",
            "join date": "19/06/2011",
            "commentid": 3,
            "body": "I don't agree with the conclusion"
        }
      ]
    }
  ]
}
```

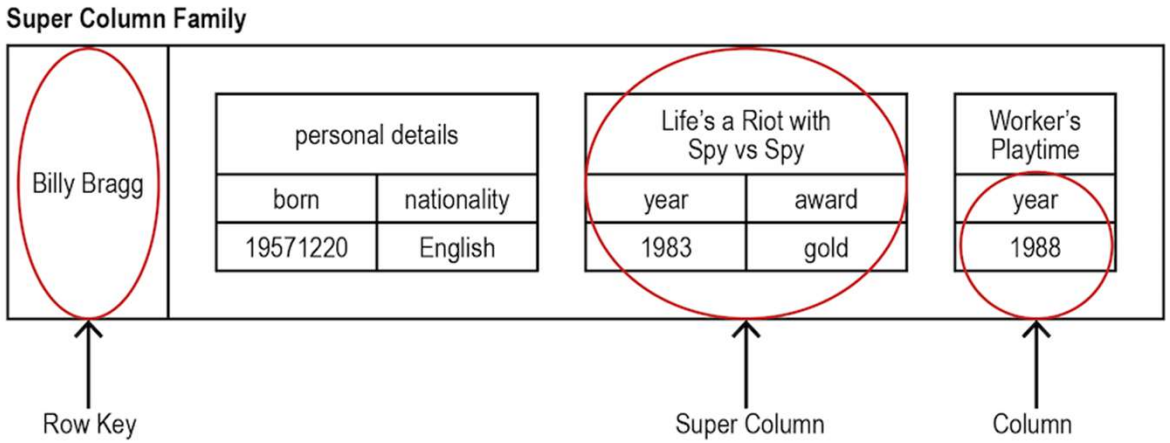
Whereas relational databases chop up data, Document stores save documents as a single entity

Graph Databases



Wide Column Database





Types of NoSQL databases

- **Key-value:** BerkeleyDB, LevelDB, Memcached, Project Voldemort, Redis, Riak
- **Document:** OrientDB, RavenDB, Terrastore, CouchDB, MongoDB
- **Column-family:** Amazon SimpleDB, Cassandra, Hypertable, HBase
- **Graph:** FlockDB, HyperGraphDB, Infinite Graph, Neo4J

DB-Engines Ranking

https://db-engines.com/en/ranking

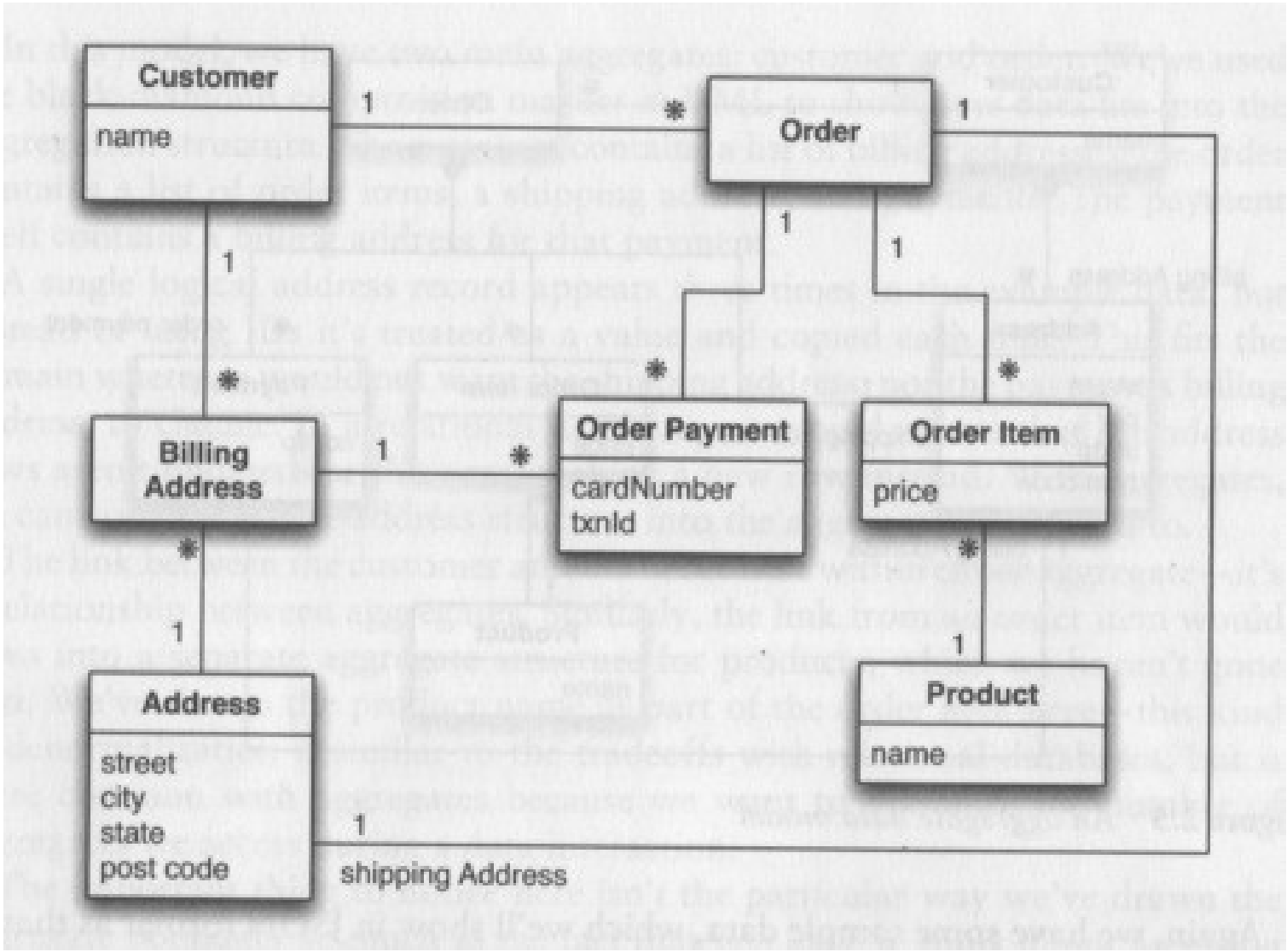
NoSQL: aggregate data model

- Explicit storage of a rich structure of closely related data that is accessed as a unit (called **aggregates)**
- Aggregates provide a natural unit of interaction for many applications
- Suitable for distributed environment
- **Downside:** difficulty in handling relationships between entities in different aggregates

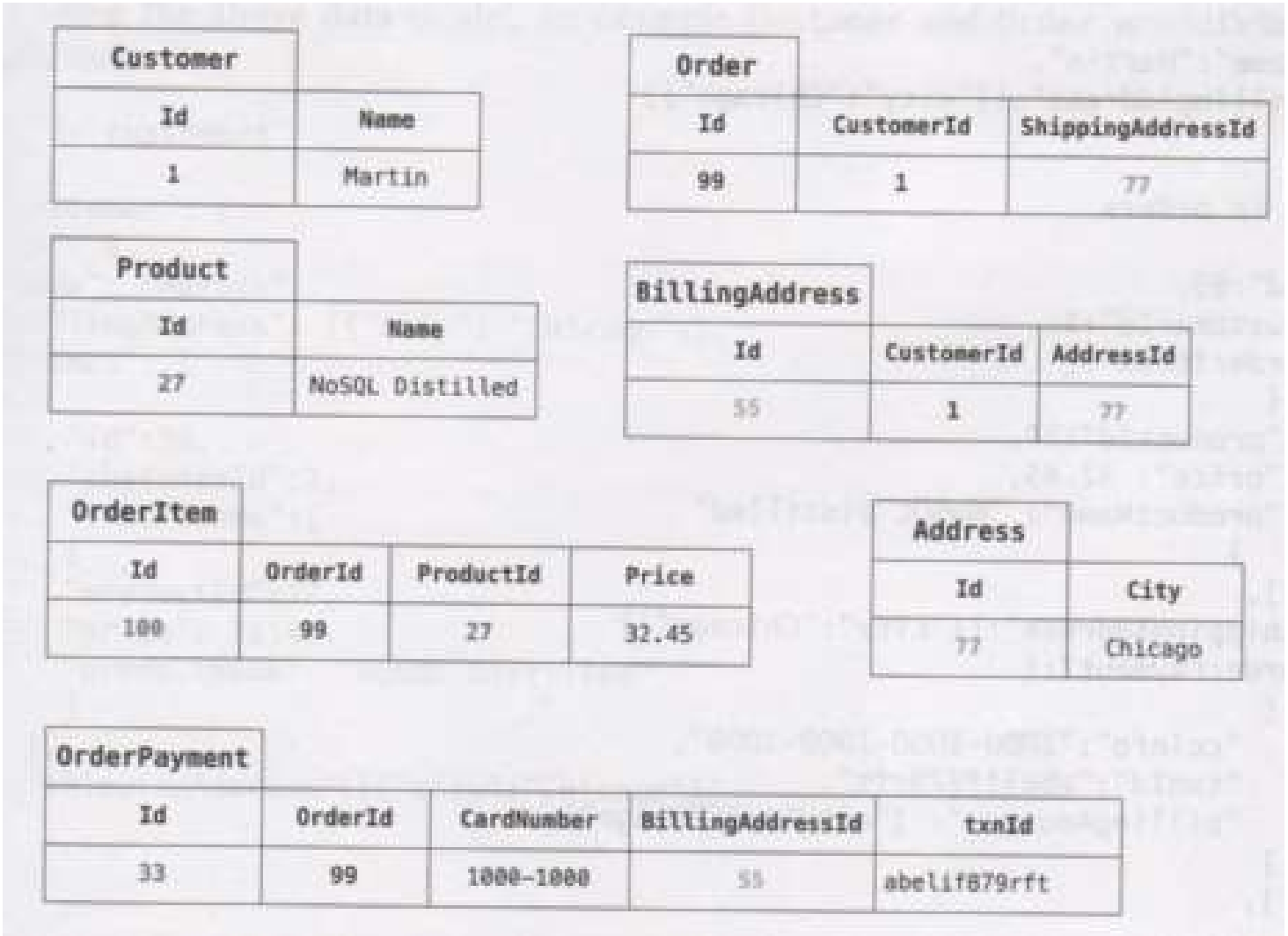
Aggregate

- Complex record allowing lists and other record structures to be nested inside it
- Collection of related objects that are treated as a unit

Relational schema



Relational data model



Customer

Id	Name
1	Martin

Order

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

Example of aggregates

```
// in customers
{
"id":1,
"name":"Martin",
"billingAddress":[{"city":"Chicago"}]
}

// in 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"}
  }
  ],
}
```

Aggregate vs. relational data model

- No normalization:
  - instead of using IDs, some records may be duplicated and copied with an aggregate
  - minimize the number of aggregates we access during data interaction
  - minimizing the number of nodes to query for data and data transfer overhead when gathering the data
- Relations between aggregates are still possible:
  - e.g., between orders and customers
  - aggregate boundaries are context-specific (i.e. depend on the task and how the data is manipulated by the application)
- Relational databases are aggregate-ignorant:
  - and so are NoSQL graph databases

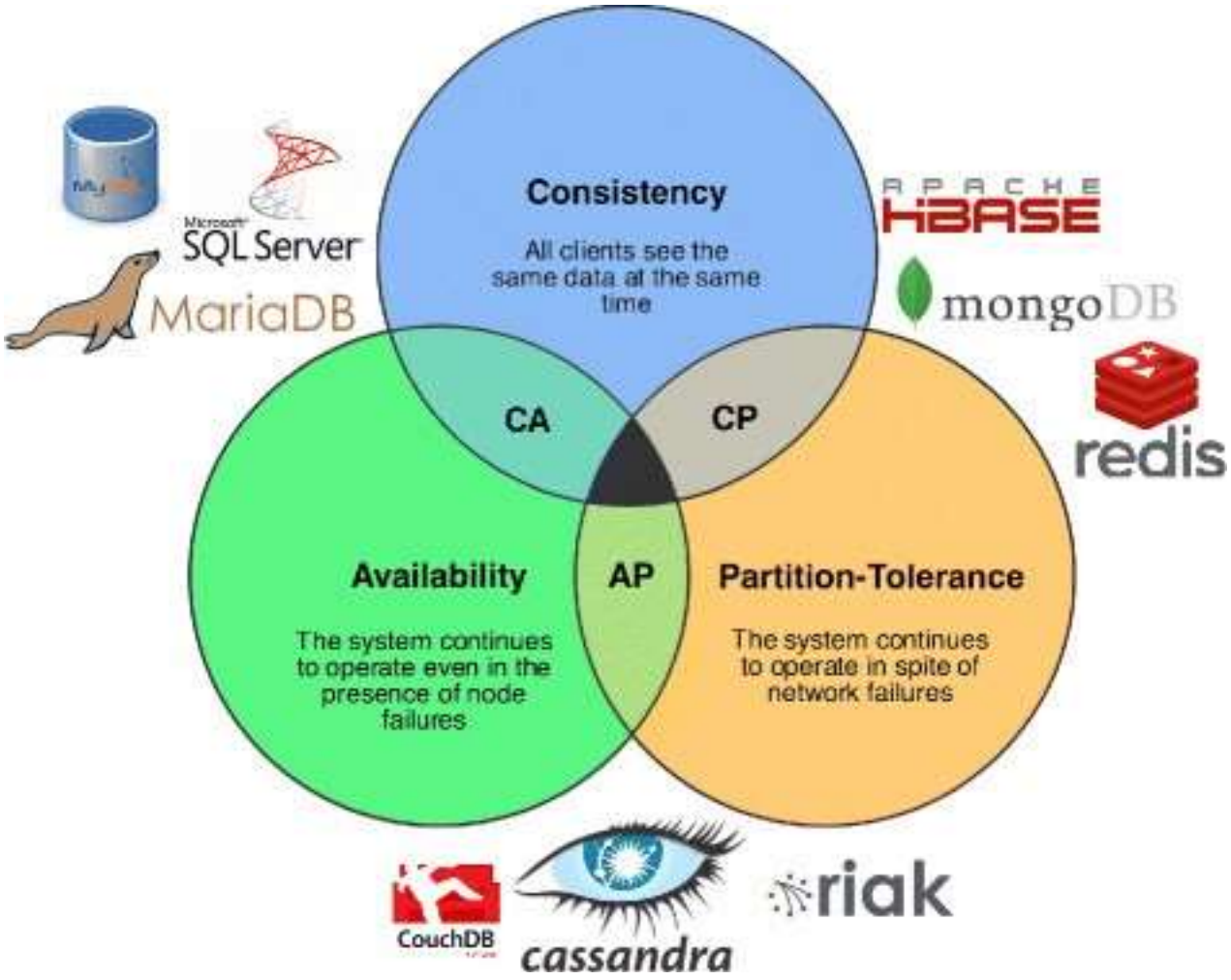
Relational vs. NoSQL DBs: atomicity

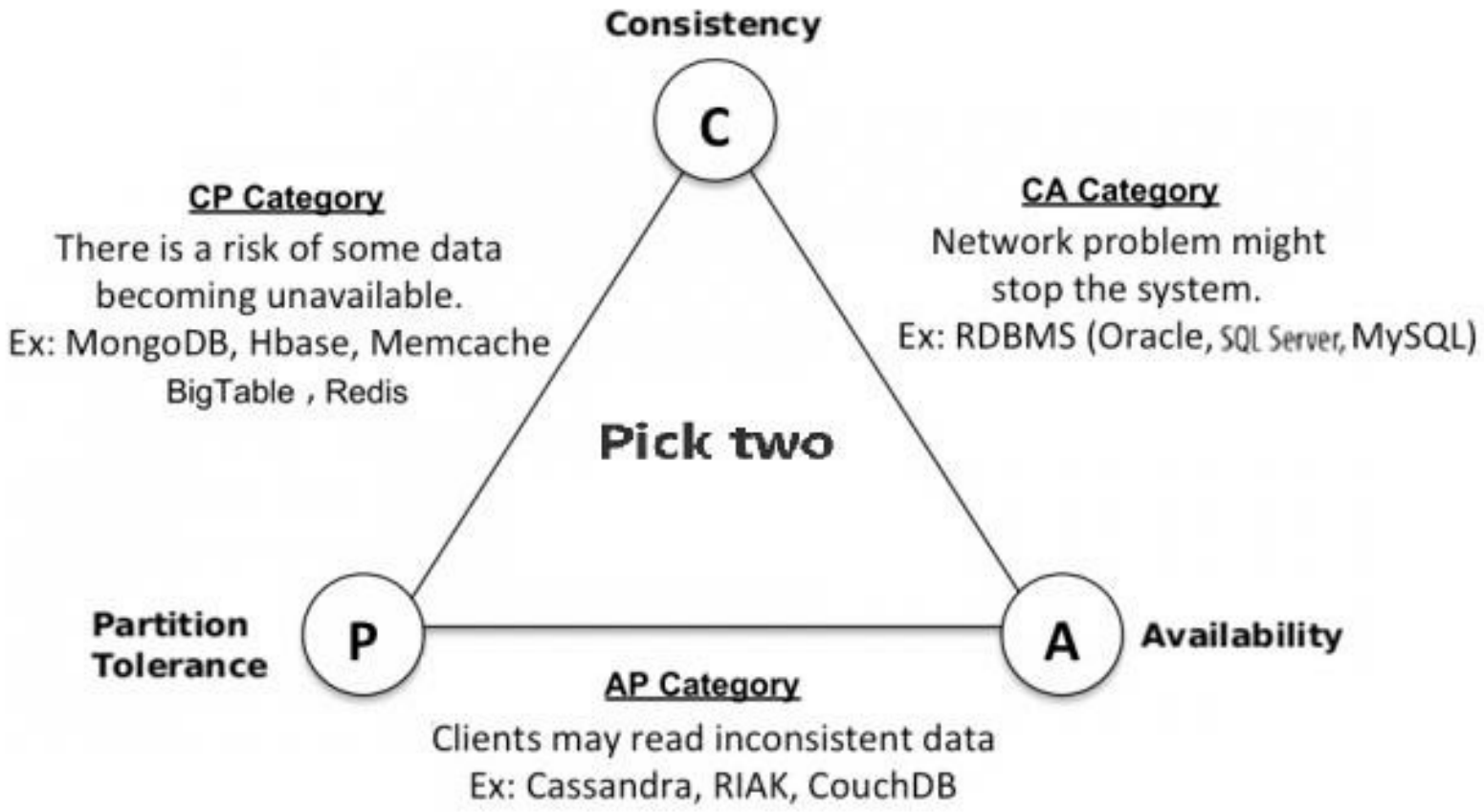
- RDBs allow to manipulate any combination of rows from any tables in a single **ACID** (**A**tomic, **C**onsistent, **I**solated and **D**urable) transaction:
  - many rows spanning many tables are updated as a single atomic operation
  - atomic operations succeed or fail entirely
- NoSQL databases support atomic manipulation of single aggregate at a time:
  - cross-aggregate atomic operations need to be implemented programmatically
- Aggregate-ignorant NoSQL DBs support ACID transactions similar to relational DBs

CAP theorem

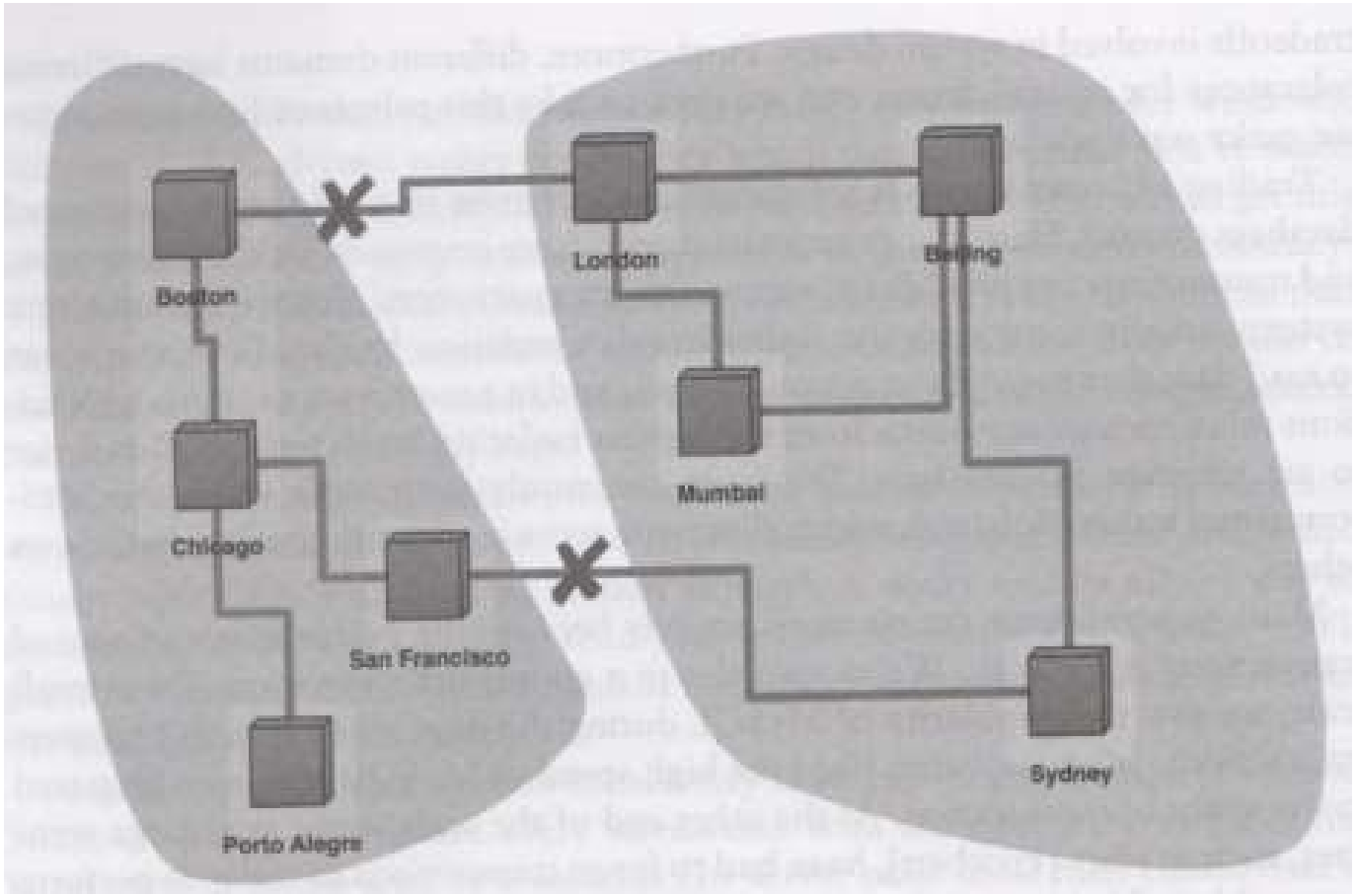
The CAP theorem

- Many database systems forgo transactions entirely, because the performance impact is too high
- MySQL was popular since it was lightweight and didn’t support transactions
- Consistency can and should often be relaxed


Consistency
All clients see the same data at the same time
Availability
The system continues to operate even in the presence of node failures
Partition-Tolerance
The system continues to operate in spite of network failures
CA
CP
AP
SQL Server
MariaDB
APACHE HBASE
mongoDB
redis
CouchDB
cassandra
riak


Consistency
C
CP Category
There is a risk of some data becoming unavailable.
Ex: MongoDB, Hbase, Memcache
BigTable , Redis
CA Category
Network problem might stop the system.
Ex: RDBMS (Oracle, SQL Server, MySQL)
Pick two
Partition Tolerance
P
A
Availability
AP Category
Clients may read inconsistent data
Ex: Cassandra, RIAK, CouchDB

The CAP theorem



Choose DBs

https://www.dataversity.net/choose-right-nosql-database-application/#

Map-Reduce and Hadoop

What is Hadoop?

- A software framework that supports data-intensive distributed applications.
- It enables applications to work with **thousands of nodes** and **petabytes of data**.
- Hadoop was inspired by Google's MapReduce and Google File System (GFS).
- Hadoop is a top-level Apache project being built and used by a global community of contributors, using the Java programming language.
- Yahoo! has been the largest contributor to the project, and uses Hadoop extensively across its businesses.

Who uses Hadoop?

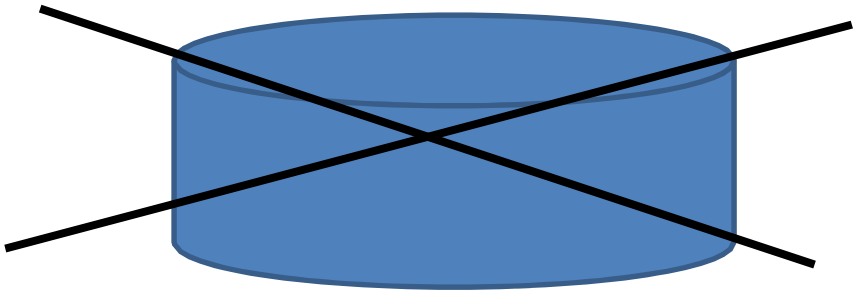
http://wiki.apache.org/hadoop/PoweredBy

Who uses Hadoop?

- Yahoo!
  - More than 100,000 CPUs in >36,000 computers.
- Facebook
  - Used in reporting/analytics and machine learning and also as storage engine for logs.
  - A 1100-machine cluster with 8800 cores and about 12 PB raw storage.
  - A 300-machine cluster with 2400 cores and about 3 PB raw storage.
  - Each (commodity) node has 8 cores and 12 TB of storage.

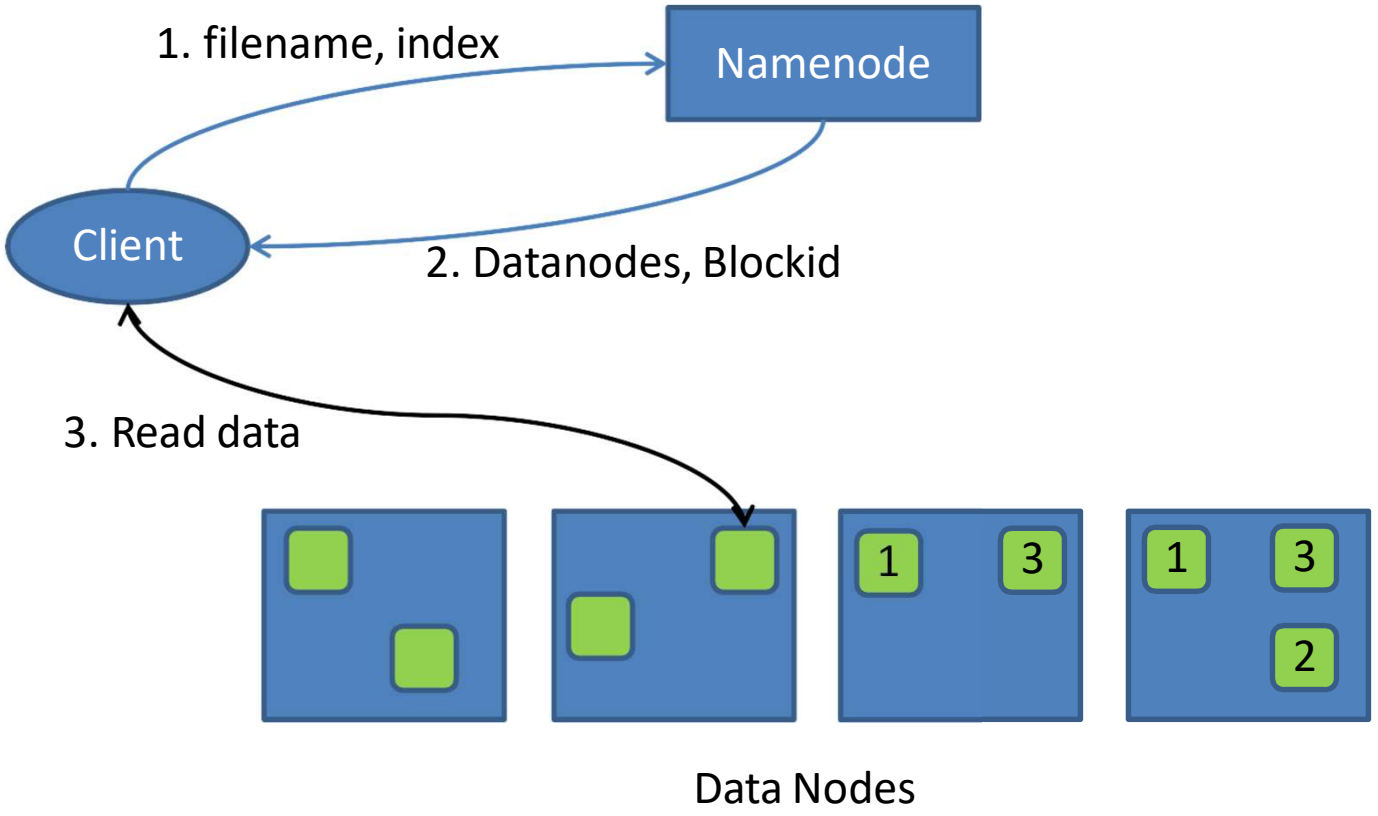
Very Large Storage Requirements

- Facebook has Hadoop clusters with 15 PB of raw storage (15,000,000 GB).
- No single storage can handle this amount of data.



- We need a large set of nodes each storing part of the data.

HDFS: Hadoop Distributed File System



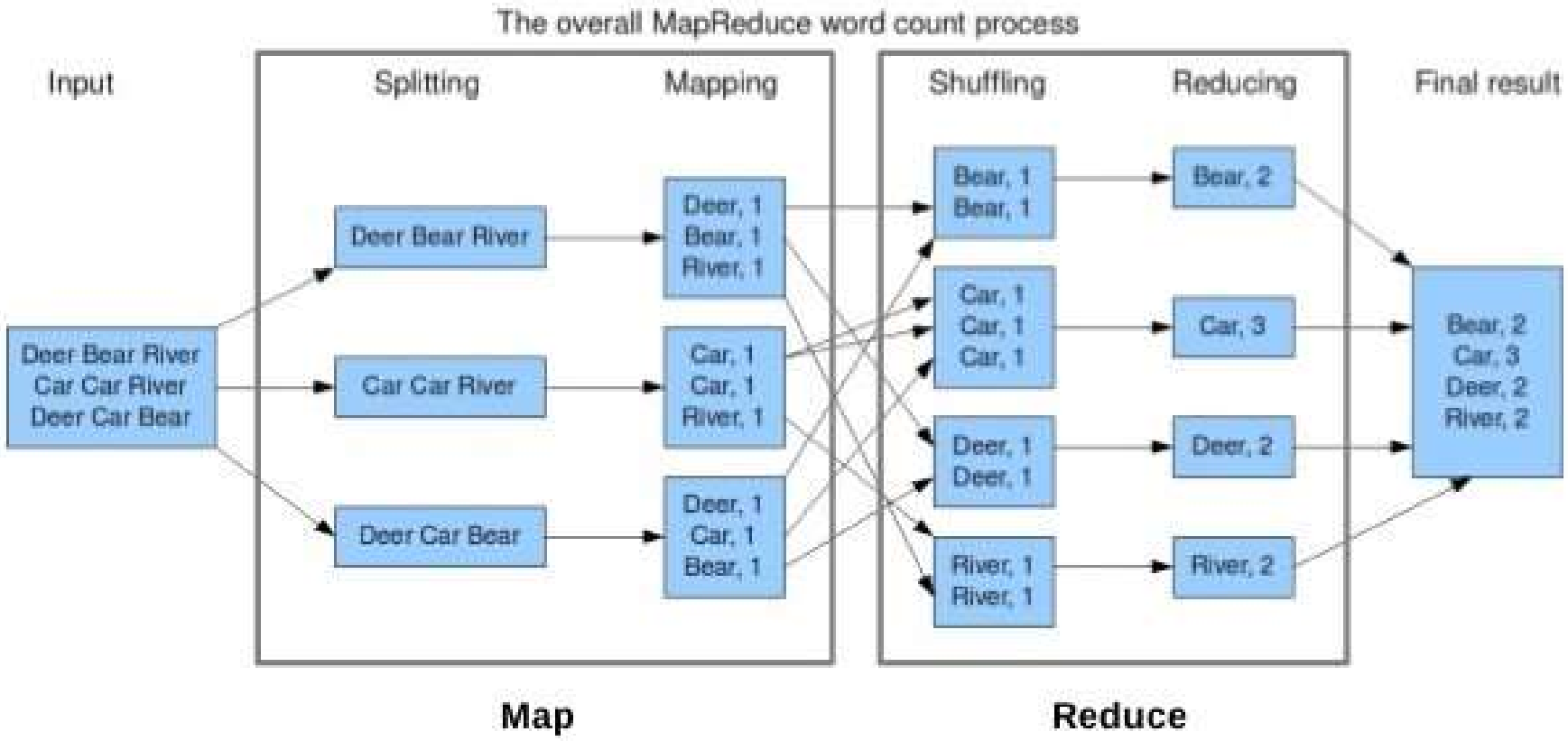
Terabyte Sort Benchmark

- http://sortbenchmark.org/
- Task: Sorting 100TB of data and writing results on disk (10^12 records each 100 bytes).

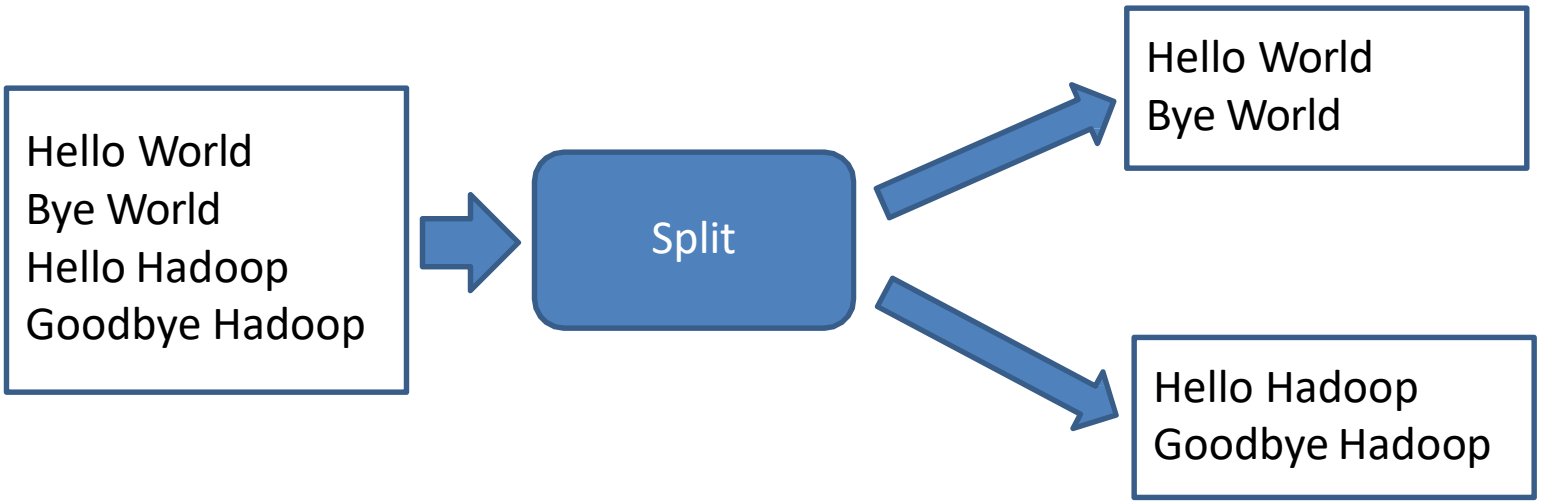
- Yahoo's Hadoop Cluster is the current winner:
  - **173 minutes**
  - 3452 nodes x (2 Quadcore Xeons, 8 GB RAM)

This is the first time that a **Java** program has won this competition.

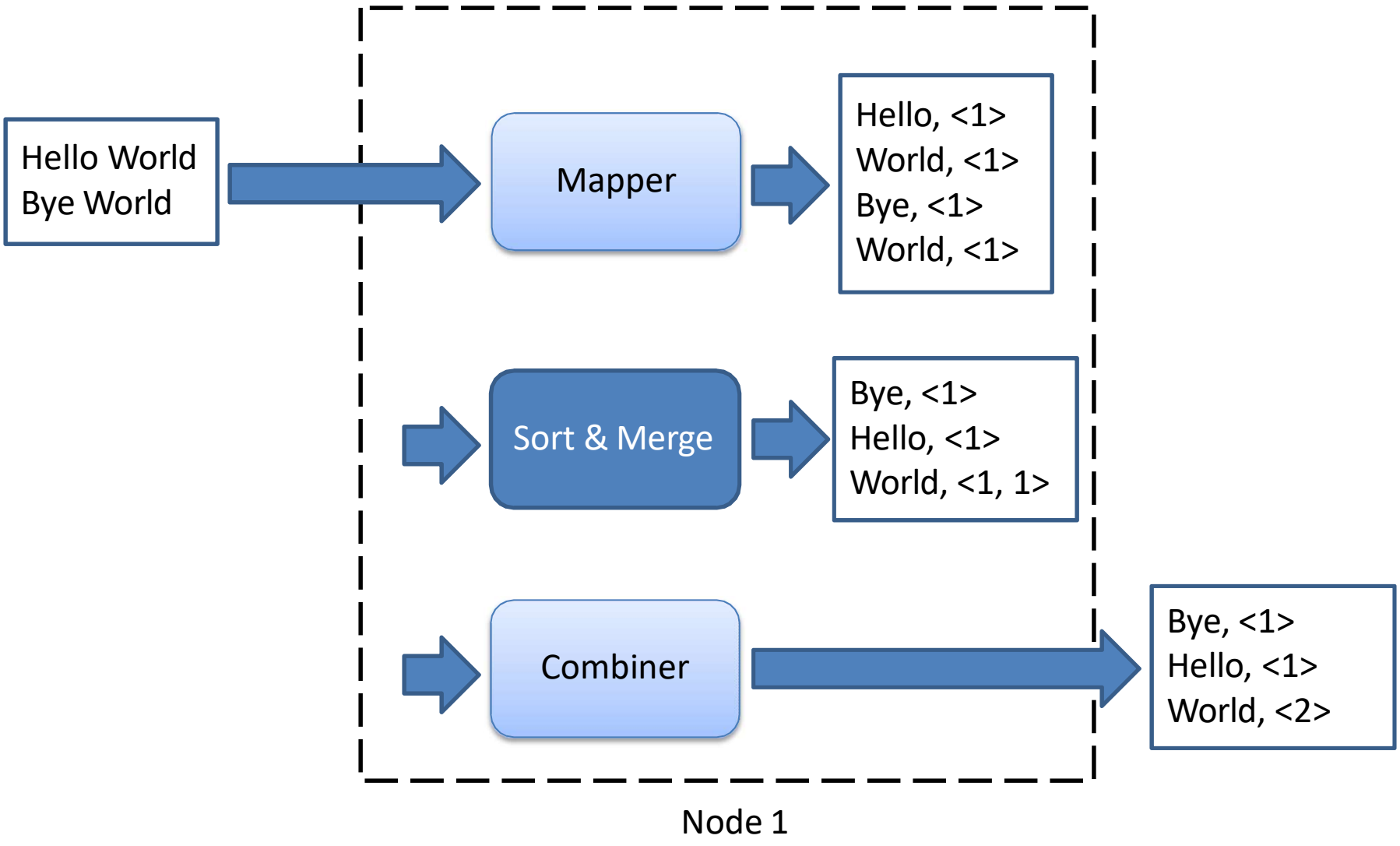
Example: word count



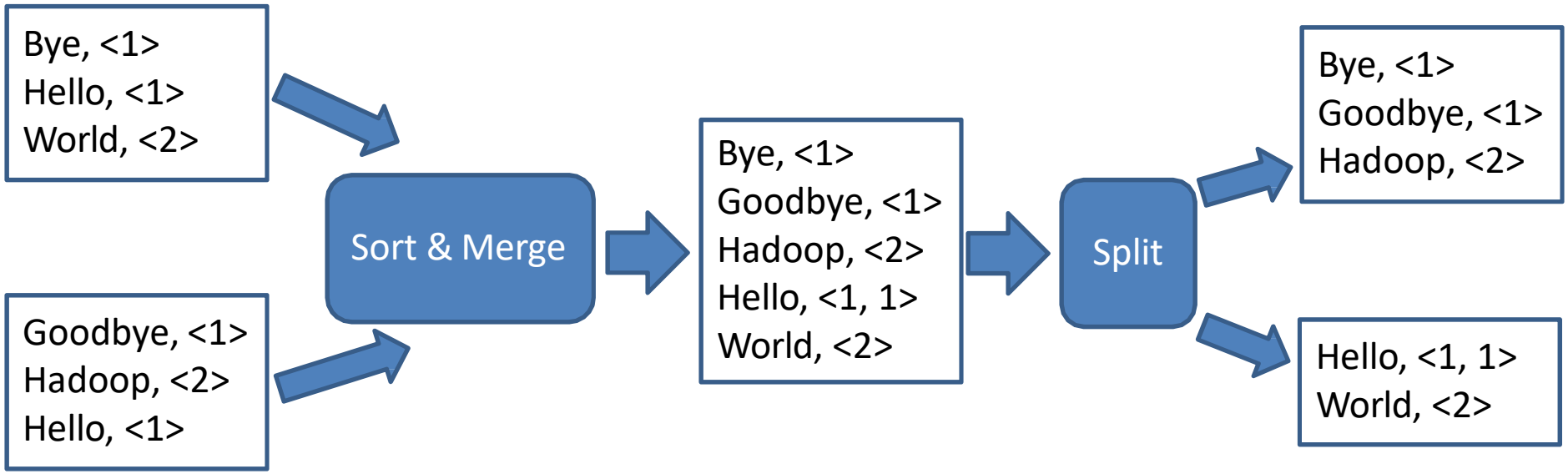
Counting Words by MapReduce



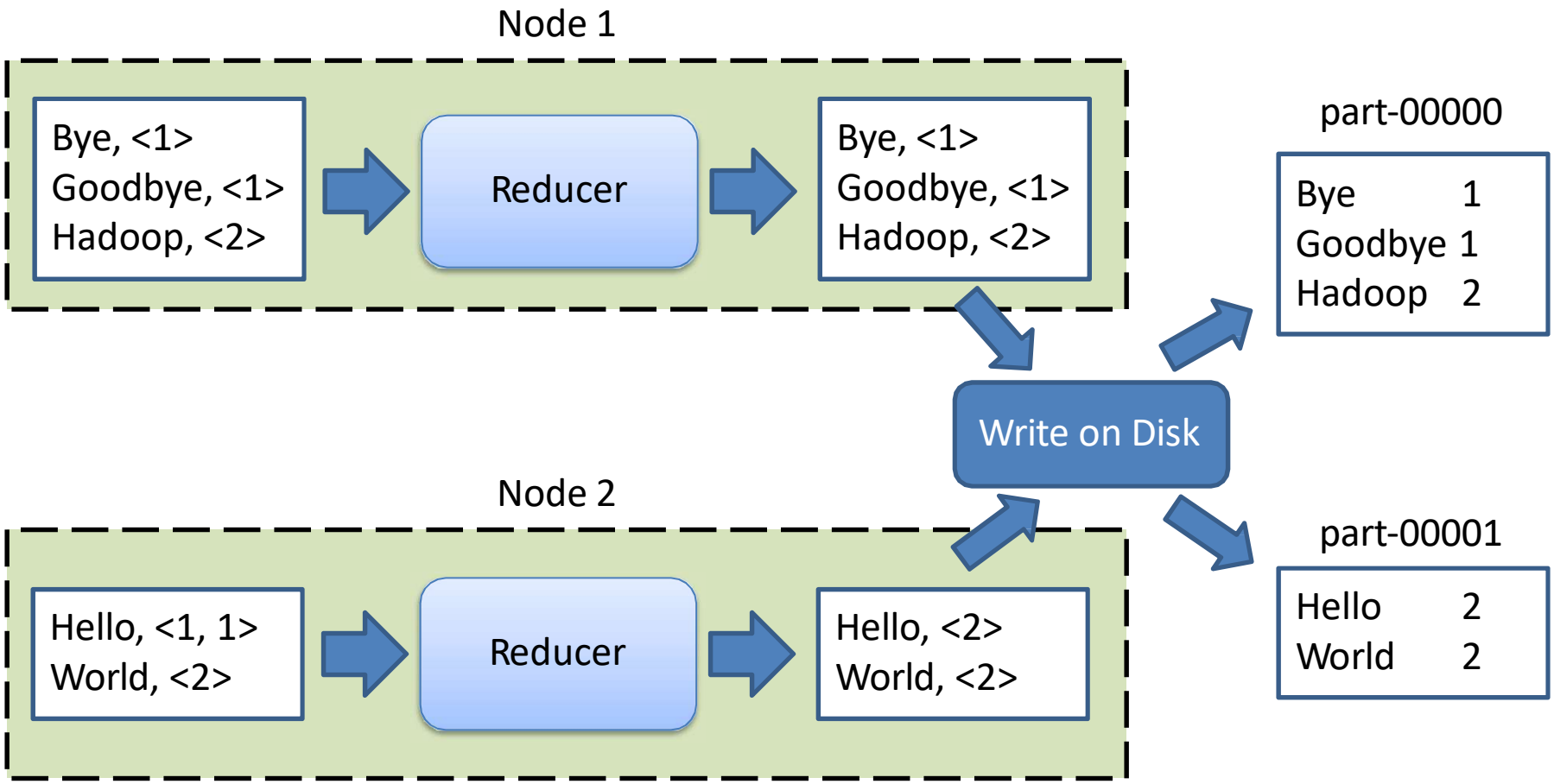
Counting Words by MapReduce



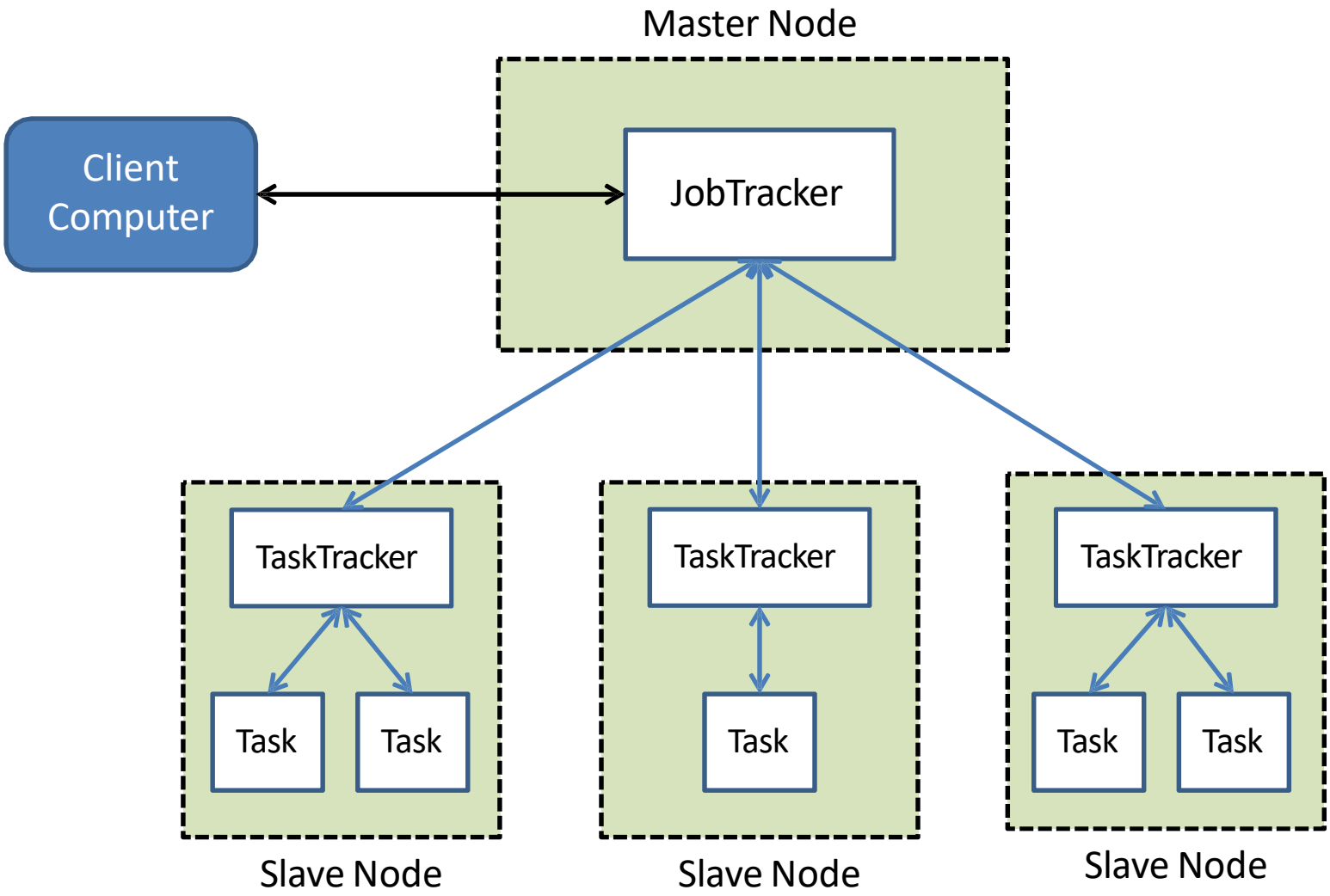
Counting Words by MapReduce



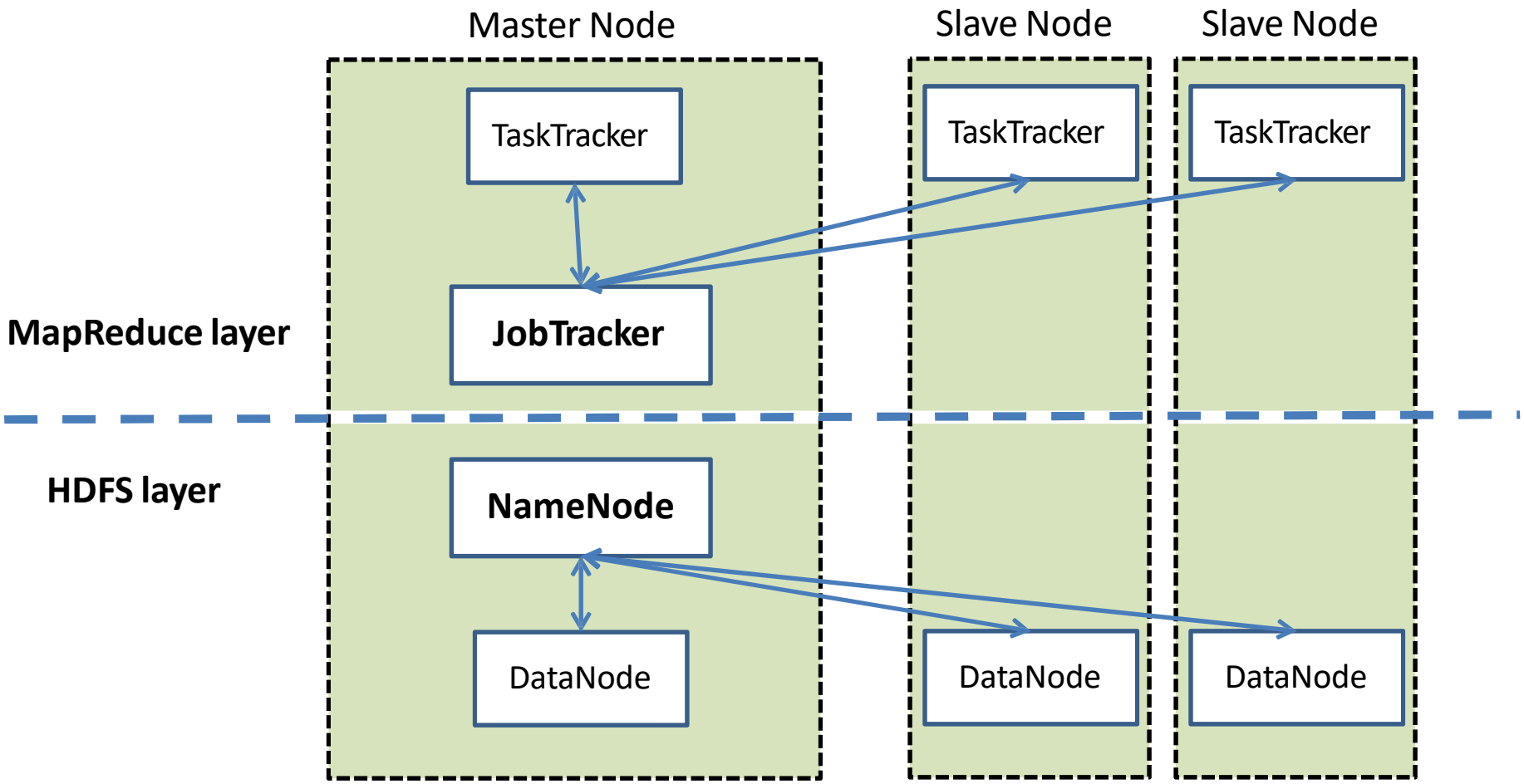
Counting Words by MapReduce



High Level Architecture of MapReduce



High Level Architecture of Hadoop



Hadoop Job Scheduling

- FIFO queue matches incoming jobs to available nodes
  - No notion of fairness
  - Never switches out running job

Distributed File Cache

- The Distributed Cache facility allows you to transfer files from the distributed file system to the local file system (for reading only) of all participating nodes before the beginning of a job.

References

- Hadoop Project Page: http://hadoop.apache.org/