732A54 / TDDE31
Big Data Analytics

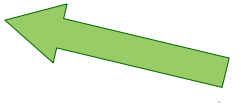
Topic: Distributed DB Management for Big Data

Olaf Hartig
olaf.hartig@liu.se

LINKÖPING UNIVERSITY

Typical* Characteristics of NoSQL Systems

- Ability to scale horizontally over many commodity servers with high performance, availability, and fault tolerance
  - achieved by giving up ACID guarantees
  - and by partitioning and replication of data



- Non-relational data model, no requirements for schemas
  - data model limitations make partitioning effective

*Attention, there is a *broad variety* of such systems and not all of them have these characteristics to the same degree

Partitioning of the Database

Fragmentation/Sharding and Allocation

Group Activity

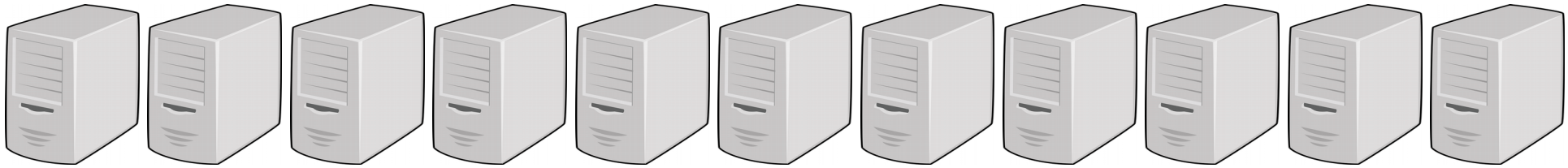
```
login : "alice12"
name : "Alice"
year :  2012
website : "http://alice.name/"
address : {
        street : "Main St"
        city : "Springfield"
 }
```

- Assume we have a *document store* set up on a cluster with *10 compute nodes*
- Assume our document DB consists of many documents where each of them has a field *year* with some year as the value
- Assume we use year as our partitioning field (“shard key”); i.e., one shard per possible year value
- To allocate these shards to our 10 compute nodes we use *hash partitioning*

```
year : 2003
name : "Alice"
website : "http://alice.name/"
favorites : [ "bob_in_se", "charlie" ]
```

```
login : "bob_in_se"
name : "Bob"
twitter : "@TheBob"
year : 1997
```

```
login : "charlie"
name : "Charlie"
year : 2009
```



Group Activity

```
login : "alice12"
name : "Alice"
year :  2012
website : "http://alice.name/"
address : {
        street : "Main St"
        city : "Springfield"
 }
```

- Assume we have a *document store* set up on a cluster with *10 compute nodes*
- Assume our document DB consists of many documents where each of them has a field *year* with some year as the value
- Assume we use year as our partitioning field (“shard key”); i.e., one shard per possible year value
- To allocate these shards to our 10 compute nodes we use *hash partitioning*
- Task: Name a straightforward example for a *hash function* that we may use that has a good chance of distributing the shards evenly across the 10 compute nodes

```
year : 2003
name : "Alice"
website : "http://alice.name/"
favorites : [ "bob_in_se", "charlie" ]
```

```
login : "bob_in_se"
name : "Bob"
twitter : "@TheBob"
year : 1997
```

```
login : "charlie"
name : "Charlie"
year : 2009
```

Quiz

- Given this hash function, assume now that we want to retrieve all documents for which their *year* value is one of the 10 years within in a particular decade
  - for instance, all documents with a year value ≥ 1980 and < 1990
- How many of the 10 nodes may have to be accessed when executing this query?

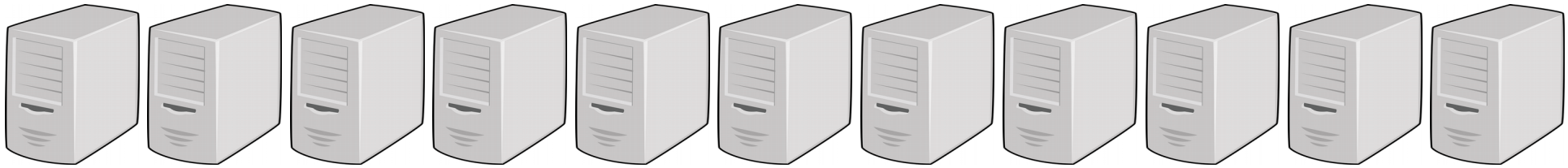
1, 2, 3, 4, 5, 6, 7, 8, 9, or 10 ?

```
login : "alice12"
name : "Alice"
year :  2012
website : "http://alice.name/"
address : {
        street : "Main St"
        city : "Springfield"
 }
```

```
year : 2003
name : "Alice"
website : "http://alice.name/"
favorites : [ "bob_in_se", "charlie" ]
```

```
login : "bob_in_se"
name : "Bob"
twitter : "@TheBob"
year : 1997
```

```
login : "charlie"
name : "Charlie"
year : 2009
```



Replication of Database Objects

Architectures

Quiz

Which of the following statements *is correct*?

1) In master-slave replication, one compute node is selected as the master node for all the database objects (e.g., all key-value pairs in a key-value database) and the other compute nodes become slave nodes.
2) In master-slave replication, a replica (copy) of a database object at a slave node never changes.
3) In master-slave replication, the replica of a database object on the master node can only be written, not read.
4) There can be systems with master-slave replication in which we may see different values for a database object depending on which slave node we read from.

Quiz

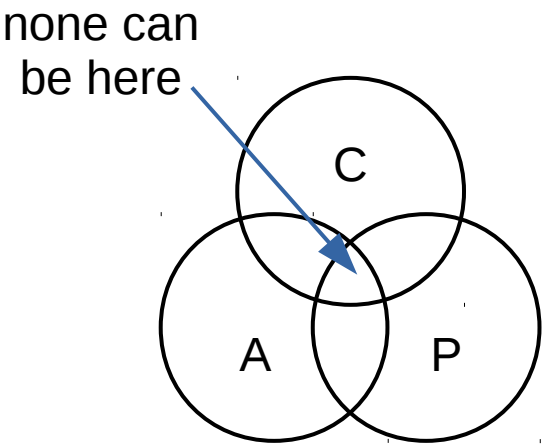
Which of the following statements *is correct*?

1) In multi-master replication, for any given database object, all compute nodes are masters for a replica of this object.
2) In multi-master replication, we can read a database object only from the node at which this object was last updated.
3) When using multi-master replication it is impossible to add new compute nodes at runtime into the system.
4) It is possible to achieve strong consistency in a system with multi-master replication.

CAP Theorem

CAP Theorem

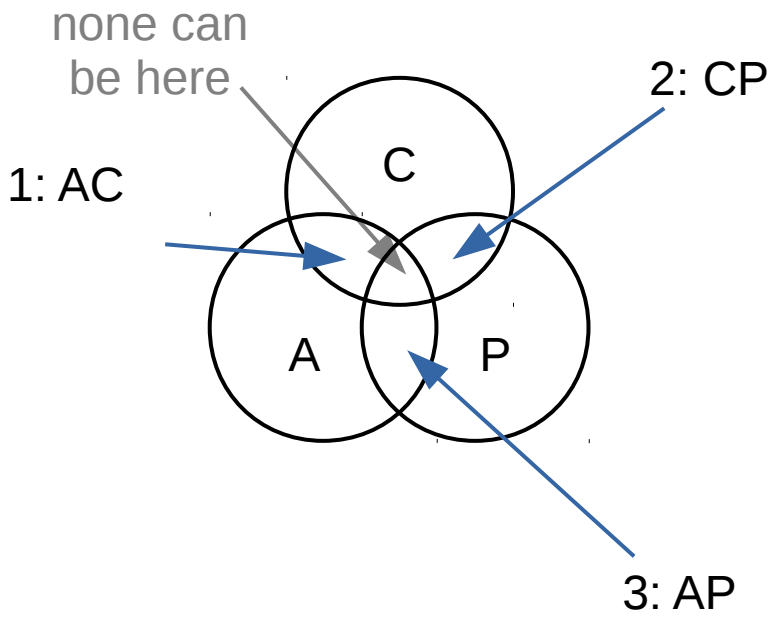
- Only 2 of 3 properties can be guaranteed at the same time in a distributed system with data replication



- ***C***onsistency: the same copy of a replicated data item is visible from all nodes that have this item
  - Note that this is something else than consistency in ACID
- ***A***vailability: all requests for a data item will be answered
  - Answer may be that operation cannot be completed
- ***P***artition Tolerance: system continues to operate even if it gets partitioned into isolated sets of nodes

Quiz

Which two of these three properties do traditional RDBMSs have?



BASE rather than ACID

What is BASE?

- Idea: by giving up ACID guarantees, one can achieve much higher performance and scalability

- ***B***asically ***A***vailable
  - system available whenever accessed, even if parts of it unavailable
- ***S***oft state
  - the distributed data does not need to be in a consistent state at all times
- ***E***ventually consistent
  - state will become consistent after a certain period of time
- BASE properties suitable for applications for which some inconsistency may be acceptable

Levels of Consistency

Quiz

- *Strong consistency*: after an update completes, every subsequent access will return the updated value
- *Weak consistency*: no guarantee that all subsequent accesses will return the updated value

Quiz

- *Strong consistency*: after an update completes, every subsequent access will return the updated value

- Assume a (multi-master) system in which each database object is replicated at 5 nodes
- We want to allow the system to require a quorum of only 2 nodes when we read a database object
- On the other hand, we also want strong consistency
- Then, when writing, how many nodes have to confirm a write of a database object for the write to be considered successful?

A) at least 2, B) at least 3, C) at least 4, D) all 5

Quiz

- *Strong consistency*: after an update completes, every subsequent access will return the updated value

- Assume a (multi-master) system in which each database object is replicated at 5 nodes
- We want to allow the system to require a quorum of only 2 nodes when we read a database object
- Assume we don't care about consistency for reads, but we want strong consistency for writes ("write consistency")
- Then, how many nodes have to confirm a write in this case for the write to be considered successful?

A) at least 2, B) at least 3, C) at least 4, D) all 5

www.liu.se

LINKÖPING UNIVERSITY