Algoteq Brown Bag series

Pilot Episode - Apache Ignite

Ignite - what’s that?

- A distributed database - can hold data on multiple physical machines
- Key-value pairs storage (Redis style), with optional SQL access
- Distributed transactions supported
- In-memory-centric, with optional persistence
- Written in Java

So… what?

So it’s just a Java-written in-memory database… right?

Y waste our time then?!?!!!

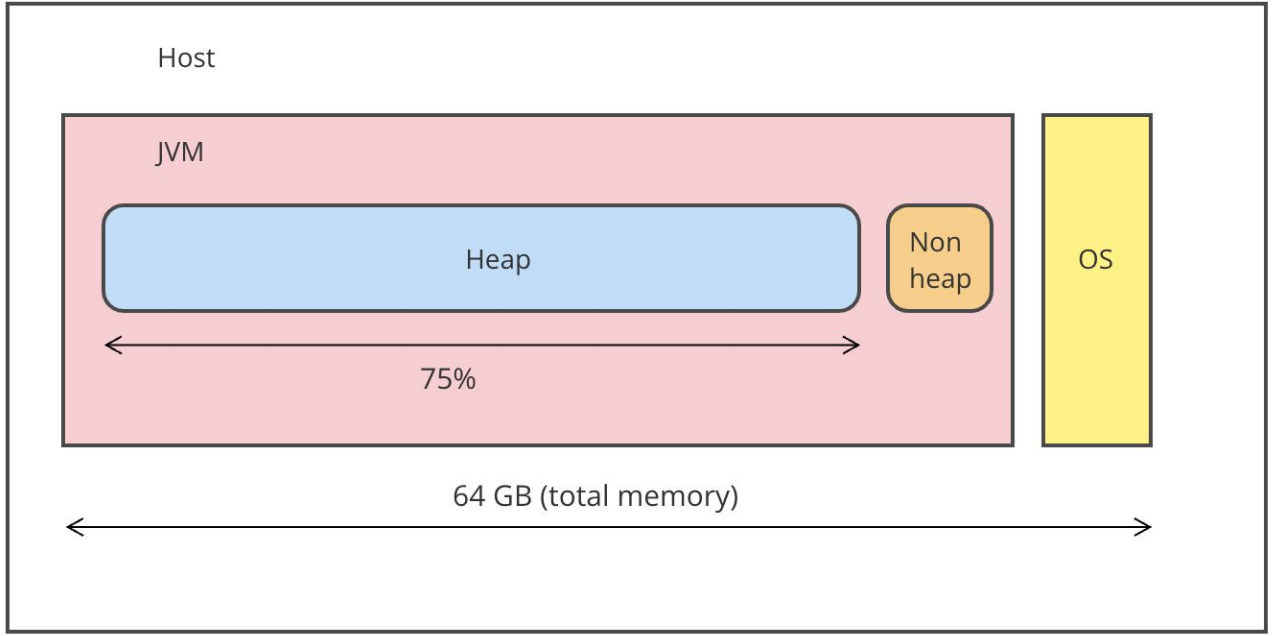
Because…

because OFF-HEAP

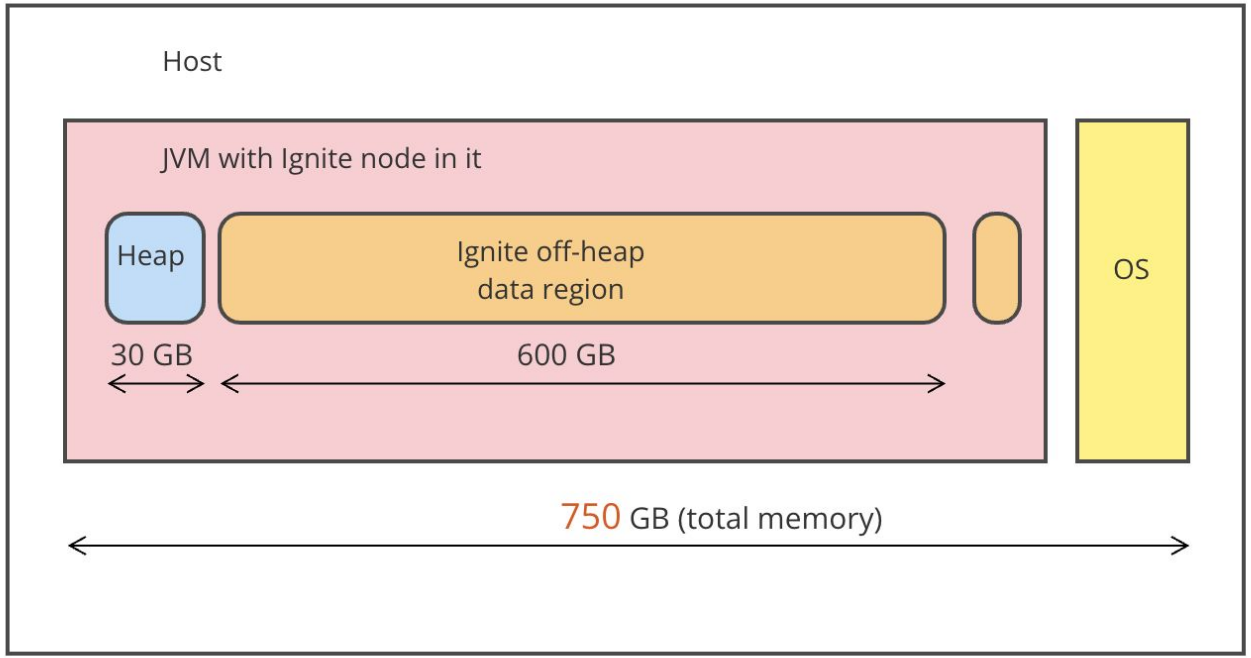
… because DATA COLOCATION & FAILOVER

… because INTEGRATED SERVICES

Typical Java server



Single (!!!) PROD Ignite node

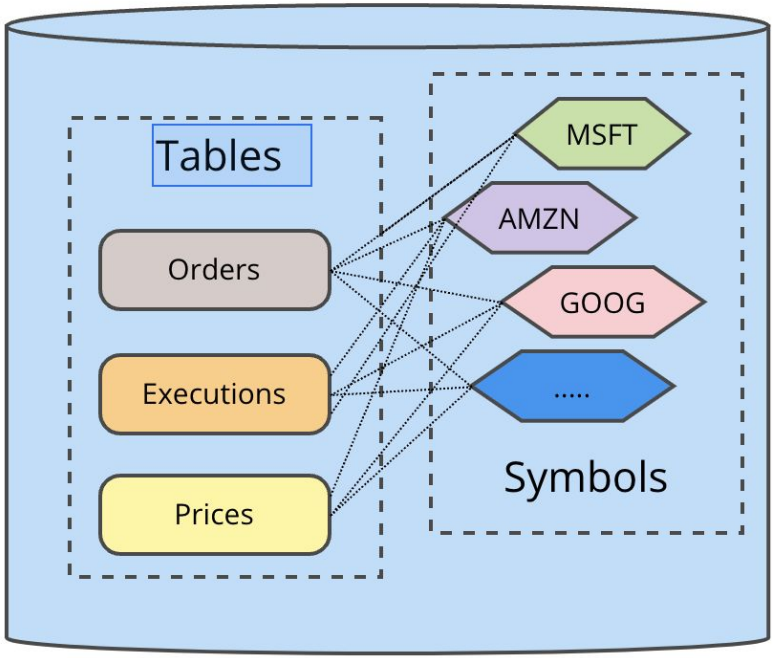


Show me the code

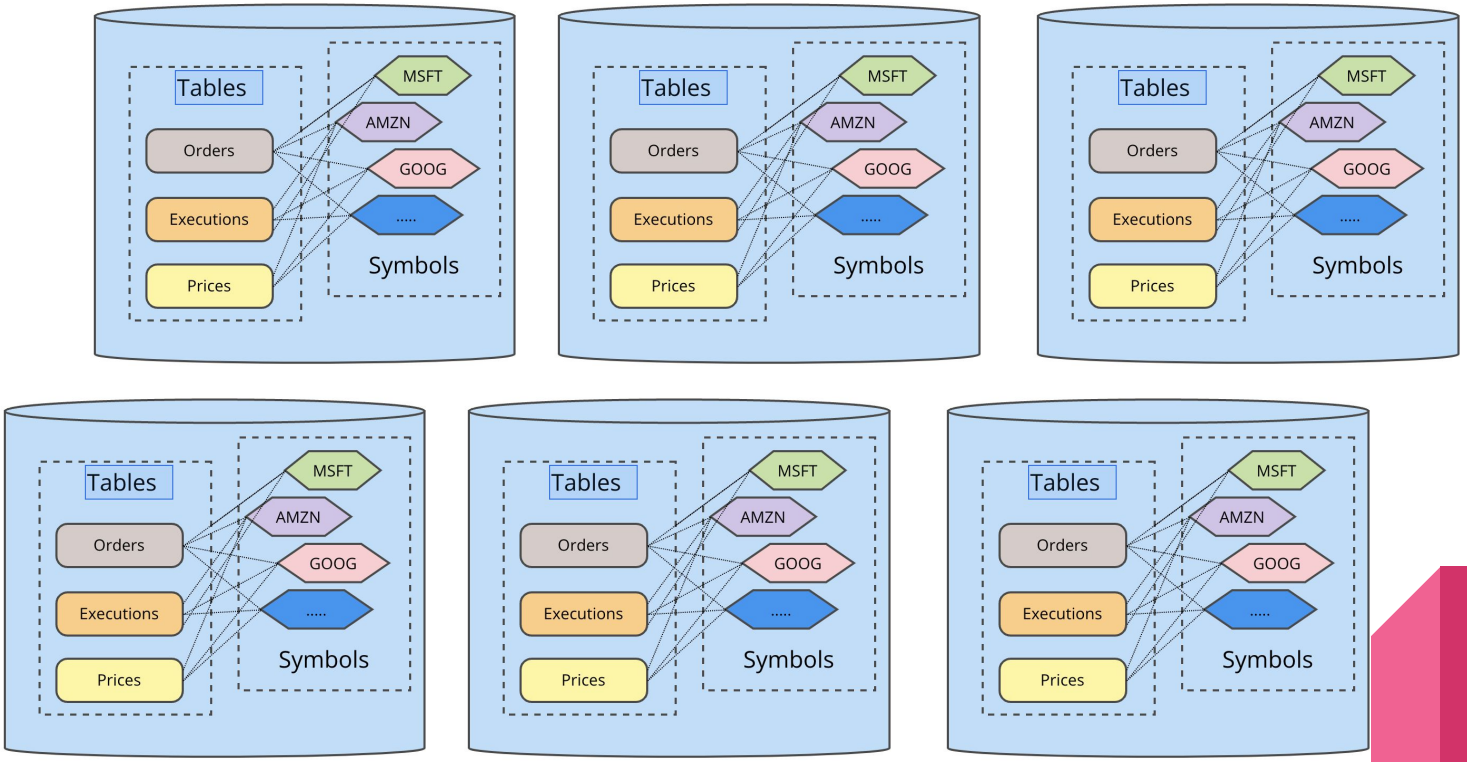
(code)
(code)(code)
(code)(code)(code)(code)
(code)(code)
(code)(code)
(code)(code)(code)
(code)(code)(code)(code)
(code)(code)(code)(code)(code)
(code)(code)(code)(code)(code)(code)

Data colocation - why

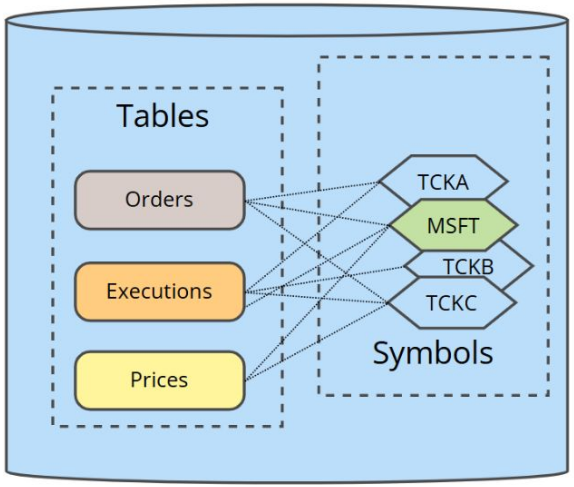
Consider millions of fast-ticking records here:

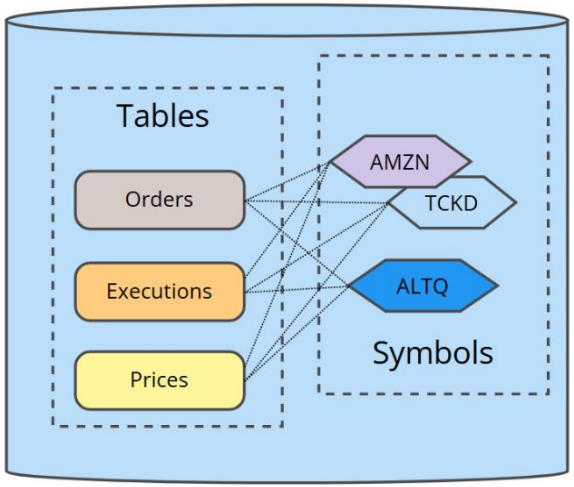


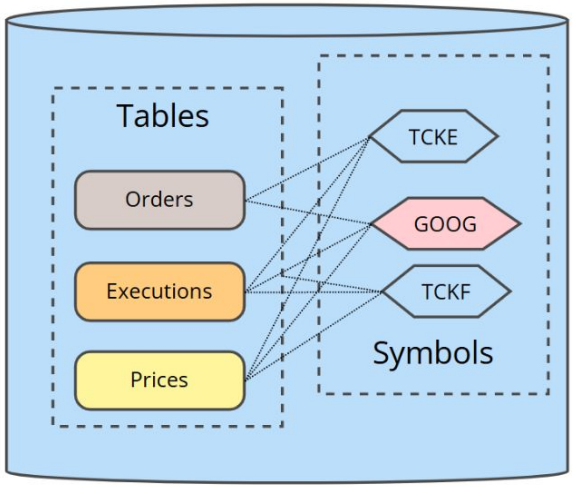
Data colocation - why



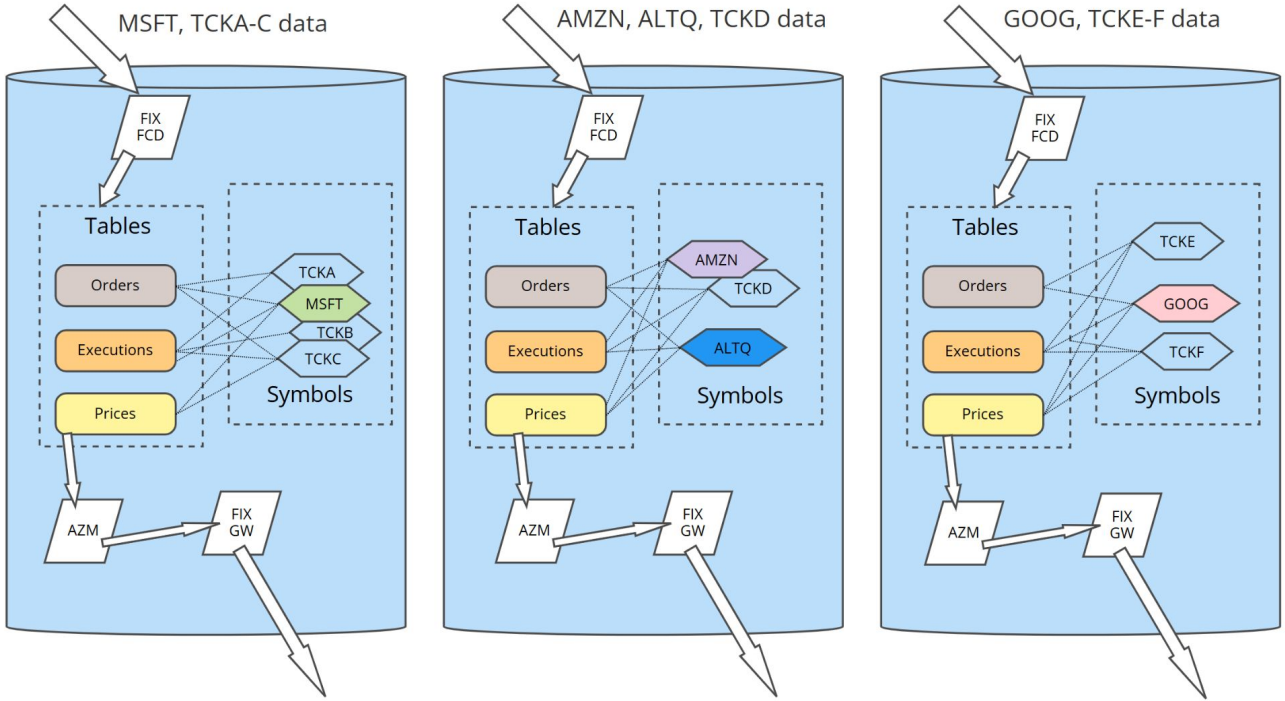
Data colocation - that’s why (A)







Data colocation - that’s why (B)

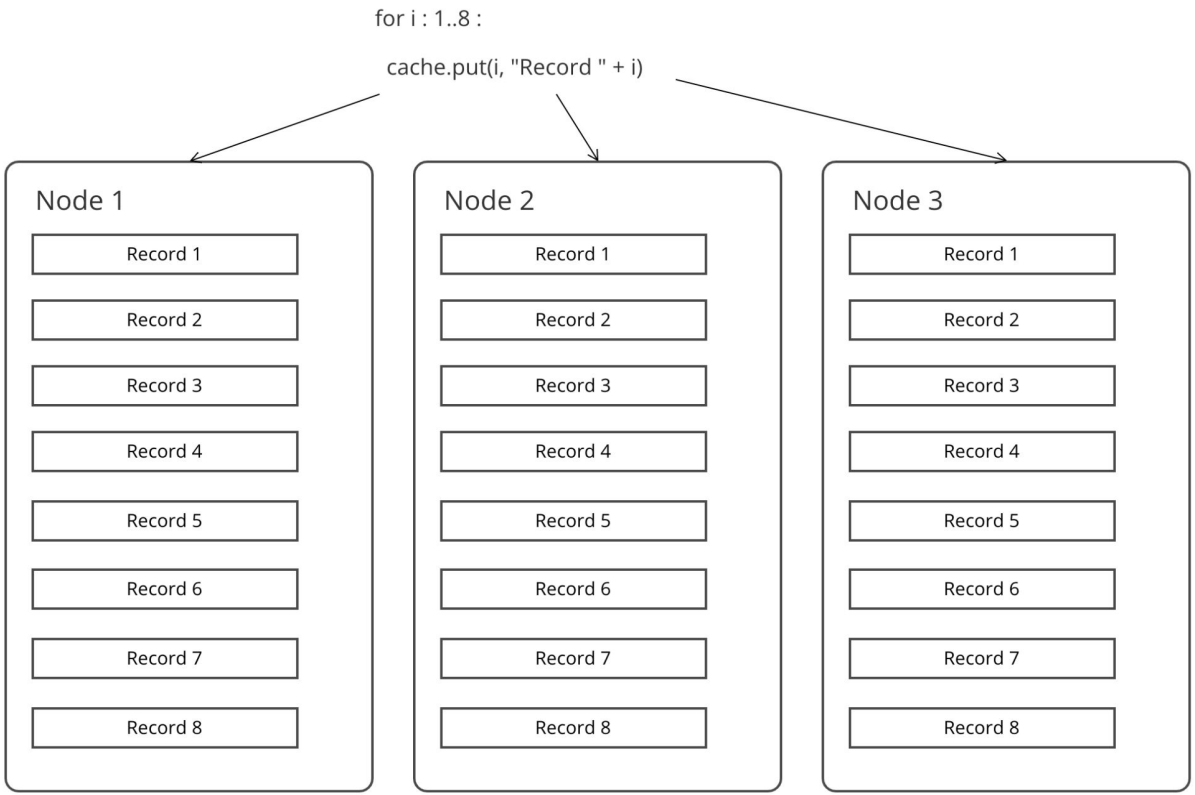


Colocation - how

Ignite caches can have one of two mode:

- REPLICATED
- - or -
- PARTITIONED

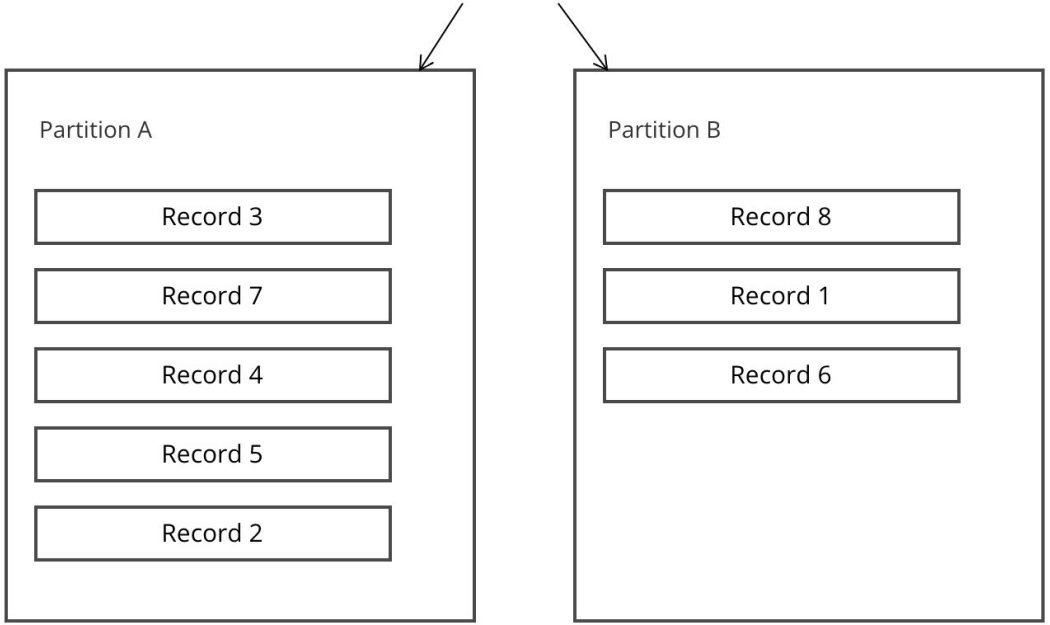
REPLICATED cache



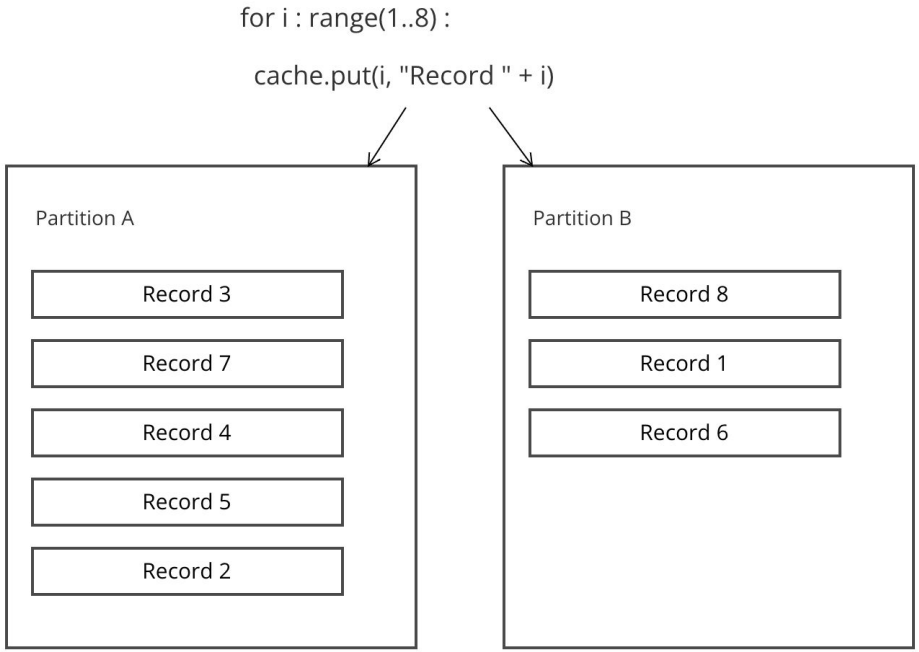
What is a partition?

```
for i : range(1..8) :

  cache.put(i, "Record " + i)
```

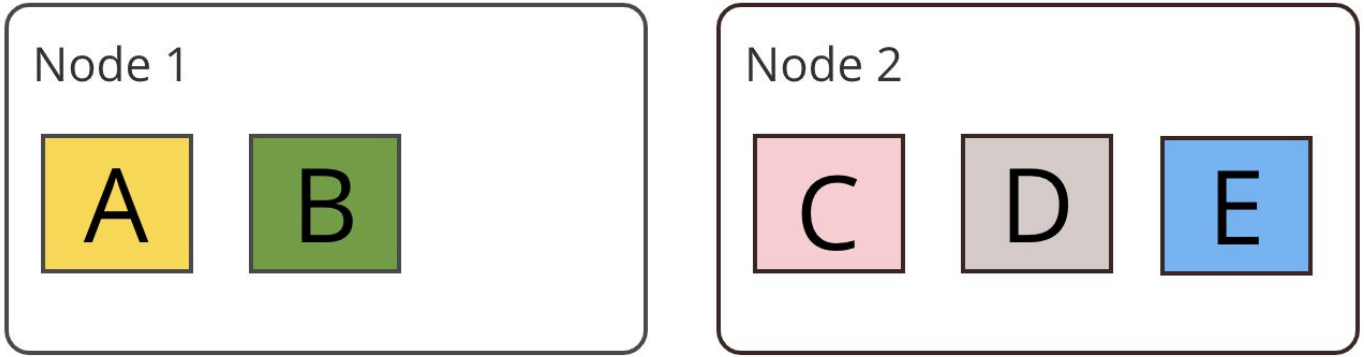


What is a partition?

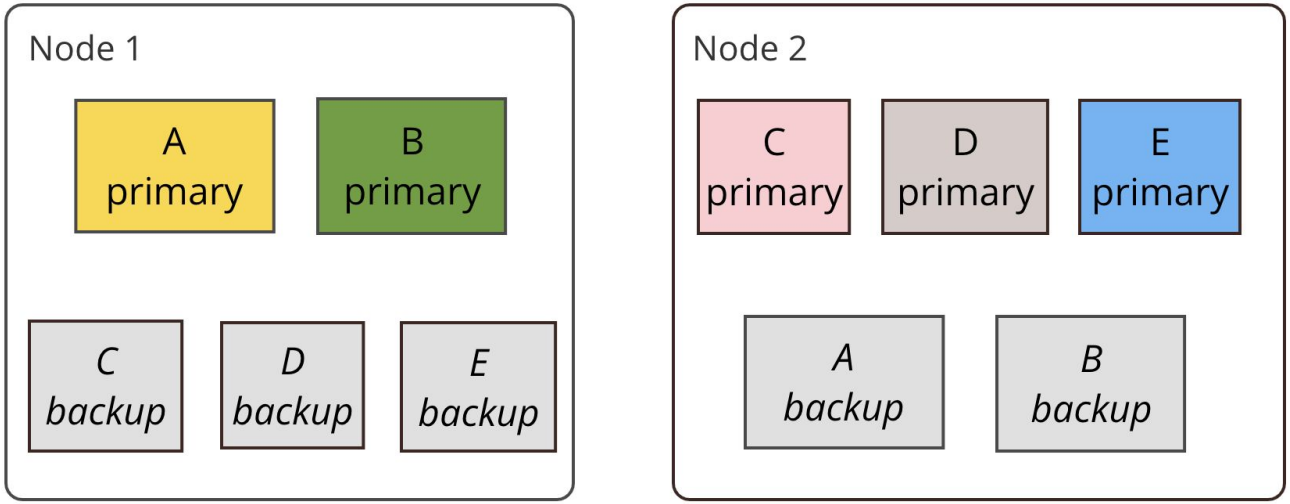


- Cache consists of the set of records.
- Partition is a logical subset of those records.
- Partition is a _stable_ subset, once an ID is assigned to a partition, it never moves to another one.

Simplest case - partitions with no back-ups

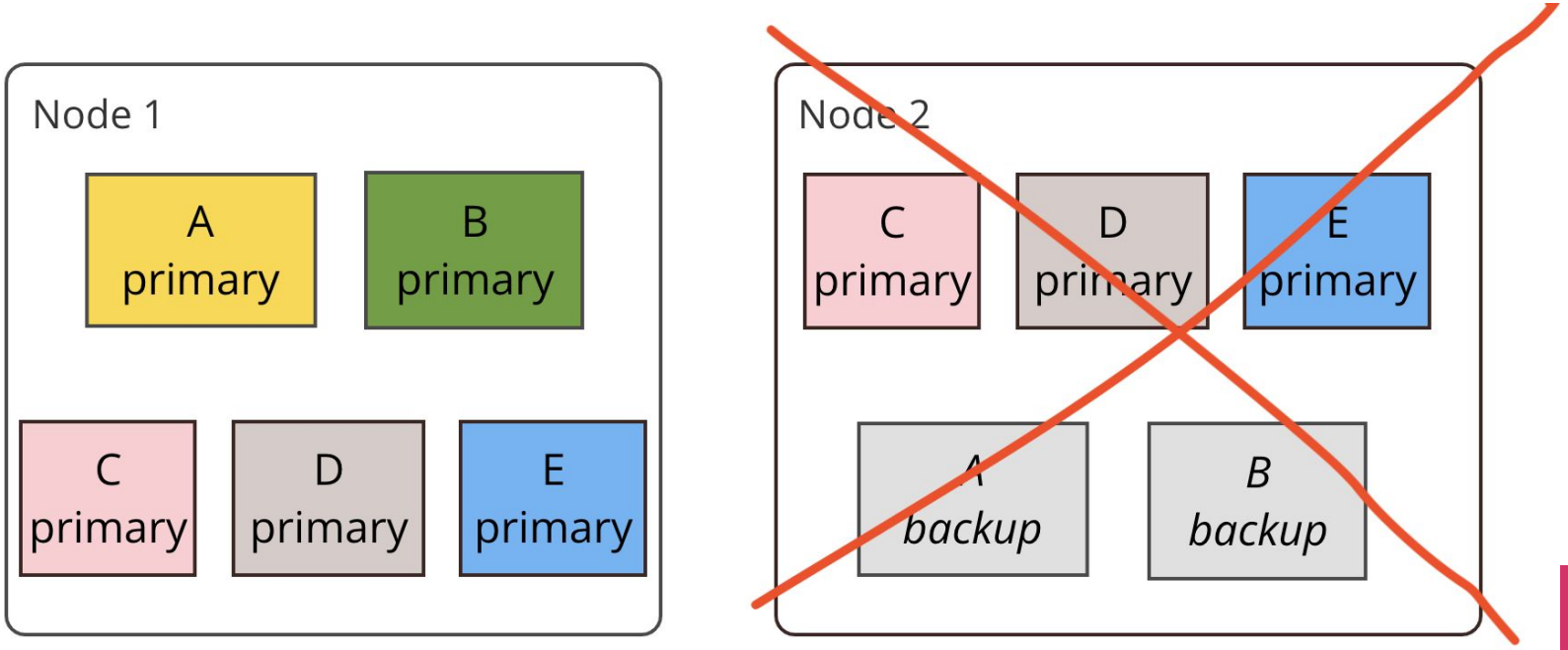


Partitions with backup



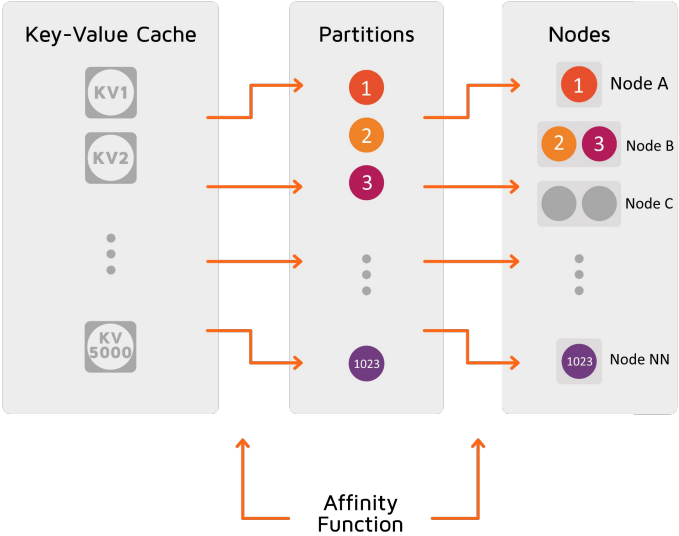
UBER RULE: a partition is never split between the nodes

Partitions with backup - fail-over after a node killed



Affinity function

- The affinity function determines the mapping between keys and partitions.
- Partition are identified by a number from 0 to 1023.
- The set of partitions is distributed between the server nodes available at the moment.
- When the number of nodes in the cluster changes, the partitions are re-distributed.
- The affinity function takes the **affinity key** as an argument.



Affinity key

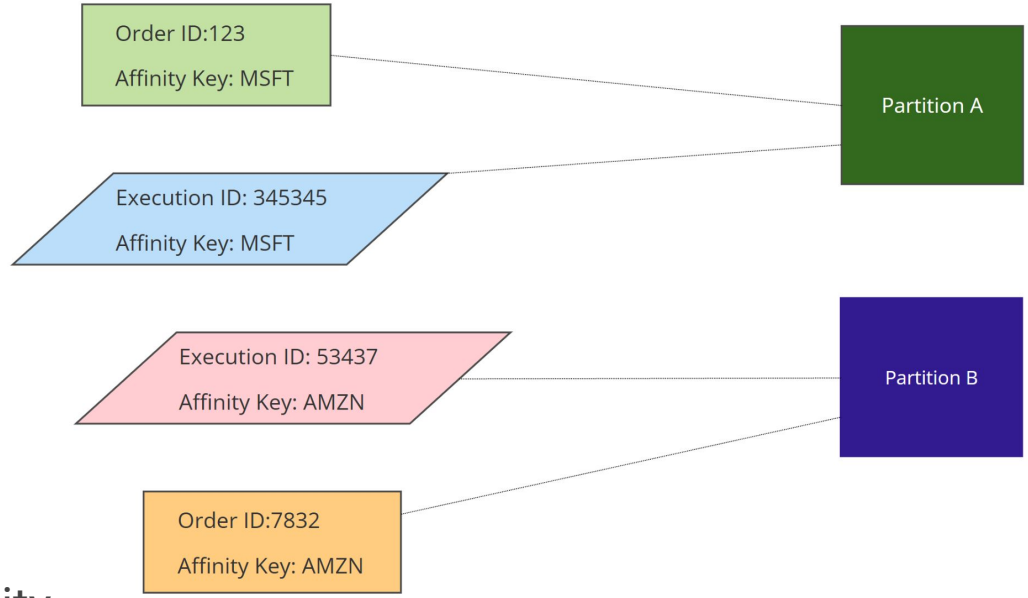
```
8   public class DataKey {
9     public long id;
10
11    @AffinityKeyMapped
12    public final String affinityKey;
13
14    public DataKey(long id, String affinityKey) {
15      this.id = id;
16      this.affinityKey = affinityKey;
17    }
18
```

declares affinity key

Cache with affinity key sample

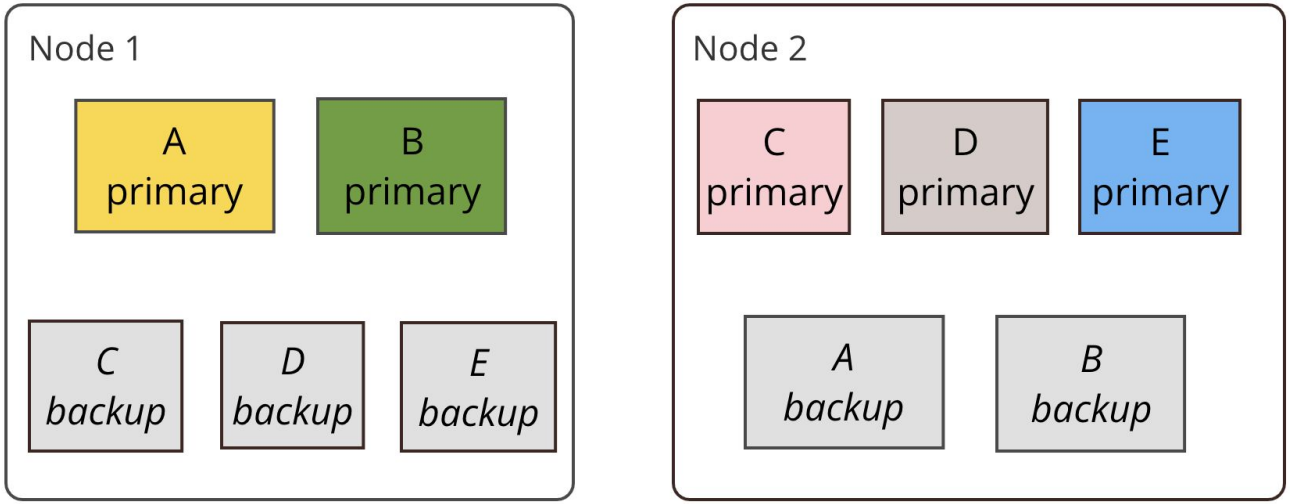
(code)

Data co-location



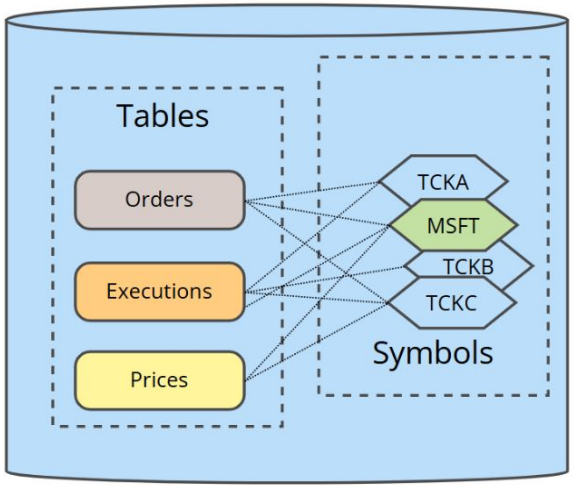
Affinity function looks at the affinity key only, NOT the cache type

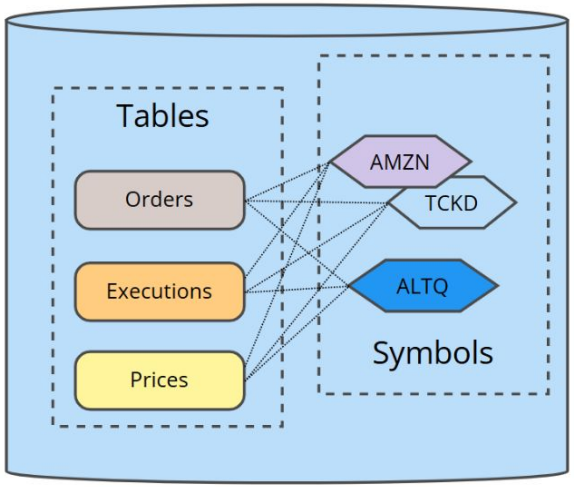
Partitions with backup

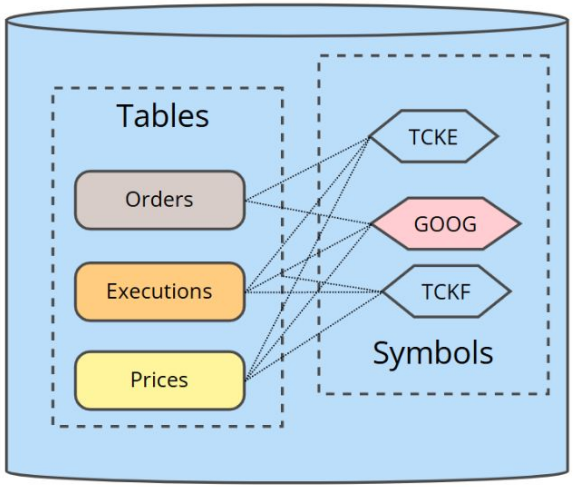


UBER RULE: a partition is never split between the nodes

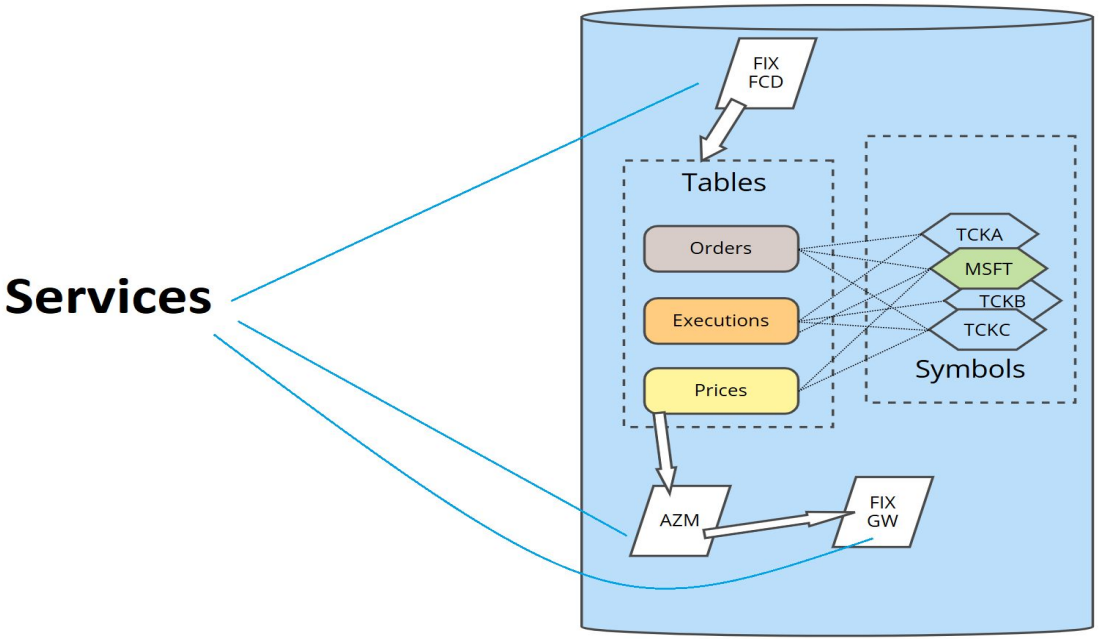
Data is colocated now





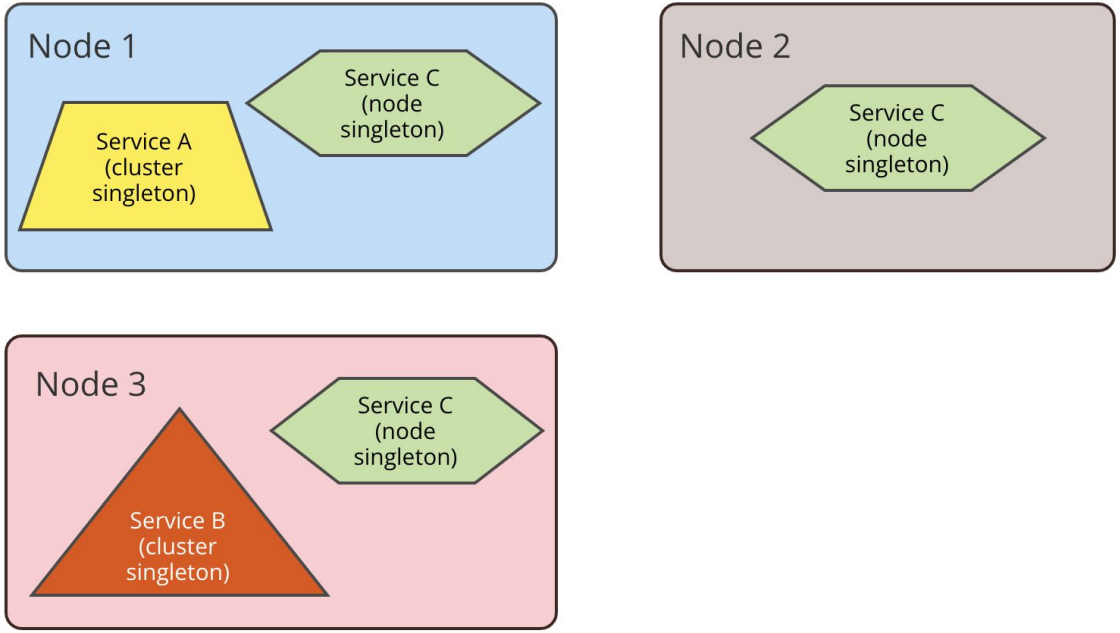


Services


Services
FIX
FCD
Tables
Orders
Executions
Prices
TCKA
MSFT
TCKB
TCKC
Symbols
AZM
FIX
GW

Many types of services-to-nodes-config

- Cluster singleton
- Node singleton
- Affinity services
- …



Services rebalancing