GridGain

Distributed computing with Apache Ignite

Denis Mekhanikov

Jan 29, 2020

Plan

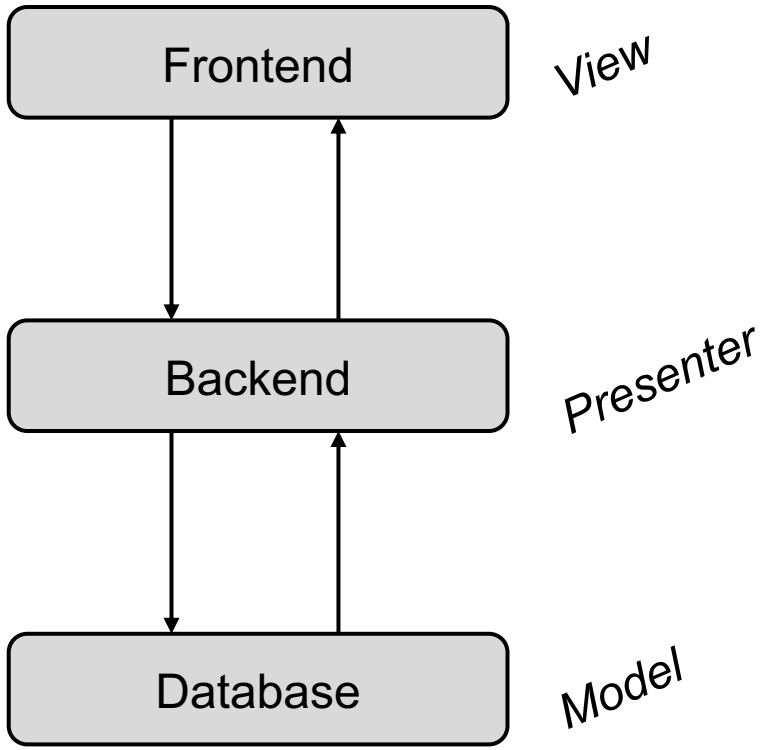
- Traditional view on databases
- Distributed computations
  - Moving computation to data
  - Affinity collocation
  - Map-Reduce
  - Code deployment
  - Work with data
- Distributed SQL

Traditional view on databases

GridGain

Traditional view

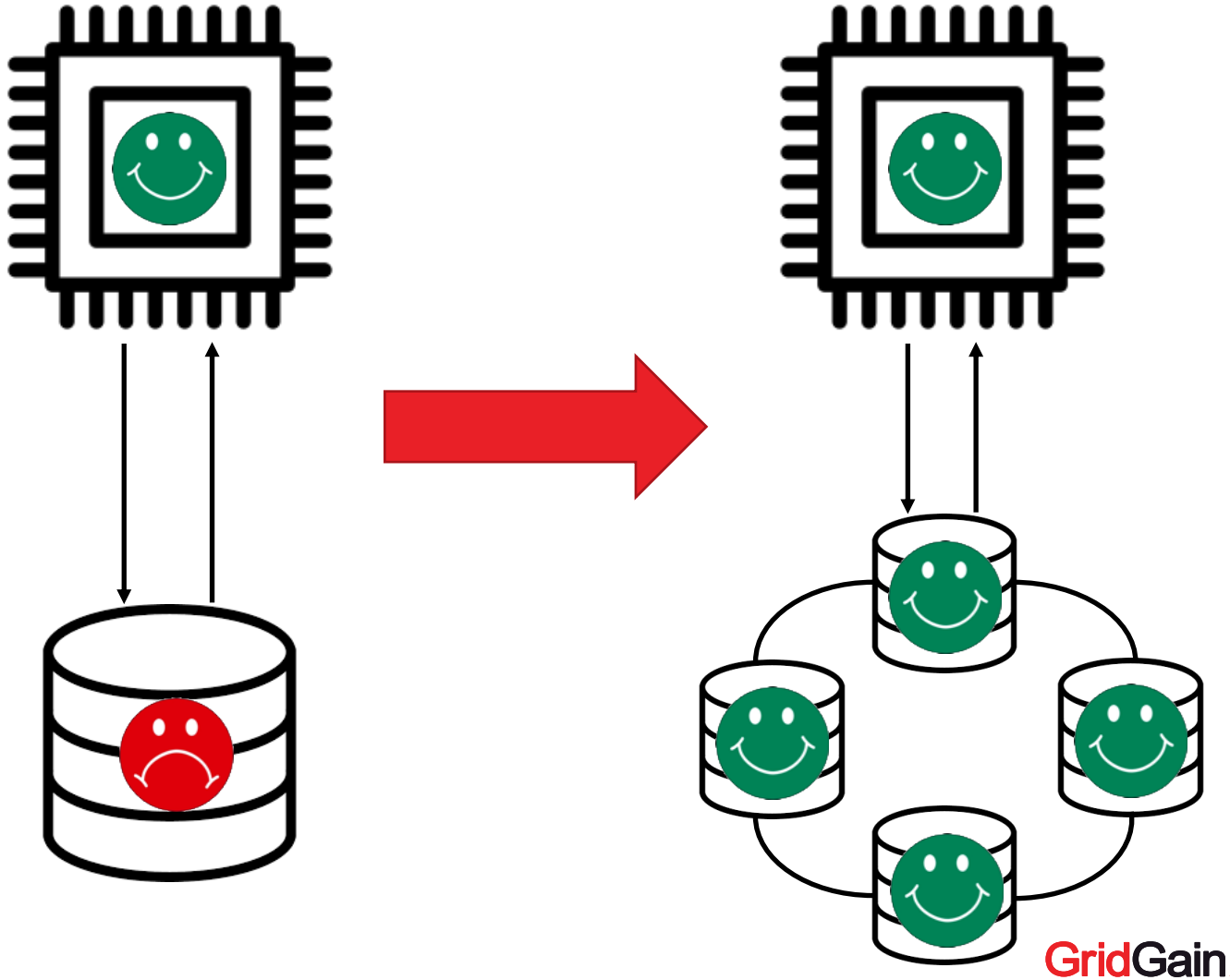
- Data storage layer stays separated
- Data is moved to logic



Migration to a distributed database

- Data is distributed
- The bottlenect is eliminated

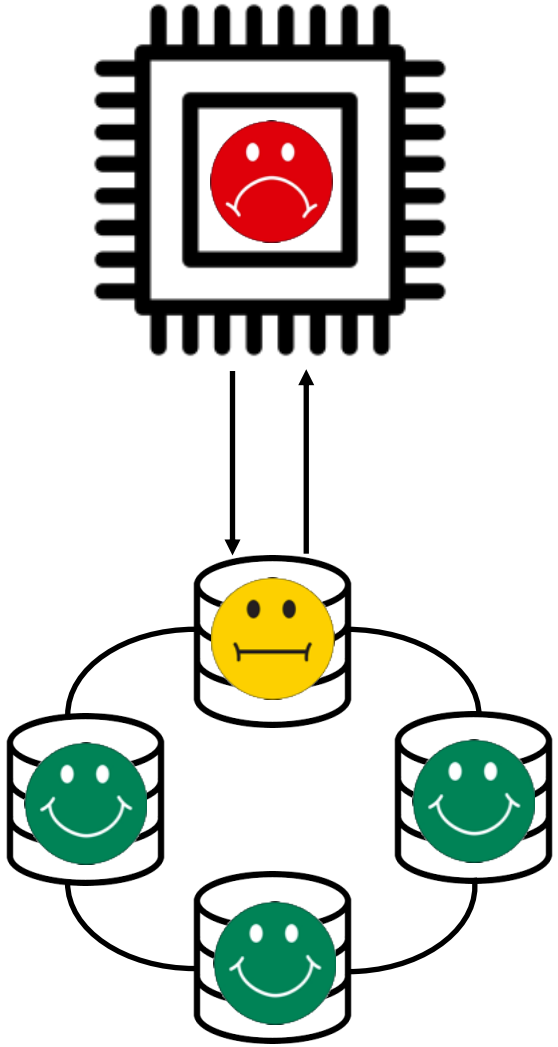
- But over time…



Computation can also become a bottleneck

Two ways to solve this:

- Buy better hardware
- Distribute the computation as well

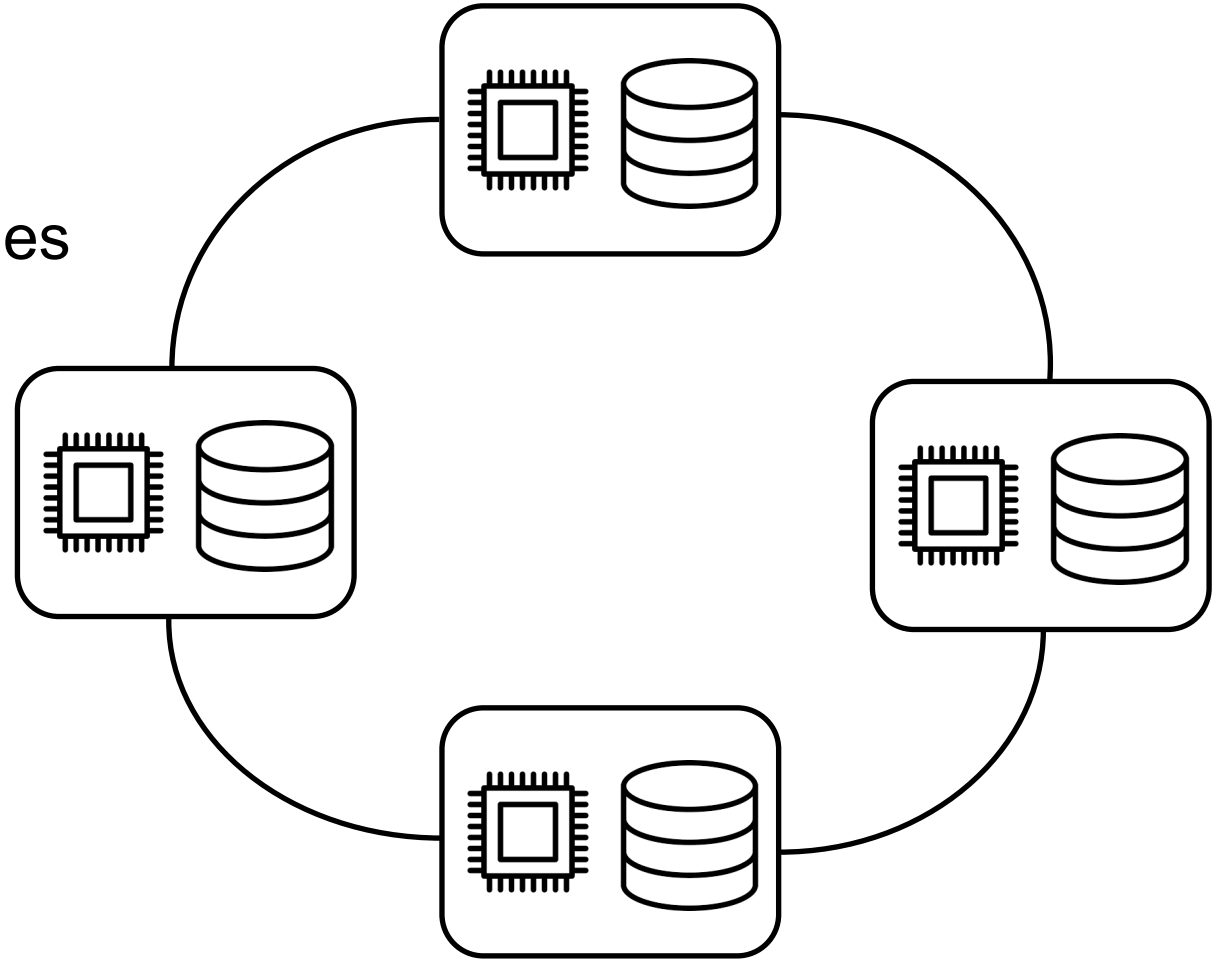


Distributed approach

GridGain

Deploy the application next to the required data

- Computation is sent to data
- Work is evenly distributed among nodes
- Great scalability

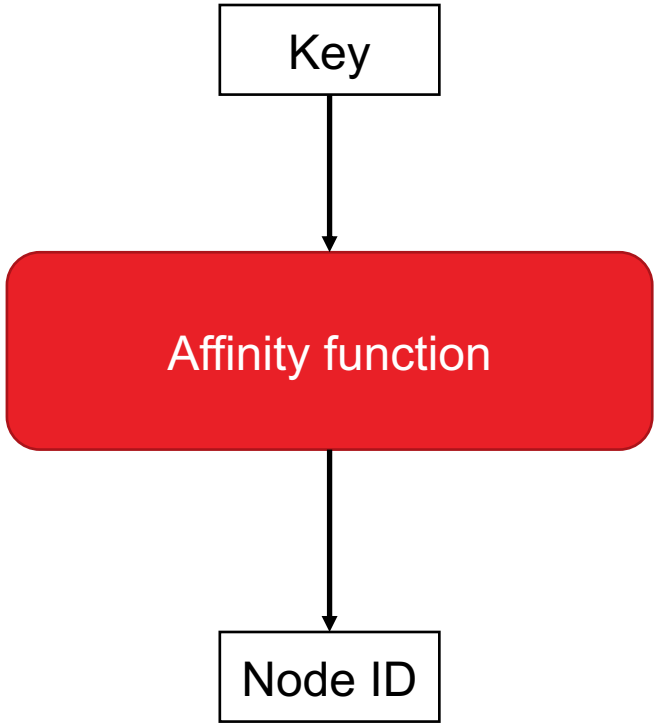


Challenges of a distributed system design

- How to implement load-balancing?
- How to process data stored on multiple nodes?
- How to deploy code into the cluster?

Affinity function

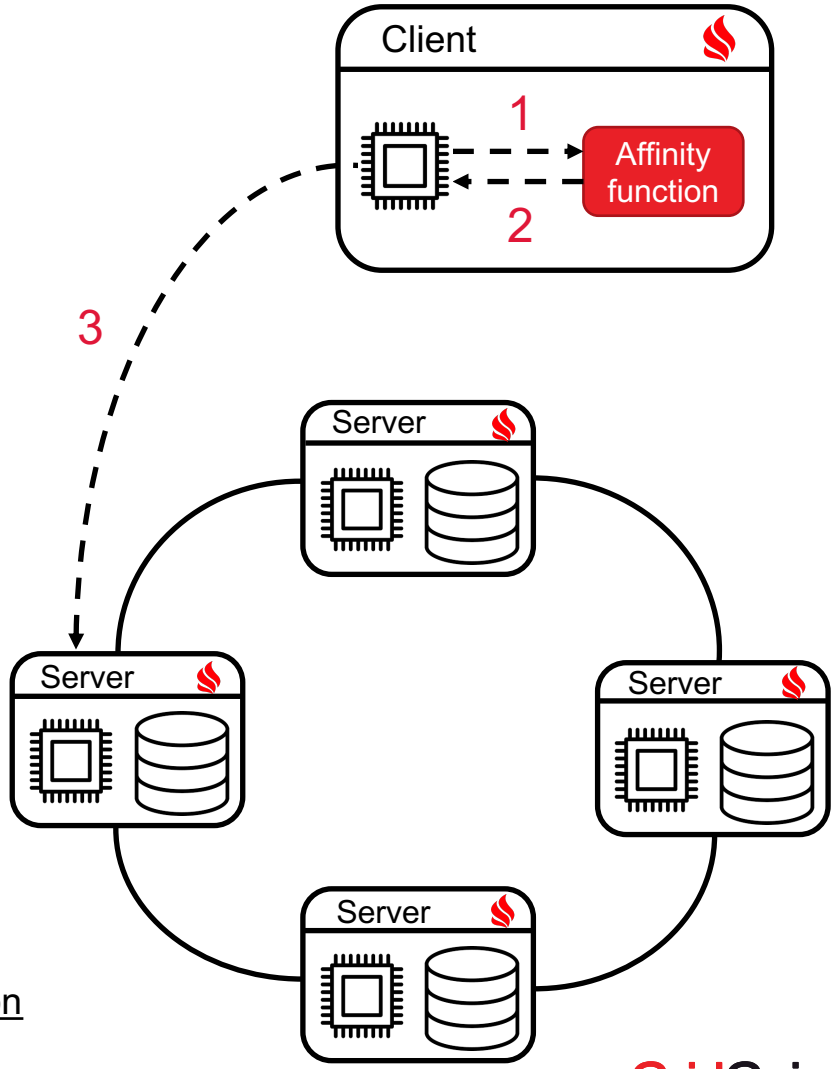
- Distributes the data in the cluster
- Tells where to look for a record by a key
- Can be used for load-balancing



Affinity collocation

- **Data** with **Data**:
  Related data should be kept together

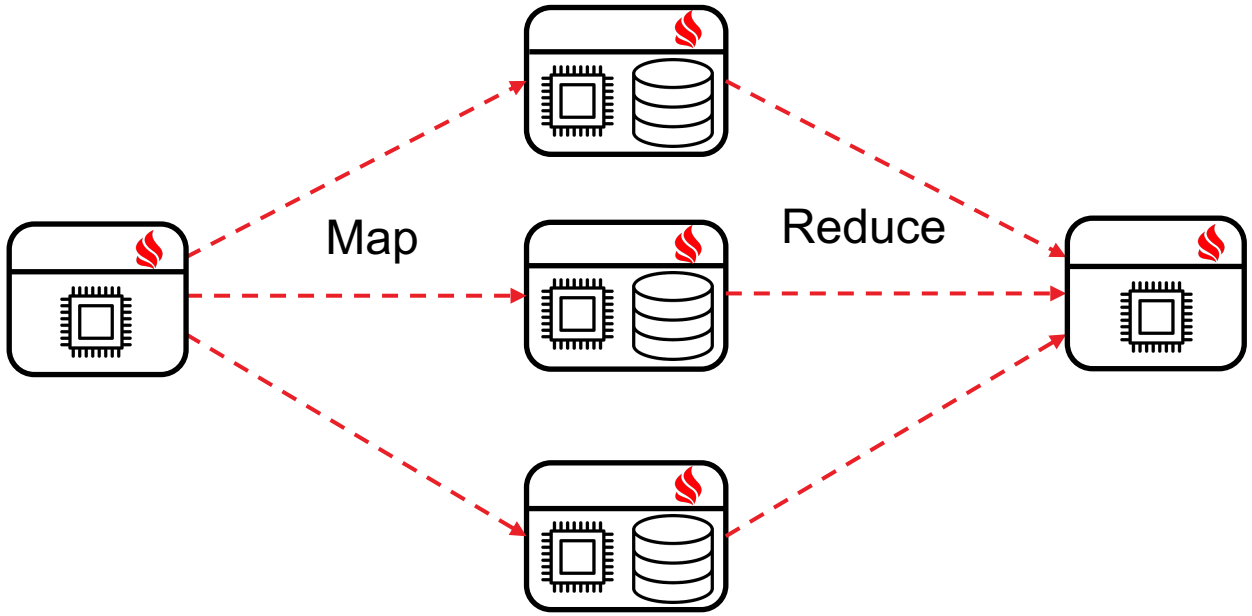
- **Compute** with **Data:**
  Computations can be sent to the data directly



https://www.gridgain.com/docs/latest/developers-guide/data-modeling/affinity-collocation

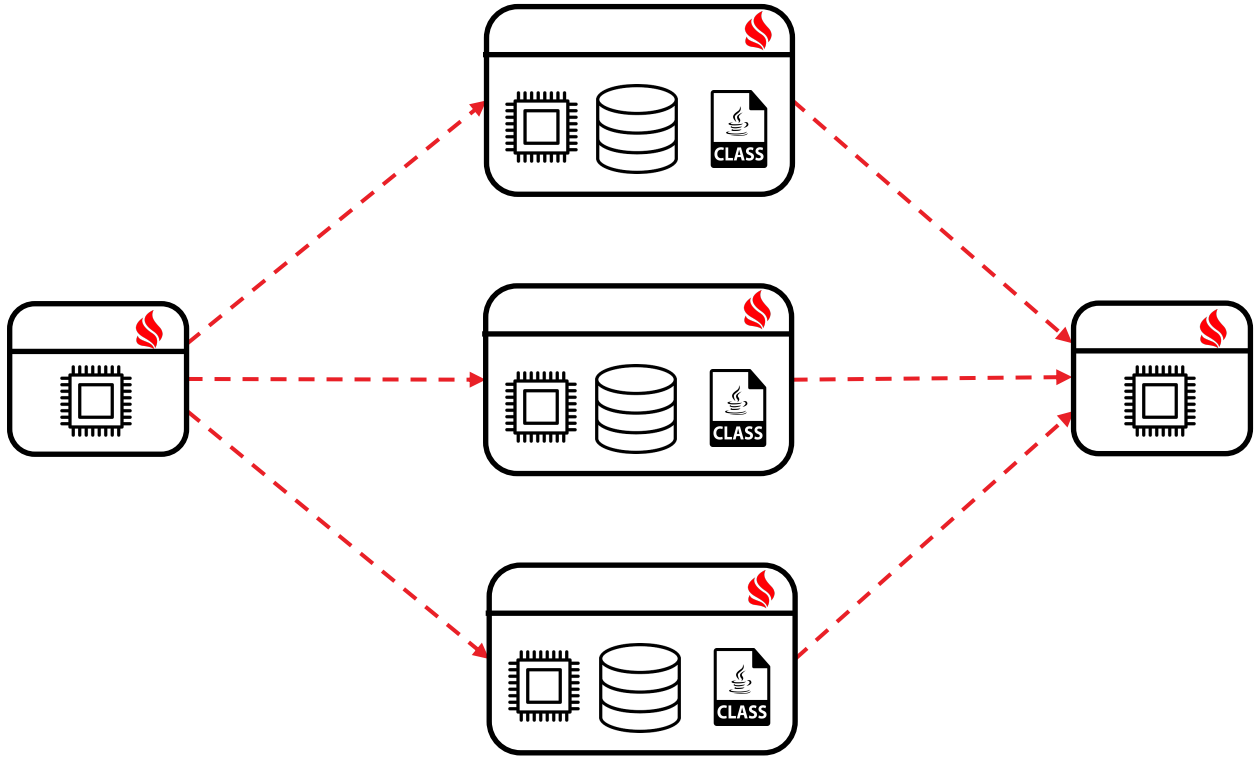
Map-Reduce

1. Map: computation is sent to data nodes
2. Reduce: results are sent back and combined



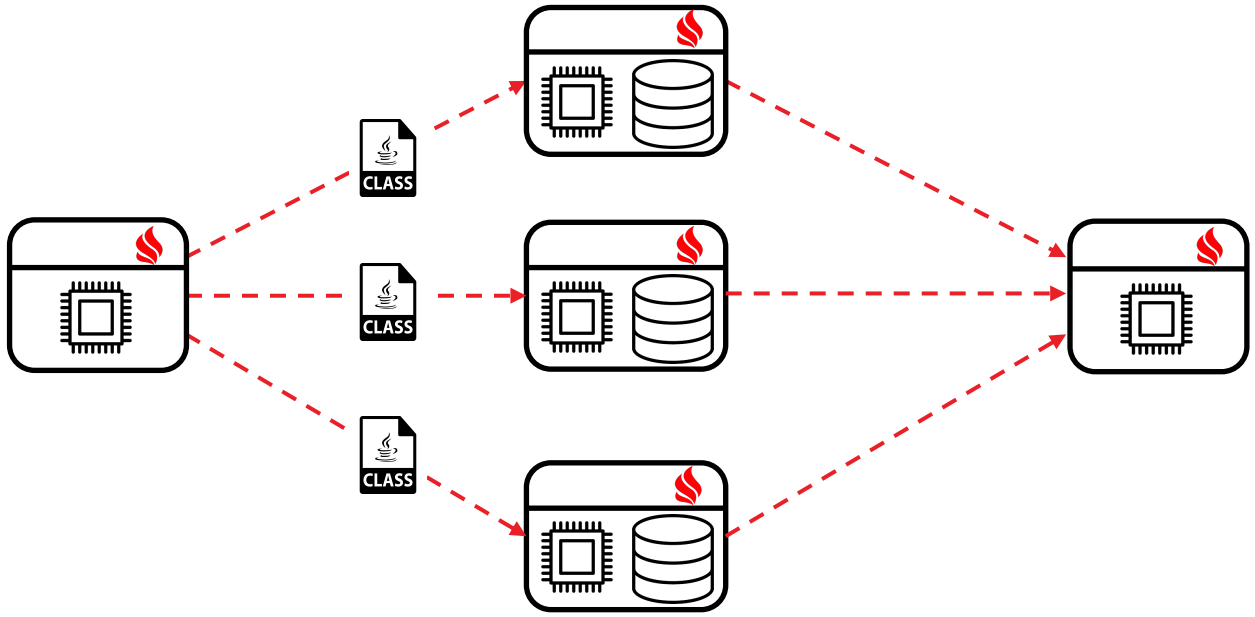
Code deployment: On classpath of every node

- The code is always available on the server side
- Changes in logic require restarting of nodes



Code deployment: Peer deployment

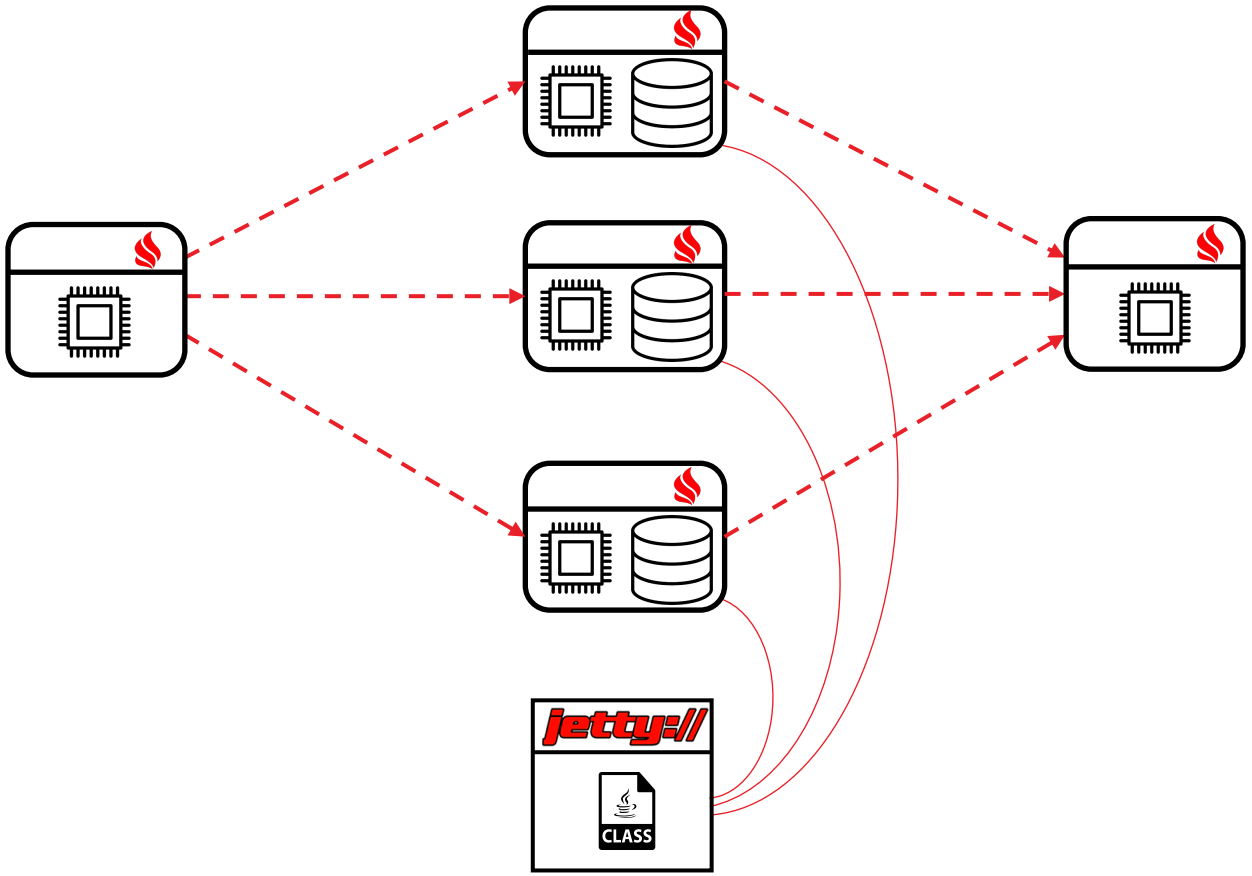
- Classes are loaded over network when requests are made
- Changes in logic require restarting of a client node only



https://www.gridgain.com/docs/latest/developers-guide/peer-class-loading

Code deployment: DeploymentSPI

- Classes are loaded from a dedicated repository
- Need to restart the repository only



https://apacheignite.readme.io/docs/deployment-spi

Work with data: POJO

```
Person p = cache.get(id);

p.age++;

cache.put(id, p);
```

- Convenient to use

- Deserialization takes time
- Peer class loading is not applicable

Work with data: Binary Object

```
BinaryObjectBuilder p =
        cache.get(id).toBuilder();
int age = p.getField("age");
p.setField("age", age + 1);
cache.put(id, p.build());
```

- Data model classes are not needed
- Data is accessed right in the serialized representation

  - More cumbersome

Work with data: Local SQL

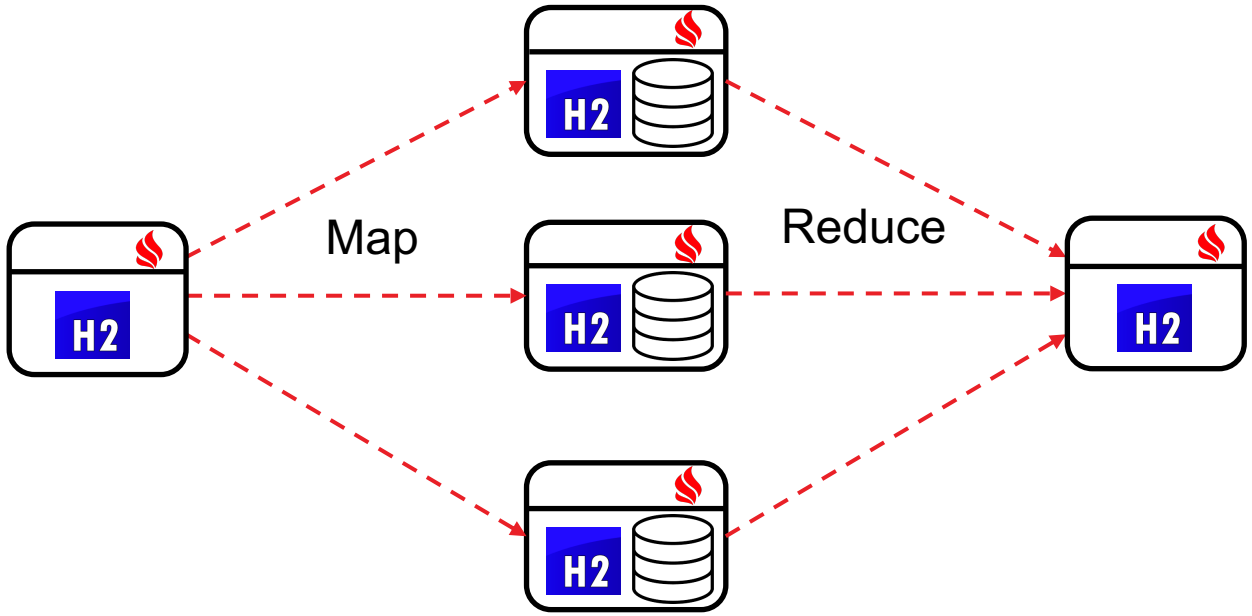
```
Query q = new SqlFieldsQuery(
        "UPDATE people SET age=age+1 " +
        "WHERE id=" + id);
q.setLocal(true);
cache.query(q);
```

- Data model classes are not needed
- SQL indexes can be used

Distributed SQL

Distributed SQL

- The same execution scheme
- No need to implement the mapping and reducing logic



Code deployment

Not needed. Yay!

Demo

Thanks!

Denis Mekhanikov
dmekhanikov@gmail.com

https://github.com/dmekhanikov/ignite-social-network

https://www.kaggle.com/kazanova/sentiment140