CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Computing@CIT

Distributed Data Management

Lecture 7: Demonstrating Sharding

Outline

1. MongoDB Independent Nodes.
2. Replica Set.
3. Configuration Nodes and Cluster Interface.

Outline

1. MongoDB Independent Nodes.
2. Replica Set.
3. Configuration Nodes and Cluster Interface.

MongoDB Independent Nodes

- MongoDB: A database engine is created via a **mongod** process.
- It is characterised by:
  i. The machine on which it is running.
  ii. The folder of the machine in which it stores its data.
  iii. The ip address and the port of the machine it listens to, so that clients can connect to it.



MongoDB Independent Nodes

- If ii and iii are not explicitly specified then a default location is chosen, e.g. “/var/lib/mongodb/data/db” on Linux, and port 27017 is used by default.
  - Any other folder and almost any port can be explicitly specified (using mongod.conf or command line parameters).
- Once the database engine is

Set up, we can connect clients to it.

Client 1: Connected to
(machine m, folder f and port p)
…
Client n: Connected to
(machine m, folder f and port p)

MongoDB Independent Nodes

Demonstrating MongoDB Independent Nodes:

- Create three folders in my machine: dir1, dir2 and dir3.
- Select three different ports of my machine: 27000, 27001, 27002.

Note: using --host with a hostname would be better that using an ip address

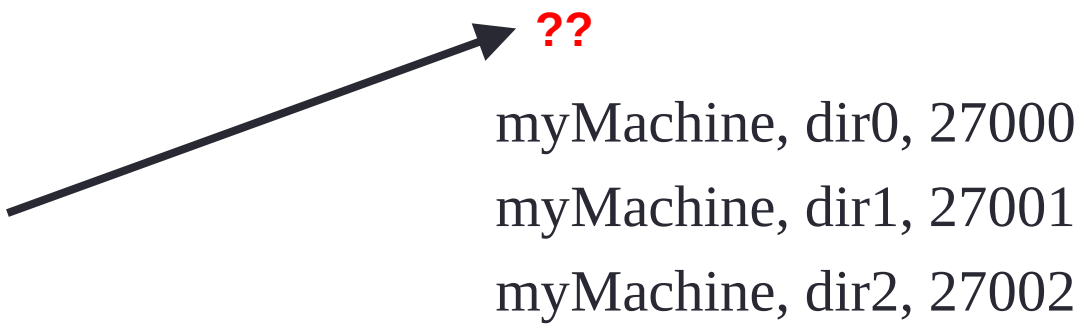
- Open three terminals and create 3 mongod.exe processes:
  - start /b mongod --dbpath dir0 --port 27000 --bind_ip 127.0.0.1
  - start /b mongod --dbpath dir1 --port 27001 --bind_ip 127.0.0.1
  - start /b mongod --dbpath dir2 --port 27002 --bind_ip 127.0.0.1

Note: `start \b` tells Windows to start a background process (on Linux or Mac Can append & to the end of your command to do the same.

MongoDB Independent Nodes

Demonstrating MongoDB Independent Nodes:

- If I try to connect a client (mongo.exe process) using my machine and the default port (27017) obviously it will fail.
  - There is no database engine (mongod.exe process) listening to this port.



MongoDB Independent Nodes

Demonstrating MongoDB Independent Nodes:

- If I try to connect a client (mongo process) using my machine and the port (27000), I connect to the first node, but not to the second or the third ones.
  - It connects by default to the test database of the node.

mongo --port 27000 

myMachine, dir0, 27000
myMachine, dir1, 27001
myMachine, dir2, 27002

MongoDB Independent Nodes

Demonstrating MongoDB Independent Nodes:

- If I create a new collection practise with 2 documents:

  { “name” : “John”},

  { “name” : “Marie”}

  Obviously the content will be stored in the node dir0, but the node dir1 and the node dir2 have no clue of this content.

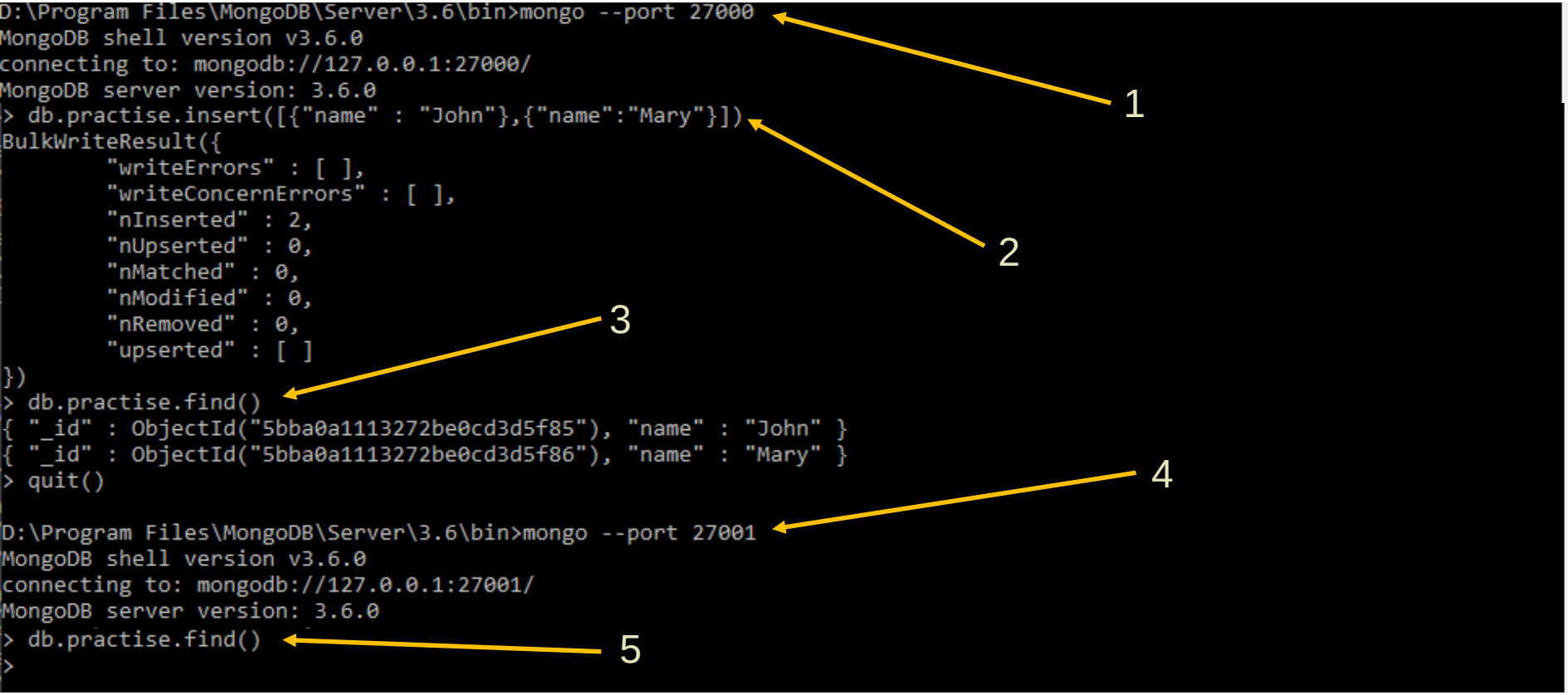
mongo.exe --port 27000 →

myMachine, dir0, 27000

myMachine, dir1, 27001

myMachine, dir2, 27002

MongoDB Independent Nodes



(1) We connect to node 1, then (2) insert 2 documents, then (3) retrieve them to demonstrate they are on node 1, then (4) quit out and connect to node 2, then (5) demonstrate the 2 documents are not on any other node

MongoDB Independent Nodes

Demonstrating MongoDB Independent Nodes:

- Obviously, when I close the client and the database engine the folder dir0 still contains the new collection practise and its two documents, but the data is not available as there is no database engine a client can connect to so as to query the data.

MongoDB Independent Nodes

Demonstrating MongoDB Independent Nodes:

- Conclusion:
  1. The nodes dir0, dir1 and dir2 are independent.

Please note that, in this case, the 3 nodes are on my own computer, but they could have been in different machines (using TCP/IP routing via LANs, WANs or internet).

Example:

dir0 in my computer,

dir1 in the data centre of the CIT (room C130, close to our lab).

dir2 in a data centre in Dublin.

MongoDB Independent Nodes

Demonstrating MongoDB Independent Nodes:

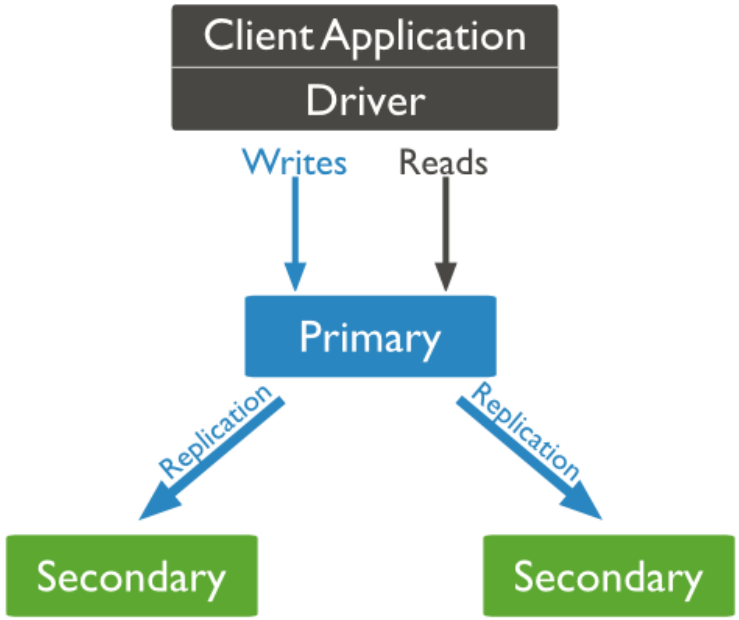
- Conclusions:
  1. The nodes dir0, dir1 and dir2 are independent.
  2. The data stored on each node is persistent.
  3. Each node needs a database engine (mongod process) to make the data available to the clients connecting to it.

Outline

1. MongoDB Independent Nodes.
2. Replica Set.
3. Configuration Nodes and Cluster Interface.

Replica Set

- A MongoDB replica set allows independent nodes to be linked, so that they act as true replicas of the same content.
- We need at least 3 nodes to set up a replica set:
  - We use the dir0, dir1 and dir2 nodes of previous example.
  - Again, the nodes of a replica set can be on different machines geographically distributed.
  - We will set dir0 as primary and dir1 and dir2 as secondary nodes. In any case, this is susceptible to change later on.

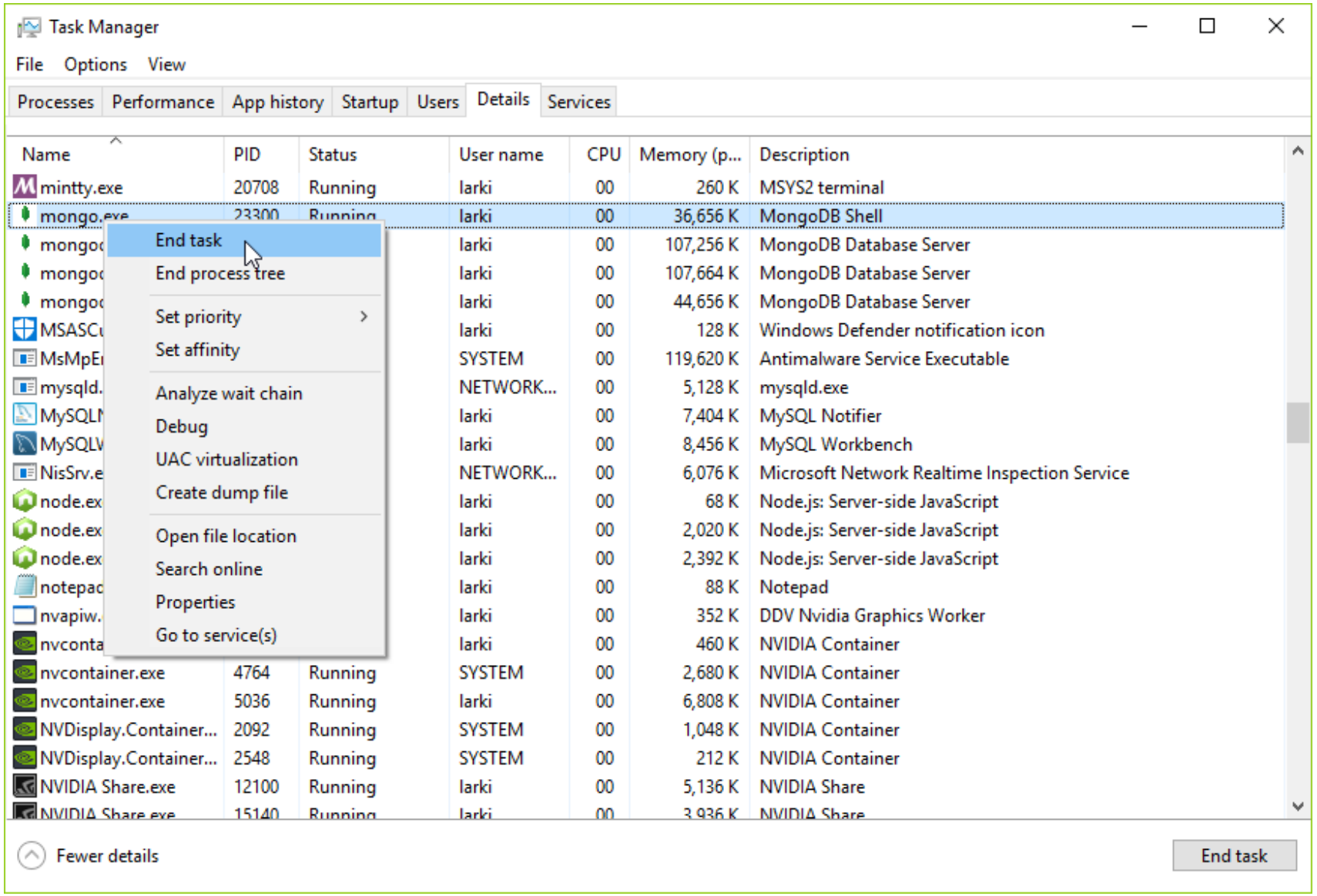


Replica Set

Demonstrating MongoDB Replica Set.

- We reproduce the steps of the previous section:
  - Create 3 folders: dir0, dir1, dir2.
  - Start three mongod.exe processes (making sure to kill the previous mongod processes), but now we include an additional option --replSet dir, which enables replication on these database engines.
    - `start /b mongod --replSet rs0 --dbpath dir0 --port 27000 --bind_ip 127.0.0.1`
    - `start /b mongod --replSet rs0 --dbpath dir1 --port 27001 --bind_ip 127.0.0.1`
    - `start /b mongod --replSet rs0 --dbpath dir2 --port 27002 --bind_ip 127.0.0.1`
  - Note: enabling *replSet* does not set them up automatically. It just allows them to be explicitly set up later. *rs0* is just a name we picked for the replica set (we will add all 3 nodes to rs0 later).

One way to kill monog processes in Windows (e.g. Ctrl+Alt+Del, then Task Mgr); sort by process name, the select and click End Task (or use right-click menu).



Replica Set

Demonstrating MongoDB Replica Set.

- We connect to dir0: mongo.exe --port 27000
- We create a variable with the ip address, as we need to use it for specifying the 3 nodes that would be set up as replica set.

```
var myName = "COM-C132-L51877";
```

Replica Set

Demonstrating MongoDB Replica Set.

- We run the command **rs.initiate** to set up the replica set with its 3 node members dir0, dir1 and dir2:

```
rs.initiate(
    {
        "_id" : "rs0",
        "members" : [
            { _id:0,host:"127.0.0.1:27000" },
            { _id:1,host:"127.0.0.1:27001" },
            { _id:2,host:"127.0.0.1:27002" }
        ]
    }
);
```

Replica Set

```
> rs.initiate(
...     {
...         "_id" : "rs0",
...         "members" : [
...             { _id:0,host:"127.0.0.1:27000" },
...             { _id:1,host:"127.0.0.1:27001" },
...             { _id:2,host:"127.0.0.1:27002" }
...         ]
...     }
... );
{
        "ok" : 1,
        "operationTime" : Timestamp(1538922013, 1),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1538922013, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
rs0:OTHER>
```

Prompt changes to rs0:OTHER initially, but will change to rs0:PRIMARY once replica set fully initialized.

Replica Set

Demonstrating MongoDB Replica Set.

- Once the command finishes we can use the command rs.status() to check the status of the 3 nodes of the replica set.
  - It might take a few seconds until all the nodes are initialized as part of the replica set and set as primary (i.e. mongod on port 27000) or secondary members (i.e. mongod on port 27001/2).

```
D:\Program Files\MongoDB\Server\3.6\bin>mongo --port 27001
rs0:SECONDARY>
```

Replica Set

```
rs0:PRIMARY> rs.status()
{
        "set" : "rs0",
        "date" : ISODate("2018-10-07T14:21:59.194Z"),
        "myState" : 1,
        "term" : NumberLong(1),
        "heartbeatIntervalMillis" : NumberLong(2000),
        "members" : [
                {
                        "_id" : 0,
                        "name" : "127.0.0.1:27000",
                        "health" : 1,
                        "state" : 1,
                        "stateStr" : "PRIMARY",
                        "uptime" : 148,
                        "optime" : {
                                "ts" : Timestamp(1538922117, 1),
                                "t" : NumberLong(1)
                        },
                        "optimeDate" : ISODate("2018-10-07T14:21:57Z"),
                        "infoMessage" : "could not find member to sync from",
                        "electionTime" : Timestamp(1538922025, 1),
                        "electionDate" : ISODate("2018-10-07T14:20:25Z"),
                        "configVersion" : 1,
                        "self" : true
                },
                {
                        "_id" : 1,
                        "name" : "127.0.0.1:27001",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 105,
                        "optime" : {
                                "ts" : Timestamp(1538922117, 1),
                                "t" : NumberLong(1)
```

Outline

1. MongoDB Independent Nodes.
2. Replica Set.
3. Configuration Nodes and Cluster Interface.

Configuration Nodes and Cluster Interface

Metadata:

- Once we understand how to create independent nodes and link them as a replica set, the concept of configuration nodes is straightforward.

- A configuration node is a database engine just like the ones we have seen in the past two sections, but it requires the option --configsvr to let MongoDB know that the node is going to contain metadata.
  - `start /b mongod --configsvr --dbpath cfg0 --port 26050 --bind_ip 127.0.0.1`
  - `start /b mongod --configsvr --dbpath cfg1 --port 26051 --bind_ip 127.0.0.1`
  - `start /b mongod --configsvr --dbpath cfg2 --port 26052 --bind_ip 127.0.0.1`

Configuration Nodes and Cluster Interface

Metadata:

- In the MongoDB version installed in the labs, the configuration servers do not need to be explicitly set up as a replica set, the parameter --configsvr does it in the background.
  - `start /b mongod --configsvr --dbpath cfg0 --port 26050 --bind_ip 127.0.0.1`
  - `start /b mongod --configsvr --dbpath cfg1 --port 26051 --bind_ip 127.0.0.1`
  - `start /b mongod --configsvr --dbpath cfg2 --port 26052 --bind_ip 127.0.0.1`
- In the most recent version of MongoDB (version 3.4) the configuration servers are set up as a replica set in the same way we saw for the previous section with the dir example.
- Once again, although in this example all configuration nodes are in my machine, they can be in geographically distributed machines.

Configuration Nodes and Cluster Interface

Config Server Replica Set Initiation:

```
rs.initiate(
    {
        "_id" : "cfg",
        configsvr: true,
        "members" : [
            { _id:0,host:"127.0.0.1:26050" },
            { _id:1,host:"127.0.0.1:26051" },
            { _id:2,host:"127.0.0.1:26052" }
        ]
    }
);
```

Configuration Nodes and Cluster Interface

You can choose another port

Now create the shard service (mongos):

```
mongos --configdb cfg/127.0.0.1:26050 --port 27011
```

You'll see output including something like this (note it adds all nodes in the *cfg* replica set as config DBs):

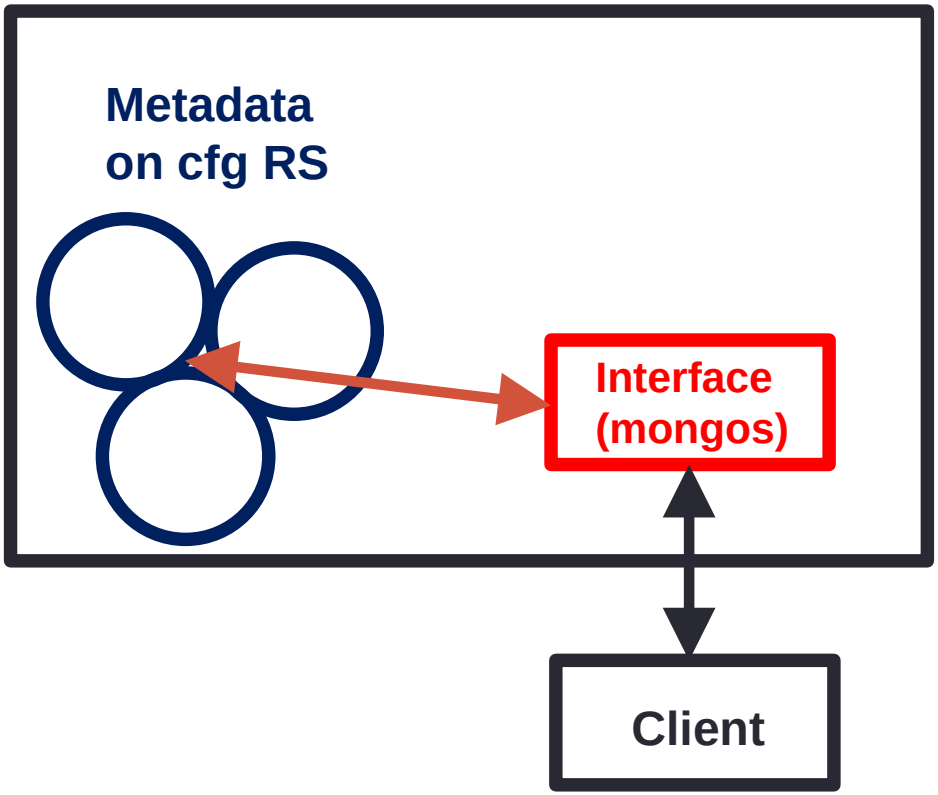
```
2018-10-07T16:47:13.844+0100 I NETWORK  [mongosMain] Starting new replica set monitor for cfg/127.0.0.1:26050
2018-10-07T16:47:13.858+0100 I SHARDING [thread1] creating distributed lock ping thread for process DESKTOP-NQ1MKNN:2701
1:1538927233:3571420696015326107 (sleeping for 30000ms)
2018-10-07T16:47:13.860+0100 I NETWORK  [ReplicaSetMonitor-TaskExecutor-0] Successfully connected to 127.0.0.1:26050 (1
connections now open to 127.0.0.1:26050 with a 5 second timeout)
2018-10-07T16:47:13.861+0100 I NETWORK  [ReplicaSetMonitor-TaskExecutor-0] changing hosts to cfg/127.0.0.1:26050,127.0.0
.1:26051,127.0.0.1:26052 from cfg/127.0.0.1:26050
2018-10-07T16:47:13.861+0100 I SHARDING [ReplicaSetMonitor-TaskExecutor-0] Updating ShardRegistry connection string for
shard config from: cfg/127.0.0.1:26050 to: cfg/127.0.0.1:26050,127.0.0.1:26051,127.0.0.1:26052
2018-10-07T16:47:13.864+0100 I NETWORK  [ReplicaSetMonitor-TaskExecutor-0] Successfully connected to 127.0.0.1:26052 (1
connections now open to 127.0.0.1:26052 with a 5 second timeout)
2018-10-07T16:47:13.865+0100 I NETWORK  [ReplicaSetMonitor-TaskExecutor-0] Successfully connected to 127.0.0.1:26051 (1
connections now open to 127.0.0.1:26051 with a 5 second timeout)
```

Configuration Nodes and Cluster Interface

mongos does not yet know about rs0

Metadata:

- The metadata contains the table of contents of the cluster.
- Once a set of nodes is established as a replica set (e.g., rs0 in our previous example), it can be added to the cluster as a shard.



Configuration Nodes and Cluster Interface

Metadata:

- Process:

1. Connect a client to the shard service (mongos) config db.

```
mongo --port 27011
```

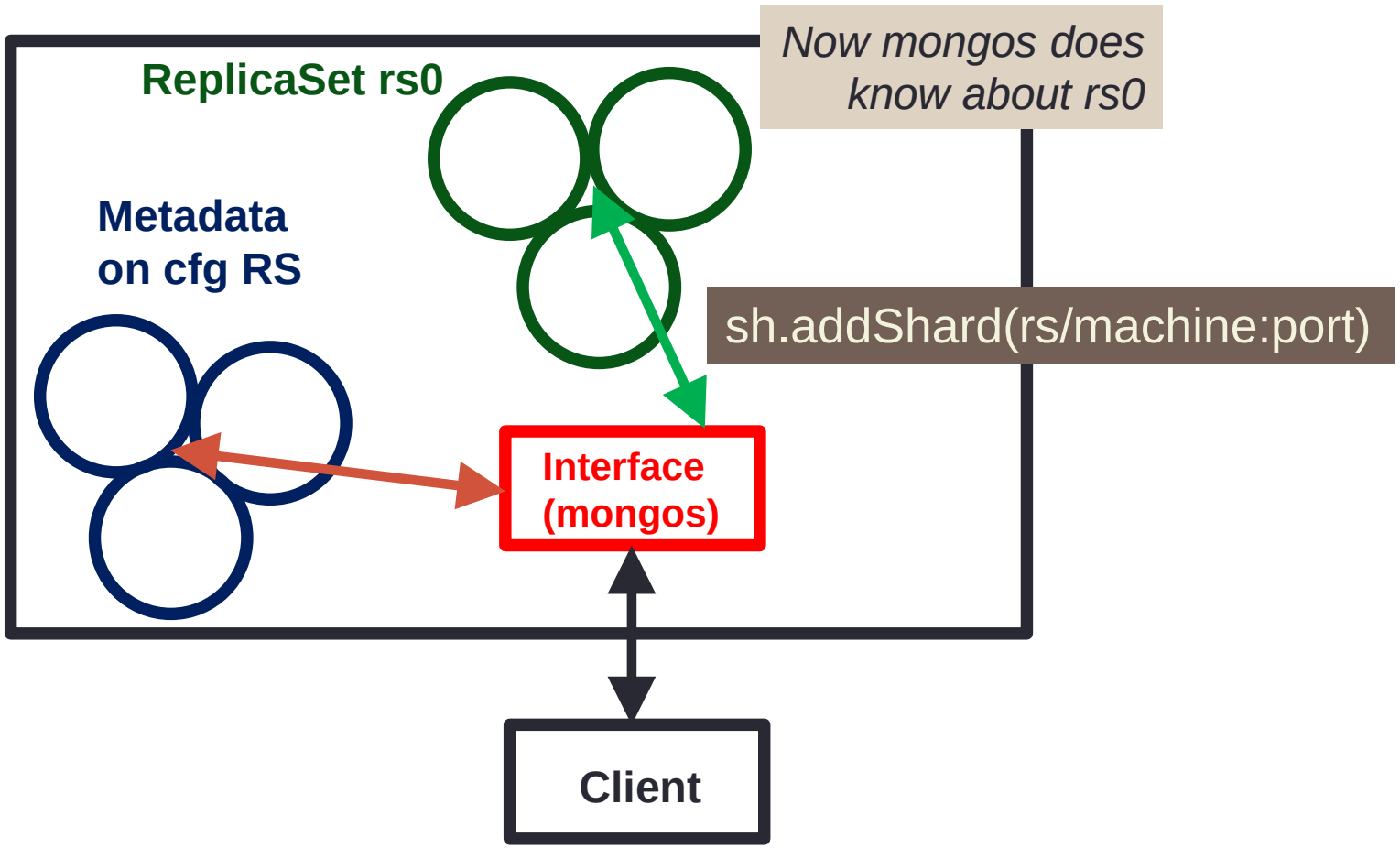
2. Run the commands:

```
use config
sh.addShard("rs0/127.0.0.1:27000")
```

```
mongos> sh.addShard("rs0/127.0.0.1:27000")
{
        "shardAdded" : "rs0",
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1538927573, 2),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1538927573, 2)
}
mongos>
```

Configuration Nodes and Cluster Interface

Metadata:



Configuration Nodes and Cluster Interface

Metadata:

- Thus, the metadata knows how many shards are in the cluster, how many databases, collections, chunks of information and the proper distribution of these chunks of information among the shards.
- Besides that, the config database can be uses to parameterise the size of the chunks and the max difference of number of chunks of a collection placed on different shards.

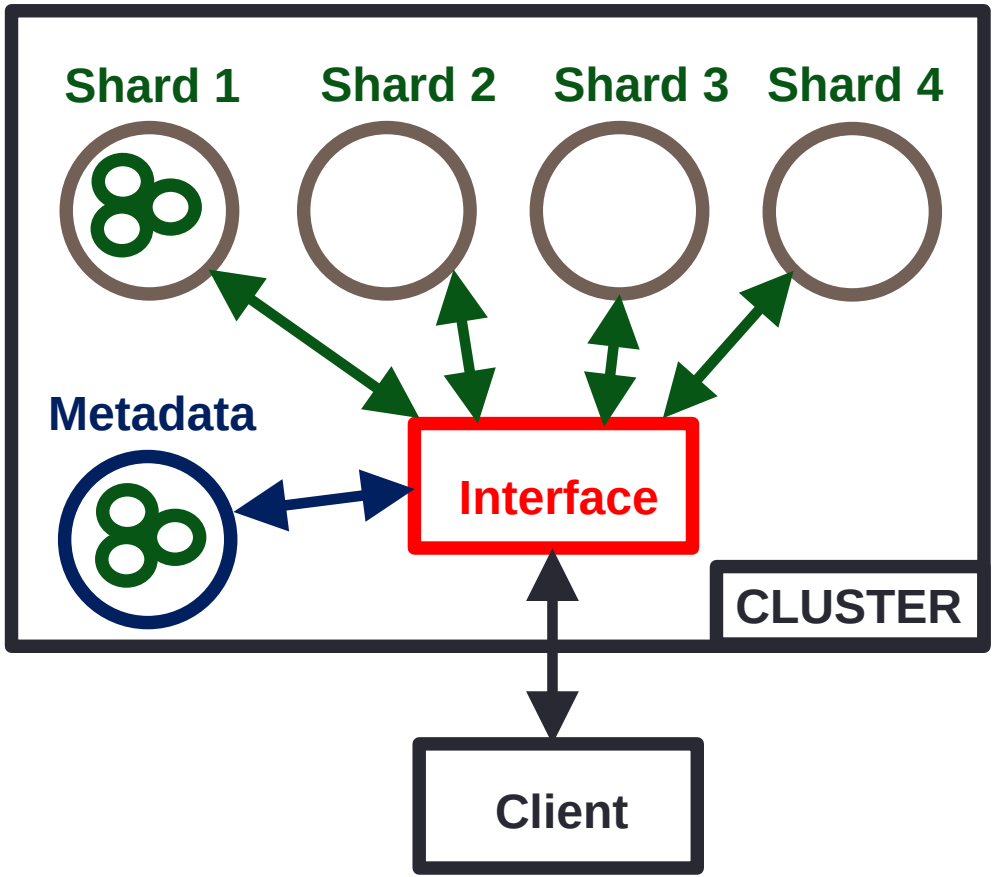
- In our case we set the chunk size (while `use`-ing `config` db) to 1MB with:

```
db.settings.save( { _id:"chunksize", value: 1 } )
```

Configuration Nodes and Cluster Interface

Cluster Interface:

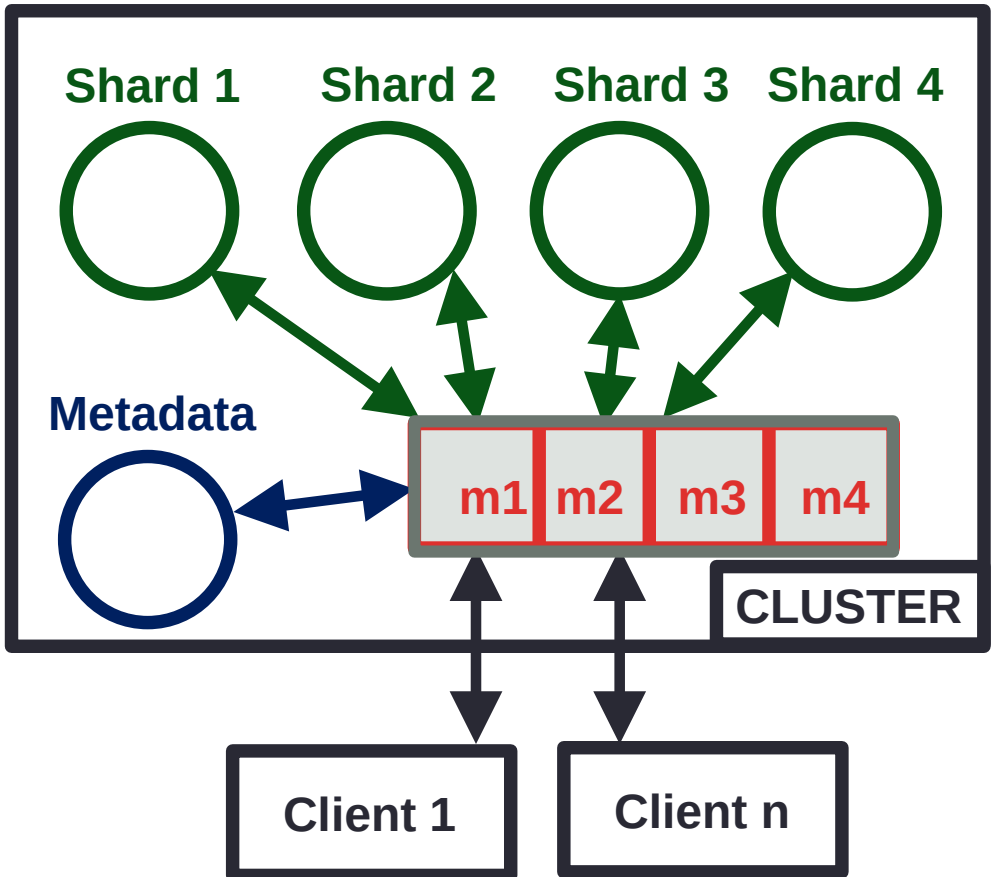
- Note that we have only added 1 shard (the rs0 replicate set).
- As a mongos process is lightweight, it can be executed either on the cluster or the client's side.



Configuration Nodes and Cluster Interface

Cluster Interface:

- We can set up as many mongos processes as we want.
- In our case we will set up four processes, listening to:
  27011
  26061
  26062
  26063.



Next week we will build a cluster (with sharding and replication) for our Tate dataset.

Thank you!