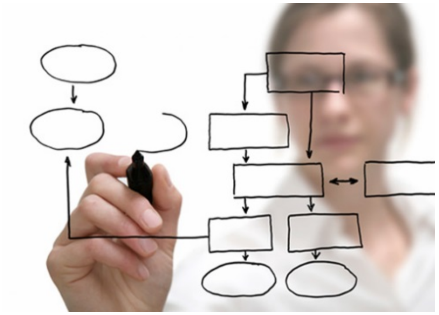
CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Computing@CIT



Distributed Data Management

Lecture 10: Aggregation Framework

Outline

1. Aggregation Framework: Motivation.
2. Aggregation Commands: Examples.
3. Aggregation Pipeline.

Outline

1. Aggregation Framework: Motivation.
2. Aggregation Commands: Examples.
3. Aggregation Pipeline.

Aggregation Framework: Motivation

- Think of assignment 2 part 1: Our cluster is local (i.e. all nodes are in our machine), but we can think as if the data was truly distributed among 3 DCs placed in London, Amsterdam and New York.

ed
es
Canada
United States
Mexico
PACIFIC
Guatemala
Honduras
Nicaragua
ATLANTIC
OCEAN
Iceland
Norway
Sweden
Finland
United Kingdom
Ireland
Den.
Neth.
Ger.
Pol.
Estonia
Latvia
Belarus
Ukraine
France
Rom.
Bulgaria
Spain
Portugal
Italy
Greece
Turke
Tunisia
Lebanon
Israel
Morocco
Algeria
Libya
Egypt
Western Sahara
Mauritania
Mali
Niger
Chad
Eritrea
Sudan
Senegal
Guinea-Bissau

Aggregation Framework: Motivation

- You might be more comfortable with Python coding that Mongo/JavaScript, so might think:

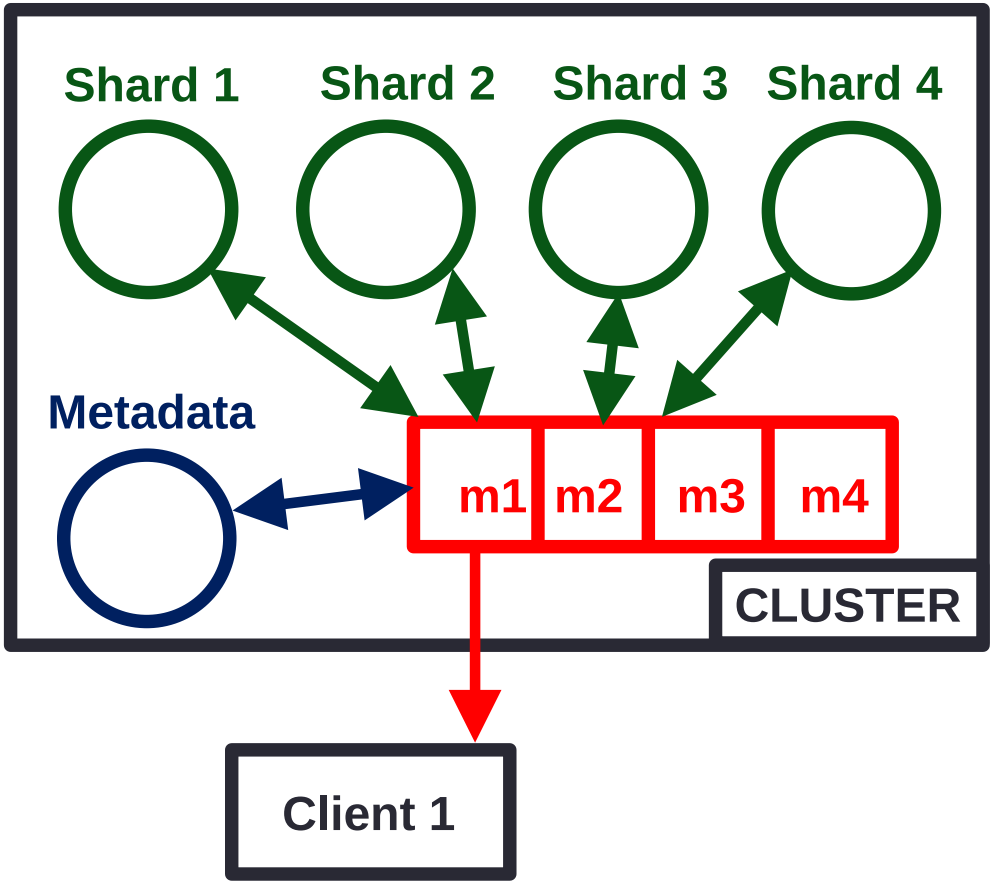
"Ok, as I know more Python than MongoDB/JavaScript, I'm going to query the collection for <u>all documents</u>. Once I get the documents transferred to my Python program (each document as a Python dictionary variable), I will process the query using my Python knowledge".

- **Why is this approach terrible?**
  - You are replicating the former storage-cluster & compute-cluster model we are trying to escape from.
  - You are moving all data from the cluster to your machine for its further processing.

Aggregation Framework: Motivation

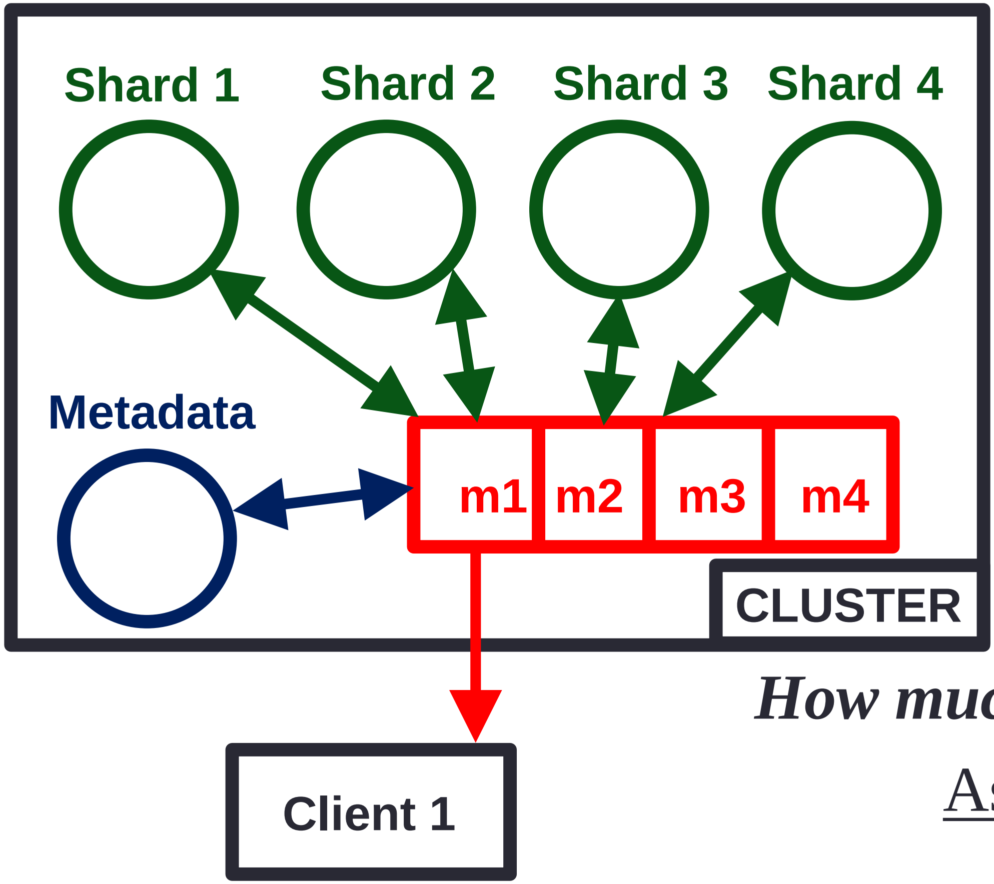
- ❑ Maxim to follow:

When querying a cluster, try to reduce to the minimum the data transferred back from the cluster to the client!



Aggregation Framework: Motivation

□ Think that, if the cluster is distributed, this data might be coming back to your client (e.g., computer, mobile phone) transferred from a DC placed somewhere in the world!



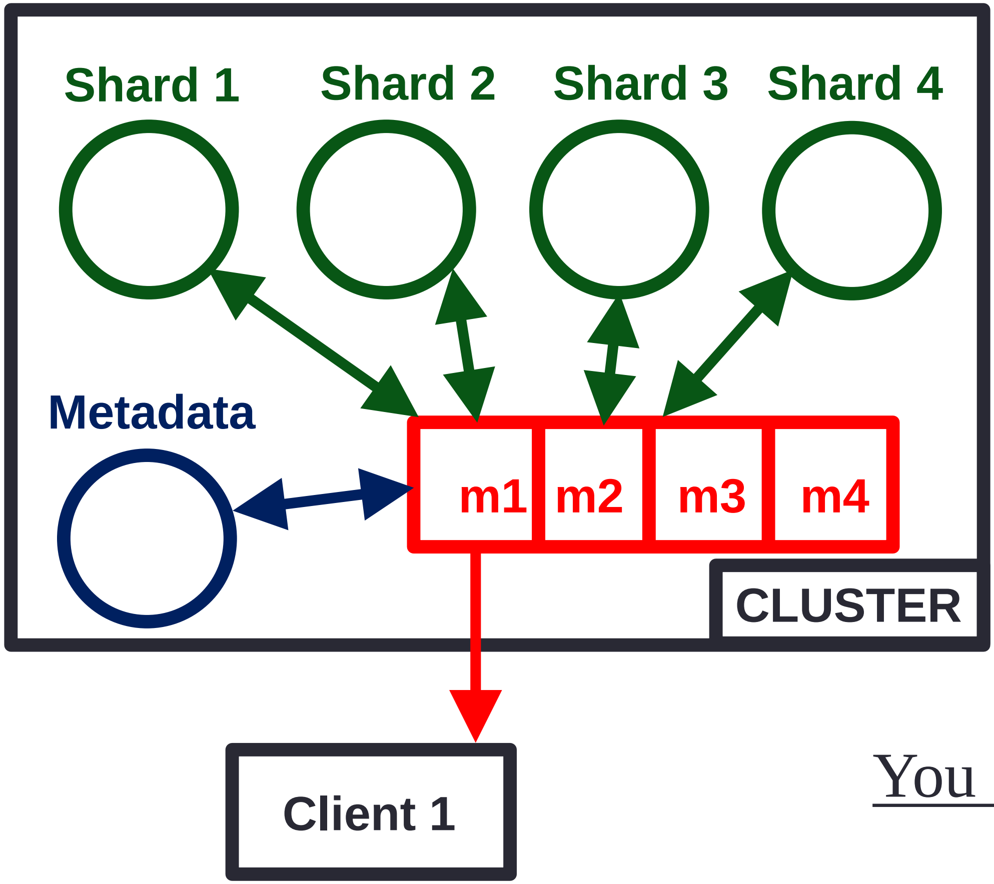
Greenland
Canada
United States
North Pacific Ocean
North Atlantic Ocean
South Pacific Ocean
South Atlantic Ocean
Southern Ocean
Brazil
Russia
China
Australia
Google

How much data do you want to transfer?

<u>As little as possible!</u>

Aggregation Framework: Motivation

- But, the data being transferred is (nothing more and nothing less) the result of the query being processed!



You queried the cluster?

<u>You got your reply transferred back!</u>

Aggregation Framework: Motivation

❑ **Key Concept**:

There is no dilemma here:

- If you need to perform a query, do it!
- And, as much data as you have to bring back, bring it back!

❑ That said:

- Try to make your query as precise as possible, so that it only gets as a result exactly what you need. Working this way:
  1. All the work is done by the cluster's shard server, not by your machine.
  2. The information being transferred is minimised, and so is the data transfer overhead.
  3. The post-processing of the result is minimal, as you are already getting it in the format you want.

Aggregation Framework: Motivation

❑ **Goals of today:**

1. Extend our MongoDB/JavaScript knowledge with the aggregation framework's supported commands, so as to make precise queries.

2. Learn about the concept of pipelines in the aggregation framework, so as to decompose an initial complex query into a set of subqueries [q_1, q_2, …, q_n] (whose global action is the same as our original complex query).

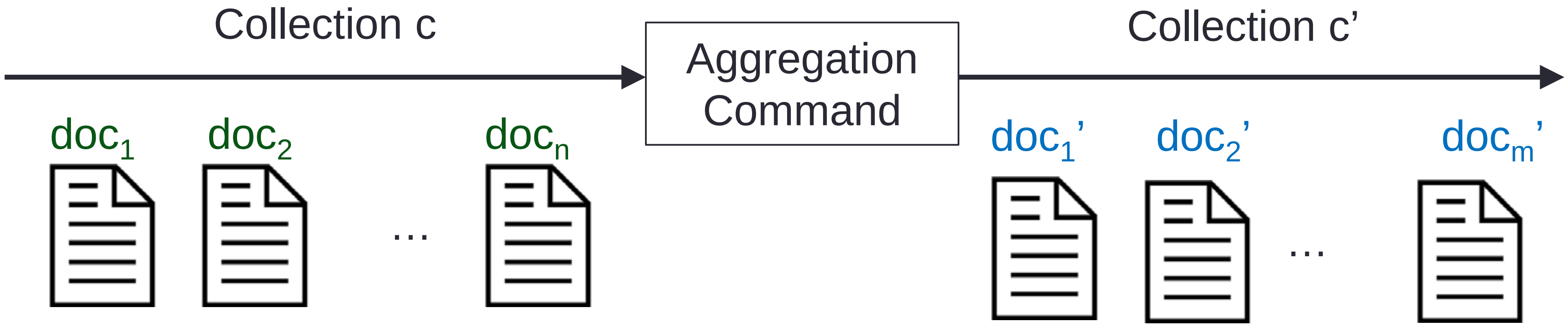
3. Trigger the execution of the set of queries [q_1, q_2, …, q_n], so they are processed in sequence within the proper shard of the cluster (before the final result is transferred back to the client).

Outline

1. Aggregation Framework: Motivation.
2. Aggregation Commands: Examples.
3. Aggregation Pipeline.

Aggregation Commands: Examples

- ❑ Aggregations are operations that process data records and return computed results.
- ❑ An aggregation command takes a collection c of n ≥ 0 documents as an input and returns a new collection c’ of m ≥ 0 documents as a result of the action taking place.



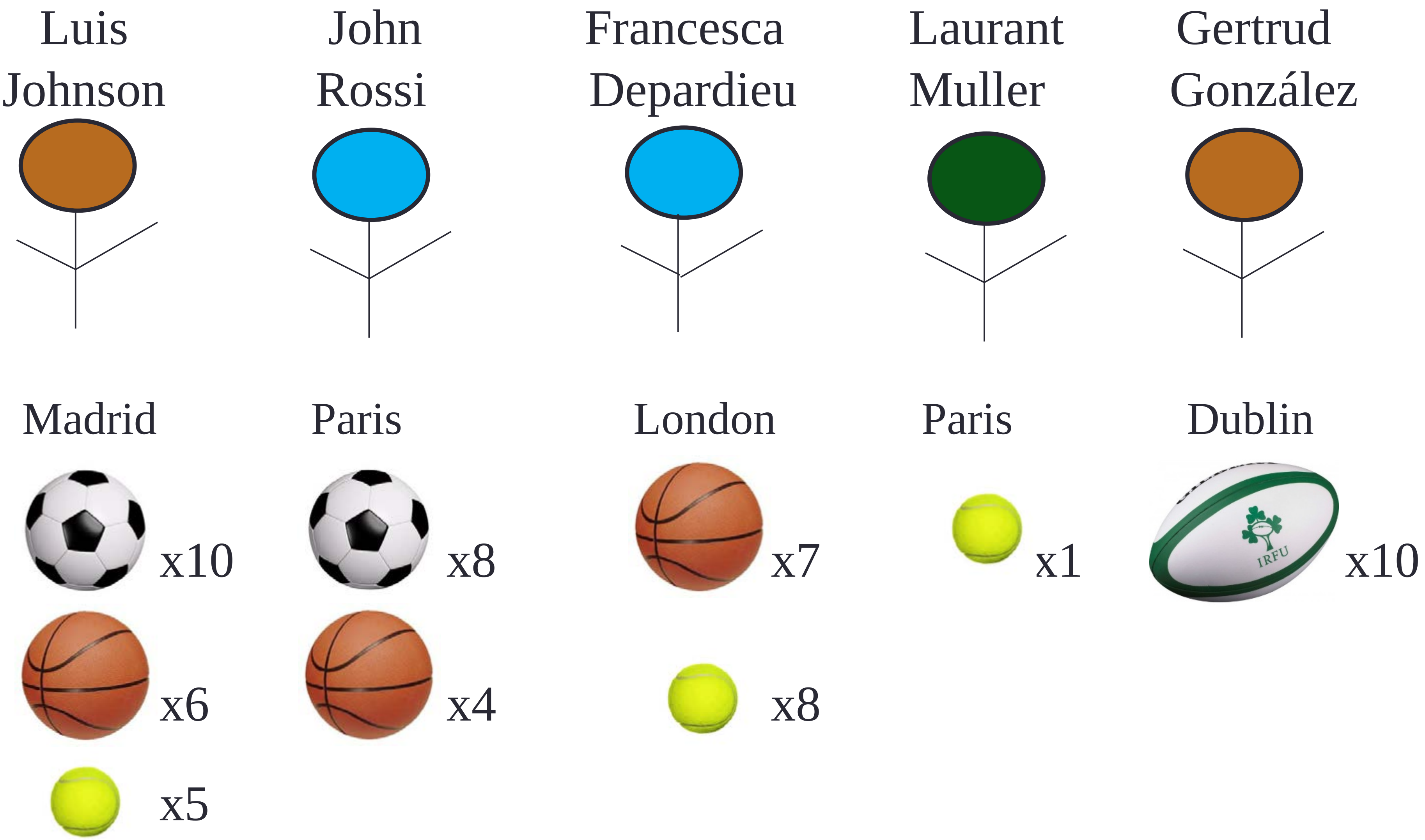
Aggregation Commands: Examples

- ❑ We are going to use a tiny collection of just 5 documents to explain and test several commands.
- ❑ The collection can be found in the Week 06 folder in Blackboard in *my_collection.json*.

The structure of each of the 5 documents is the following:

```
{  "identifier" : {  "name" : String, "surname": "String" },
   "eyes" : String,
   "city" : String,
   "likes" : [ {"Sport" : String, "score" : Integer}, {…}, …, {…} ]  }
```

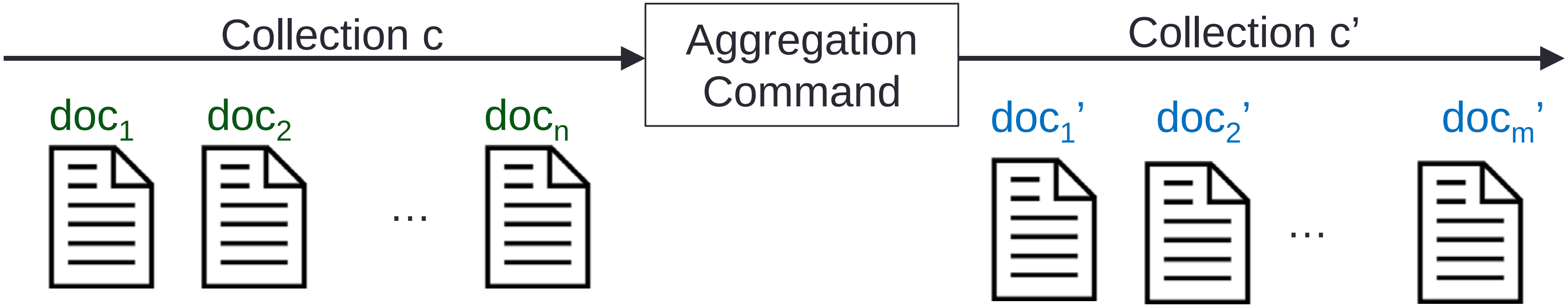
Aggregation Commands: Examples



Luis Johnson	John Rossi	Francesca Depardieu	Laurant Muller	Gertrud González
Madrid	Paris	London	Paris	Dublin
⚽ x10	⚽ x8	🏀 x7	🎾 x1	🏉 x10
🏀 x6	🏀 x4	🎾 x8		
🎾 x5				

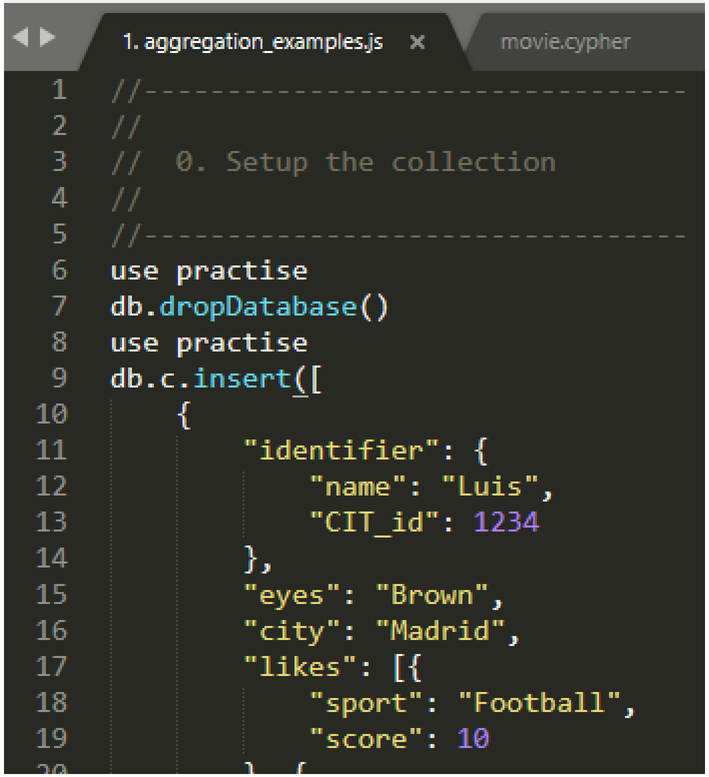
Aggregation Commands: Examples

- ❑ To study the command examples with Python (using Pymongo), use the file *aggregation_examples.py* in the Week 06 folder.
  - It contains a **my_fun(option)** function that does all the job:
    - ✓ Creates the client, a database "practise" and the collection "c" with the 5 documents and calls **aggregation_command(db, option)**.
    - ✓ This function triggers the query associated to the command option being passed and returns as a result the new collection c' obtained.
    - ✓ Finally, the collection c' obtained is printed.



Aggregation Commands: Examples

- ❑ To study the command examples with an explicit mongo.exe client (using JavaScript), use the file “1. aggregation_examples.js” in Week 06 folder.
  - Run the commands under the label “0. Setup the Collection” to create the database “practise”, the collection “c” and insert the 5 documents.
  - Once this is done, you can try the rest of the commands.



```
//---------------------------------
//
//  0. Setup the collection
//
//---------------------------------
use practise
db.dropDatabase()
use practise
db.c.insert([
    {
        "identifier": {
            "name": "Luis",
            "CIT_id": 1234
        },
        "eyes": "Brown",
        "city": "Madrid",
        "likes": [{
            "sport": "Football",
            "score": 10
        }, {
```

Aggregation Commands: Examples

Command 1: Match

- ❑ Semantics: Filters the documents of c, passing to c’ only the ones matching the specified condition(s).
  - o The documents of c returned in c’ are not modified.
- ❑ Prototype: { $match: { <query> } }
- ❑ Examples:
  - o Option = 11: Match with single condition.
  - o Option = 12: Match with multiple conditions.
  - o Option = 13: Match can return an empty collection c’.

Aggregation Commands: Examples

Command 2: Group

- ❑ Semantics: Groups documents by some specified expression and outputs a document for each distinct grouping.
  - o The documents of c’ are not related to the ones of c.
- ❑ Prototype: { $group: { _id: <expression>,

  <field$_1$>: { <accumulator1> : <expression1>},

  … ,

  <field$_n$>: { <accumulator1> : <expression1>}

  }

  }

Aggregation Commands: Examples

Command 2: Group

- ❑ Examples:
  - o Option = 21: Grouping by a single field.
  - o Option = 22: Grouping by a single field and create new field(s).
  - o Option = 23: Grouping by several fields and create new field(s).

Aggregation Commands: Examples

Command 3: Sort

- Semantics: Sorts the documents of c by a specified parameter(s).
  - The documents of c returned in c’ are not modified.
- Prototype: { $sort: { <field$_1$>: <sort order>

  …

  <field$_n$>: <sort order>

  }

  }

Aggregation Commands: Examples

Command 3: Sort

- ❑ Examples:
  - o Option = 31: Sort based on a single field.
  - o Option = 32: Sort based on several fields.

Aggregation Commands: Examples

Command 4: Limit

- ❑ Semantics: Limits the number of documents of c being returned.
  - o The documents of c returned in c’ are not modified.
- ❑ Prototype: { $limit: { <positive integer> } }
- ❑ Examples:
  - o Option = 41: Limit the number of documents returned.

Aggregation Commands: Examples

Command 5: Count

- ❑ Semantics: Count the documents of a collection.
  - It is not a proper aggregation command, but it can be *simulated*.
- ❑ Prototype:

```
{ $group: { _id: null,
            count : { $sum : 1 }
          } }
```

- ❑ Examples:
  - Option = 51: Use count command directly.
  - Option = 52: Simulate it to be an aggregation command.

Aggregation Commands: Examples

Command 6: Project

- ❑ Semantics: Modify the fields of the documents.
  - o All documents of c are modified and returned in c’.
- ❑ Prototype: { $project: { <specifications> } }
- ❑ Examples:
  - o Option = 61: Restrict the fields of the document.
  - o Option = 62: Restrict and skip the ObjectId field.
  - o Option = 63: Restrict and add new fields (belonging to a subdocument).
  - o Option = 64: Restrict and add new fields as a normal field.
  - o Option = 65: Restrict and add new fields.

Aggregation Commands: Examples

Command 7: Unwind

- ❑ Semantics: Deconstructs an array and creates a new document for each array value (exploding them out).
  - o All documents of c' are related to c in the sense that are the same documents, but for the array field, for which they have now a single value.
- ❑ Prototype: { $unwind: { <field> } }
- ❑ Examples:
  - o Option = 71: Gets the length of an array field.
  - o Option = 81: Restrict the fields of the document using unwind.

Outline

1. Aggregation Framework: Motivation.
2. Aggregation Commands: Examples.
3. Aggregation Pipeline.

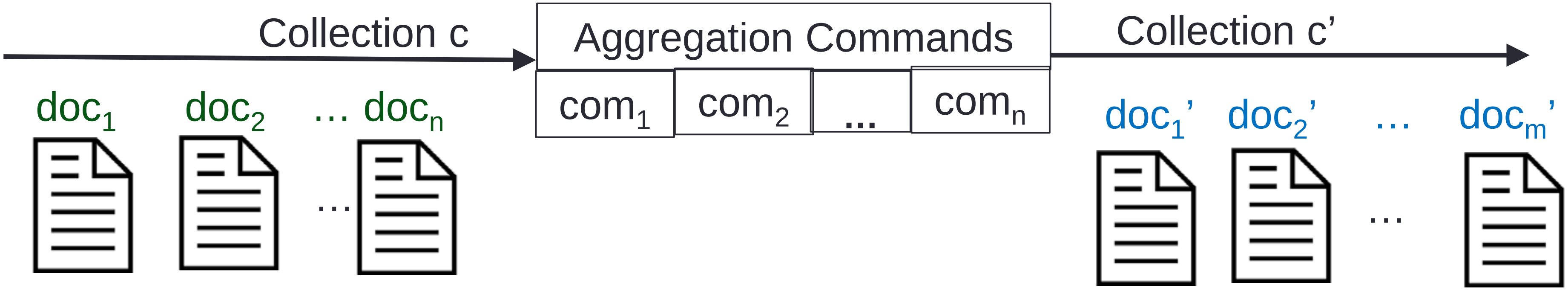
Aggregation Pipeline

- ❑ Key Concept:

Besides the commands provided to query a collection…

The most important concept provided by the aggregation framework is the concept of an **aggregation pipeline**.

- ❑ An aggregation pipeline p = [com_1, com_2, …, com_n] receives a collection c as input and applies *n* aggregation commands to it, so as to generate a new collection c’ as a result.



Collection c

Aggregation Commands

com_1 | com_2 | … | com_n

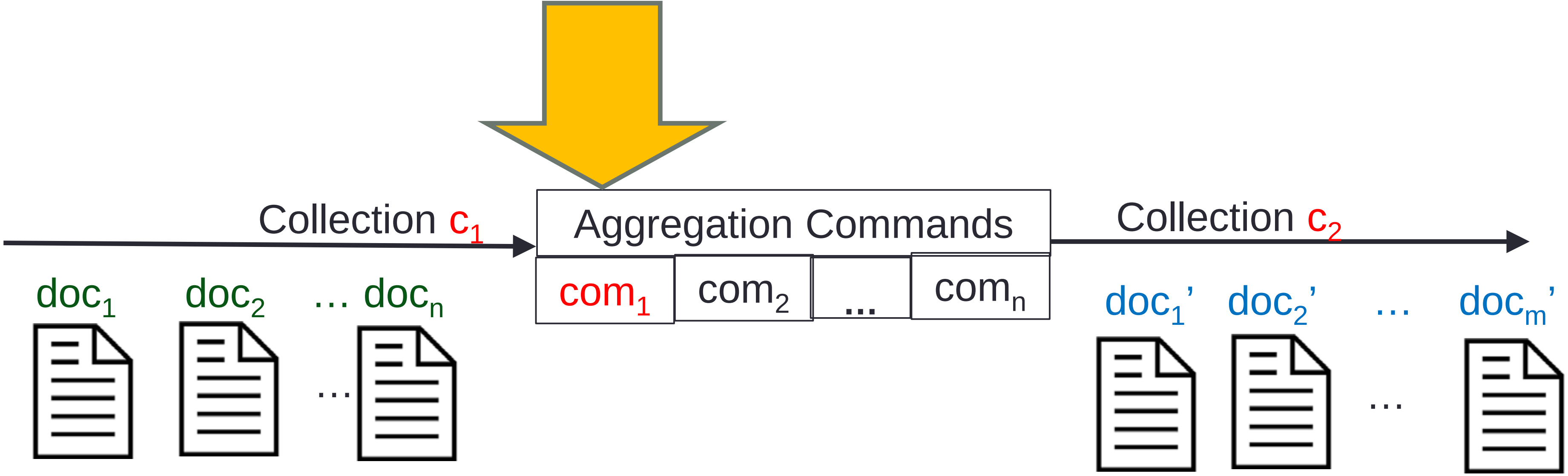
Collection c’

doc_1 doc_2 … doc_n

doc_1’ doc_2’ … doc_m’

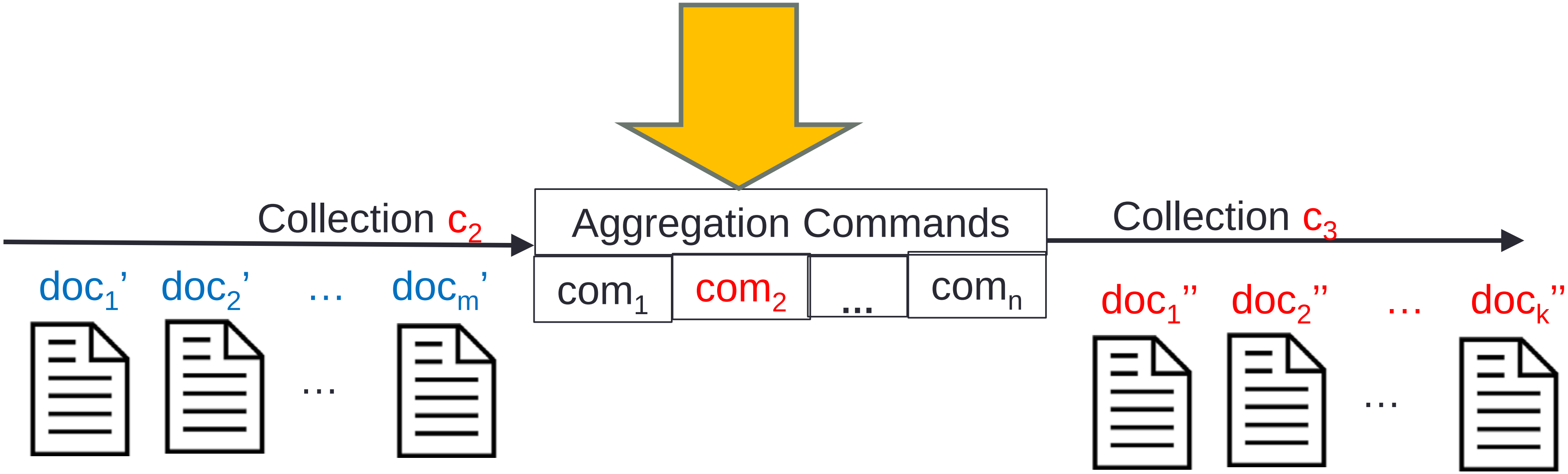
Aggregation Pipeline

- The commands work in order:
  - Each command com_i receives an input collection c_i from the previous command.
  - It modifies c_i by performing the aggregation command, generating as a result a new output collection c_{i+1}.
  - The collection c_{i+1} is passed as input to the next command com_{i+1}.



Aggregation Pipeline

- ❑ The commands work in order:
  - o Each command com_i receives an input collection c_i from the previous command.
  - o It modifies c_i by performing the aggregation command, generating as a result a new output collection c_{i+1}.
  - o The collection c_{i+1} is passed as input to the next command com_{i+1}.



Collection c_2

doc_1' doc_2' … doc_m'

…

Aggregation Commands

com_1 com_2 … com_n

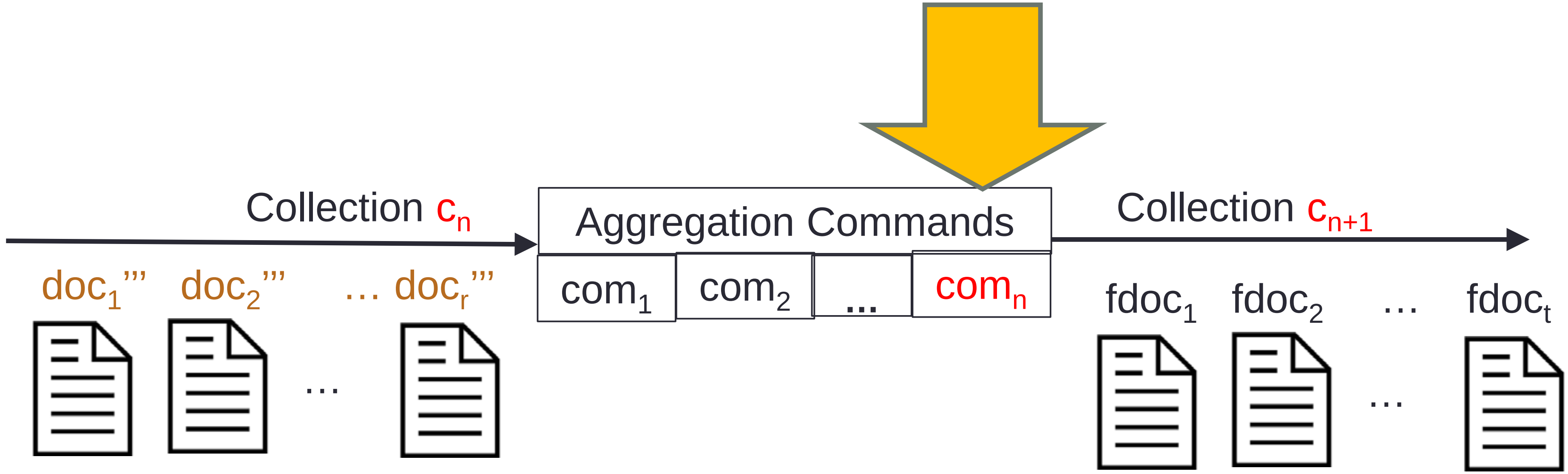
Collection c_3

doc_1'' doc_2'' … doc_k''

…

Aggregation Pipeline

- ❑ The commands work in order:
  - o Each command com_i receives an input collection c_i from the previous command.
  - o It modifies c_i by performing the aggregation command, generating as a result a new output collection c_{i+1}.
  - o The collection c_{i+1} is passed as input to the next command com_{i+1}.



Aggregation Pipeline

- ❑ By using aggregation pipelines, we can decompose a very complex query into a sequence of *n* simpler aggregation commands.
- ❑ All the commands [com_1, com_2, …, com_n] of the pipeline will be executed in the shards of the cluster for which the query is relevant.
  - o Intermediate collections c_1, c_2, …, c_n will be created *on the fly* within the cluster to perform the different commands [com_1, com_2, …, com_n].
  - o Once an intermediate collection c_i is no longer needed, it is removed.
  - o Only the final generated collection c’ is transferred back to us as clients (none of the intermediate ones).

Aggregation Pipeline

Aggregation pipeline: Example.

- ❑ Consider this complex query: Find the sport of the collection which is liked by the highest amount of points.
  - o *Definitely, not the simplest query ever!*

- ❑ Approach: Tackle this complex query by a sequence of simpler steps, each step applying an aggregation command.

Aggregation Pipeline

Aggregation pipeline: Example.

Step 1

- ❑ How can we access the sports liked by the people?
  - o They are a bit hidden, as multiple sports can be under a single document (imagine a person liking n > 1 sports).
  - o Solution: Use the unwind command to get a document for every person and sport. Now we will get each document with a single sport, but now there are so many documents!

Aggregation Pipeline

<u>Aggregation pipeline: Example.</u>

<u>Step 2</u>

- ❑ Now we have each sport in a single document, which is helpful. But we have more than 5 documents!
  - o Let’s try to reduce these documents to the minimum. They are too complex. We are just interested in:
    - ▪ The field `sport` (to know which sport we are talking about).
    - ▪ The field `points` (to know how much each person likes the sport).

- o Solution: Use the <u>project</u> command to reduce the number of fields in our documents to just these two fields.

Aggregation Pipeline

Aggregation pipeline Example.

Step 3

- ❑ Now we have many documents, but all of them are very clean.
  - o Let’s group them by sport, so that we get just one document per sport:
    - ▪ We want an extra field in our documents that is the total sum of points for the grouped documents.

- o Solution: Use the group command to group the documents.

Aggregation Pipeline

Aggregation pipeline Example.

Step 4

- ❑ Now we have one document per sport, and with the total amount of points.
  - o Let’s sort them by points in decreasing order, so that we get them by preference, then by person.

- o Solution: Use the sort command to sort the documents.

Aggregation Pipeline

Aggregation pipeline Example.

Step 5

- ❑ Now we just want the preferred sport, so we limit the collection to just one document.

- o Solution: Use the limit command to get just the first document.

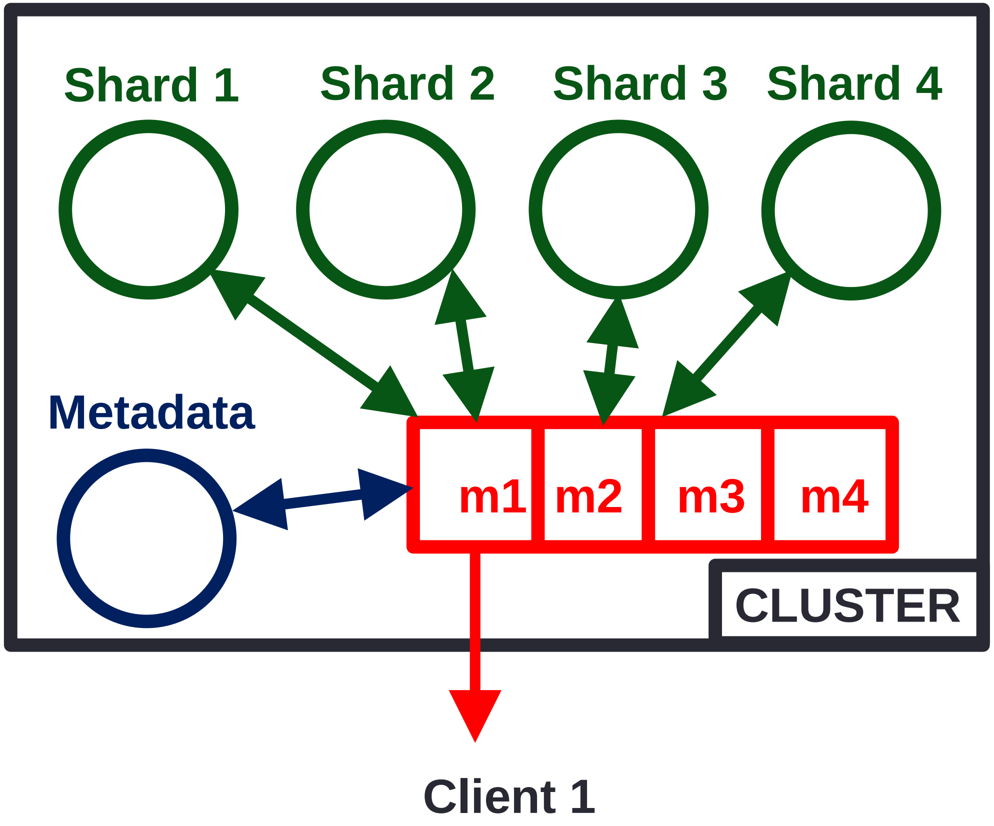
Aggregation Pipeline

Aggregation pipeline Example.

- ❑ And, voilà, only a single document is transferred back to the client with the result of the aggregation pipeline.

```
{ “_id” : “Football”,
   “points” : 18 }
```

- ❑ This is implemented by option 91 in aggregation_examples.js



Outline

1. Aggregation Framework: Motivation.
2. Aggregation Commands: Examples.
3. Aggregation Pipeline.

Thank you for your attention!