MIDDLEWARE TECHNOLOGIES FOR DISTRIBUTED SYSTEMS

A.A. 2018/2019

POLITECNICO
MILANO 1863

Matteo Moreschini
Alessio Russo Introito
Tommaso Scarlatti

Overview

1. Projects definition and data preprocessing
2. A simplified version of Twitter using **Kafka**
3. Big-data platform with **Akka**
4. Parallel K-means with **OpenMP** and **MPI**
5. Results and conclusions

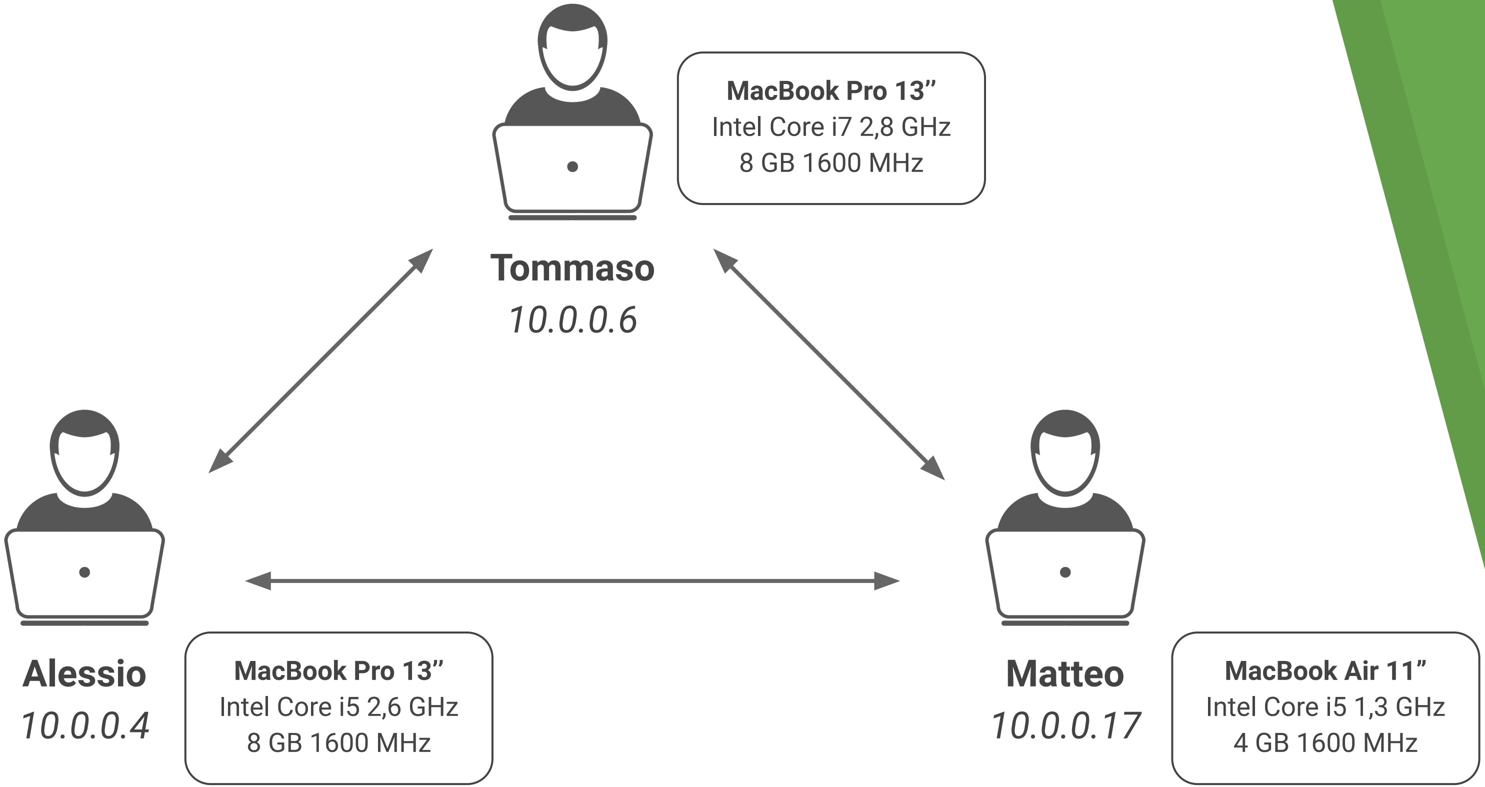
1.

PROJECTS DEFINITION

Internal organization

- We split the work in **two phases**:
  - Each member worked on a different project to get acquainted with the technology
  - All members worked on the same project one by one
- We tested our deliverables first locally and then in a **wireless ad-hoc network** using static ip addresses

LAN Overview



Tommaso
10.0.0.6

MacBook Pro 13”
Intel Core i7 2,8 GHz
8 GB 1600 MHz

Alessio
10.0.0.4

MacBook Pro 13”
Intel Core i5 2,6 GHz
8 GB 1600 MHz

Matteo
10.0.0.17

MacBook Air 11”
Intel Core i5 1,3 GHz
4 GB 1600 MHz

Three technologies, three languages…

1. Apache Kafka (Python 3.7)
2. Akka (Java 8)
3. OpenMP / MPI (C++ 11)

...One dataset

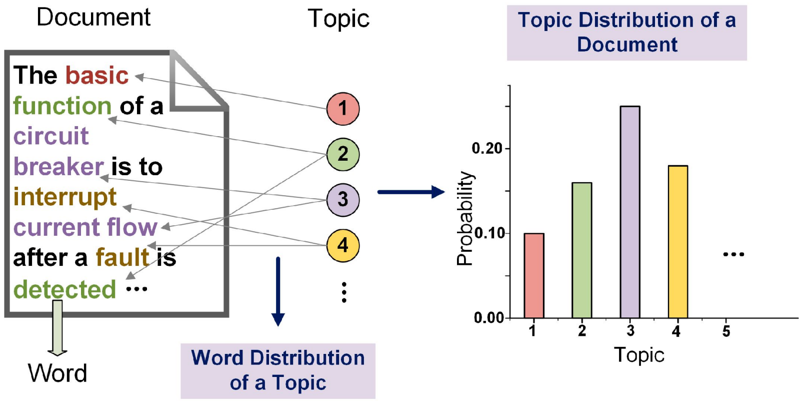
- Retrieved recent **tweets** from (real) Twitter with *Twython* library
- Cleaned text, removed duplicates and retweets
- Tweets used as **input source** for the three projects in different ways

Tweets generation (Kafka)

- Real tweets are used to publish “simplified” tweets
- Fields avaiable:
  - Author
  - Content
  - Tag (#)
  - Mention (@)

Latent Semantic Analysis (Akka)

- Trained a LDA (Latent Dirichlet Allocation) model to automatically extract topics
- Label each tweet with the topic with the highest probability


Document
Topic
Topic Distribution of a Document
The basic function of a circuit breaker is to interrupt current flow after a fault is detected …
1
2
3
4
⋮
Word
Word Distribution of a Topic
Probability
0.00
0.10
0.20
1
2
3
4
5
…
Topic

Latent Semantic Analysis (Akka)

- Each tweet has been assigned to a topic using LDA
- The following 12 topics have been extracted:

```
exercise       895
cooking        808
theatre        744
music          731
animal         643
painting       614
drawing        578
culture        547
health         453
sport          453
history        323
photography    260
```

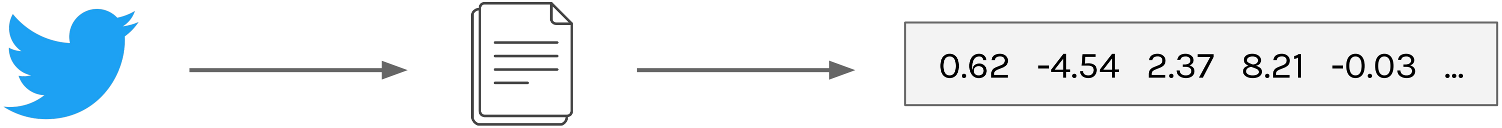
Key-value pairs (Akka)

- Each operator processes (key, value) pairs where both key and value are strings
- **Key** = topic
- **Value** = text

topic	text
cooking	Stuck between cooking and buy something...
music	Why is there only 4 songs on..

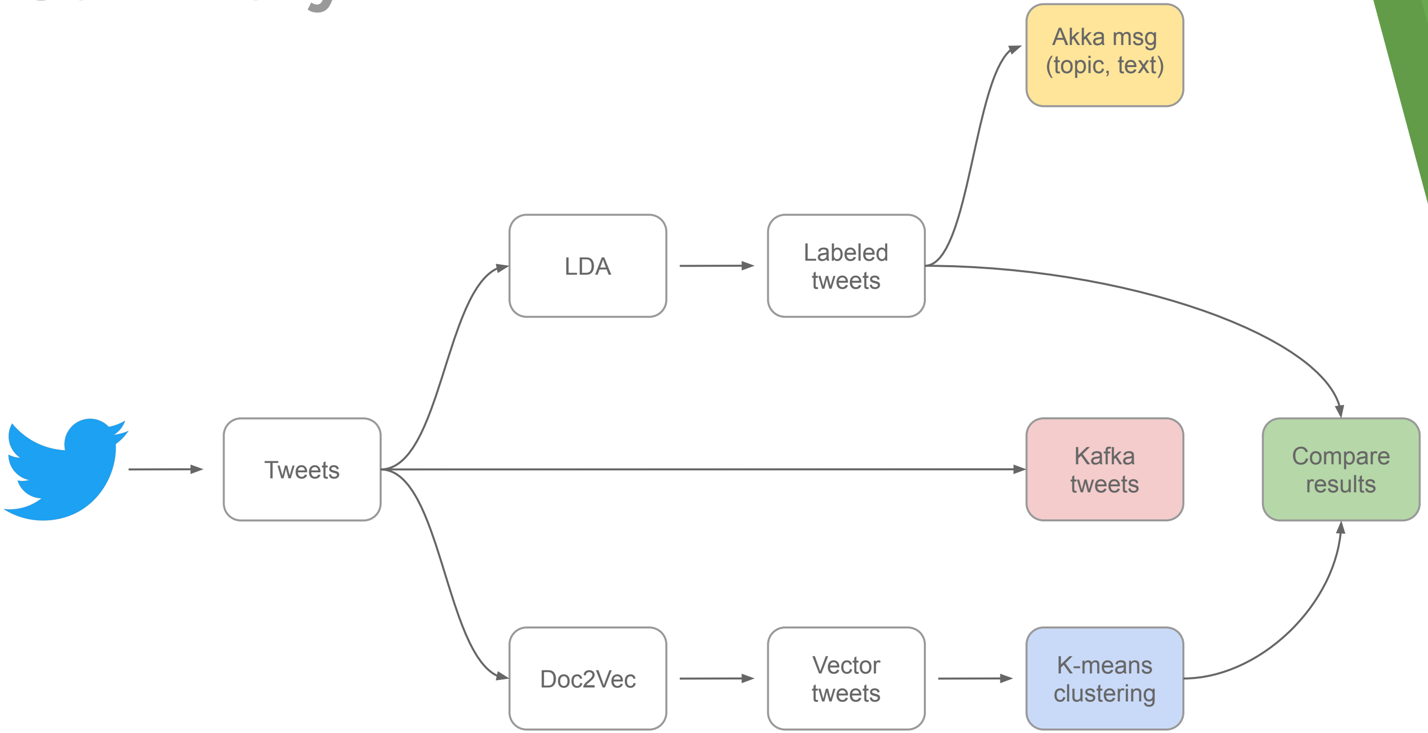
Doc2Vec (K-means clustering)

- Map each tweet in a vector of n dimensions
- Performed with **Facebook InferSent**, a sentence embeddings method for English sentences
- 7K tweets with 4096 dimensions



0.62 -4.54 2.37 8.21 -0.03 ...

Summary


Tweets
LDA
Labeled tweets
Akka msg (topic, text)
Kafka tweets
Compare results
Doc2Vec
Vector tweets
K-means clustering

Literature review

- LDA vs Doc2vec clustering
- Previous researches focused on:
  - Compare their performances in a topic recognition task
    [M. Campr, K. Ježek, *Comparing Semantic Models for Evaluating Automatic Document Summarization*. In: Král P., Matoušek V. (eds) Text, Speech, and Dialogue. TSD **2015**]
  - Combine them in a two-steps approach
    [M. Alhawarat and M. Hegazi, *Revisiting K-Means and Topic Modeling, a Comparison Study to Cluster Arabic Documents*, in IEEE Access, **2018**]

2.

SIMPLIFIED TWITTER USING KAFKA

github.com/tmscarla/**kafka-twitter**

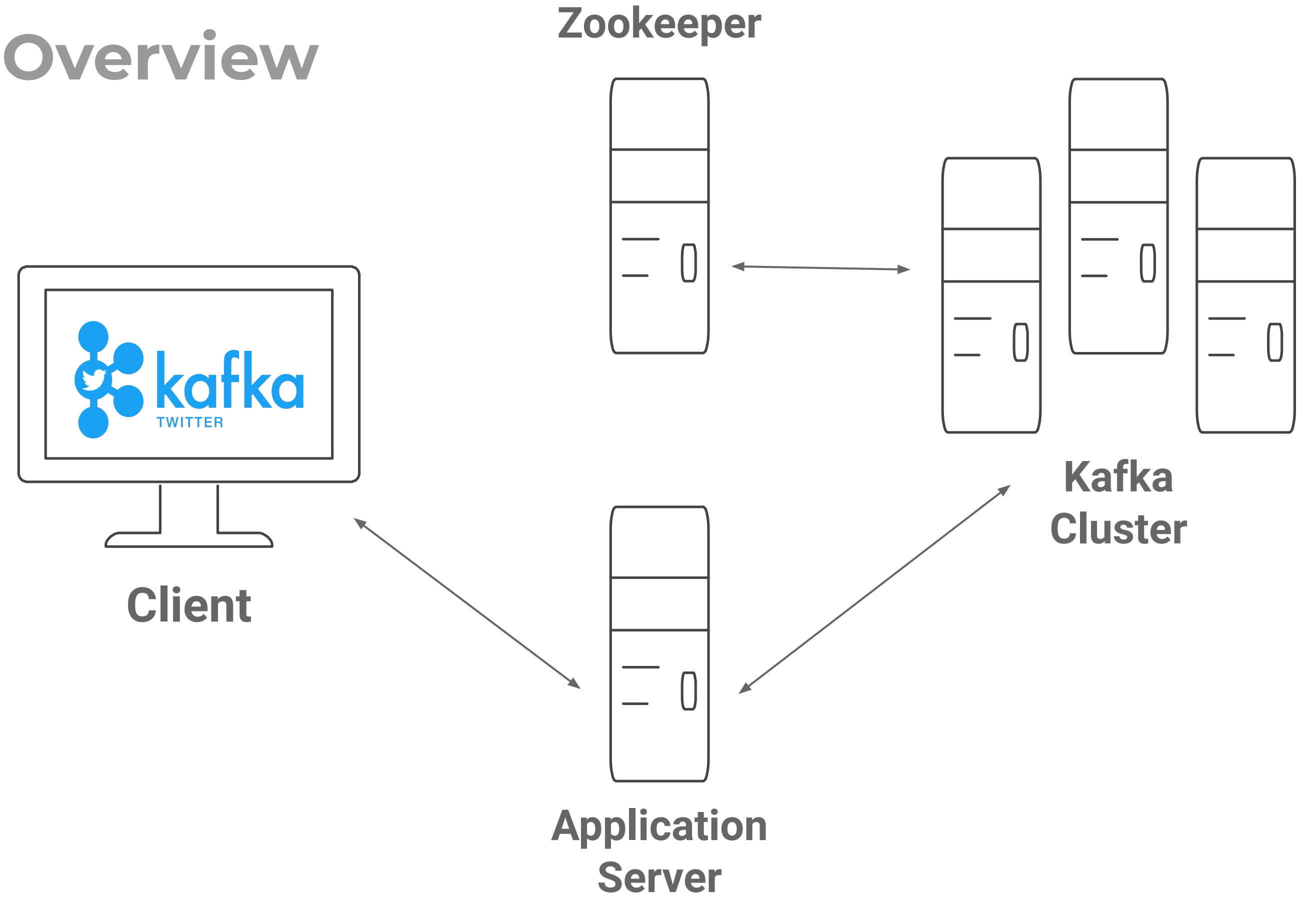
Scope

- Simplified version of Twitter: streaming message application using vanilla Kafka
- RESTful communication
- Basic operations:
  - Subscribing
  - Posting
  - Reading: batch/streaming
  - Filtering messages

Kafka Twitter

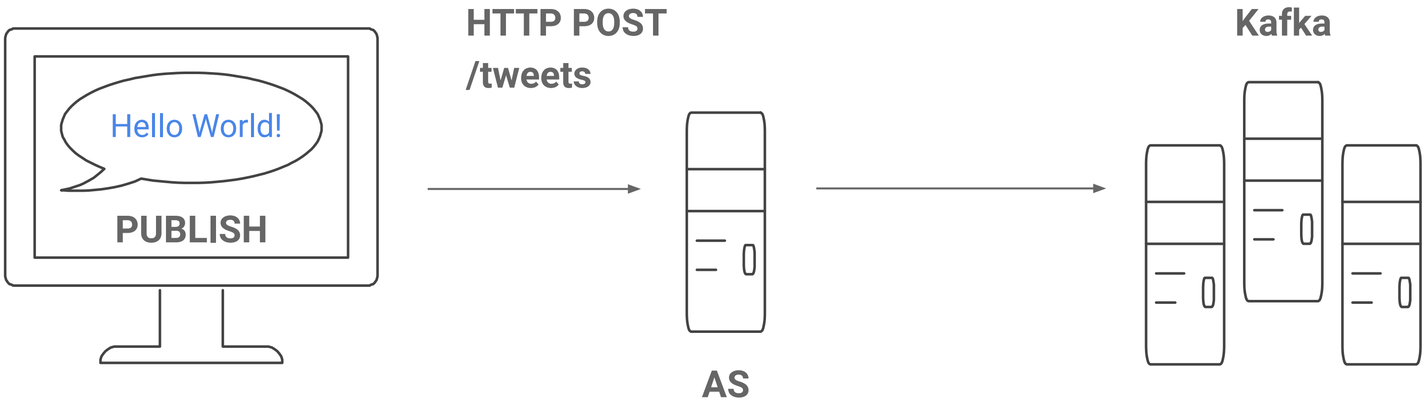
- **Python**
  (3.7.4)
- **confluent-kafka**
  (0.11.6)
- **Tkinter**
  (8.5)
- **Flask**
  (1.1.1)

Overview


Zookeeper
kafka
TWITTER
Client
Kafka
Cluster
Application
Server

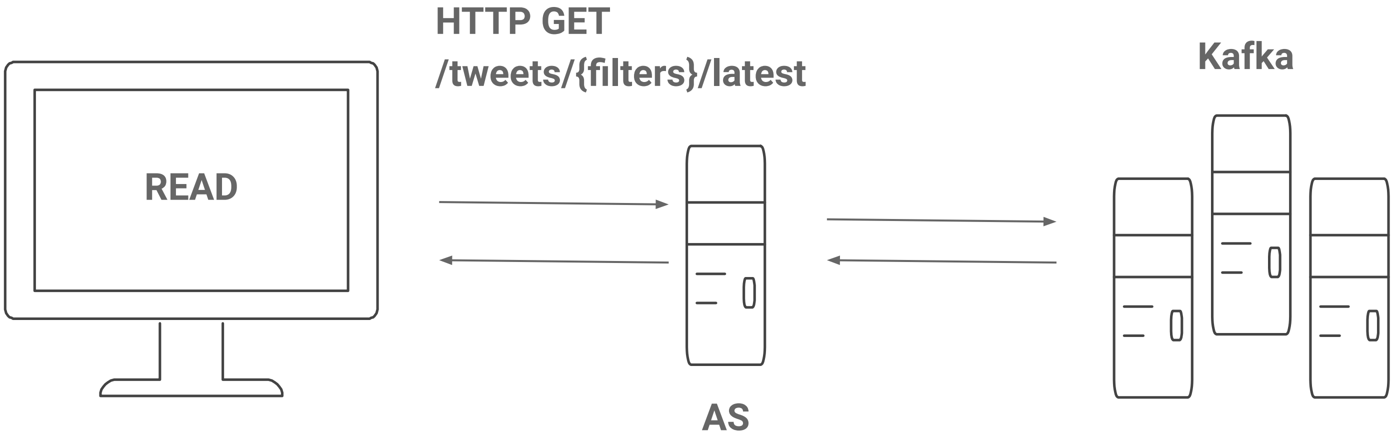
Write

- **AvroProducer**: Avro serialization (avro 1.9.0)
- **tweet_schema.avsc**: author, content…



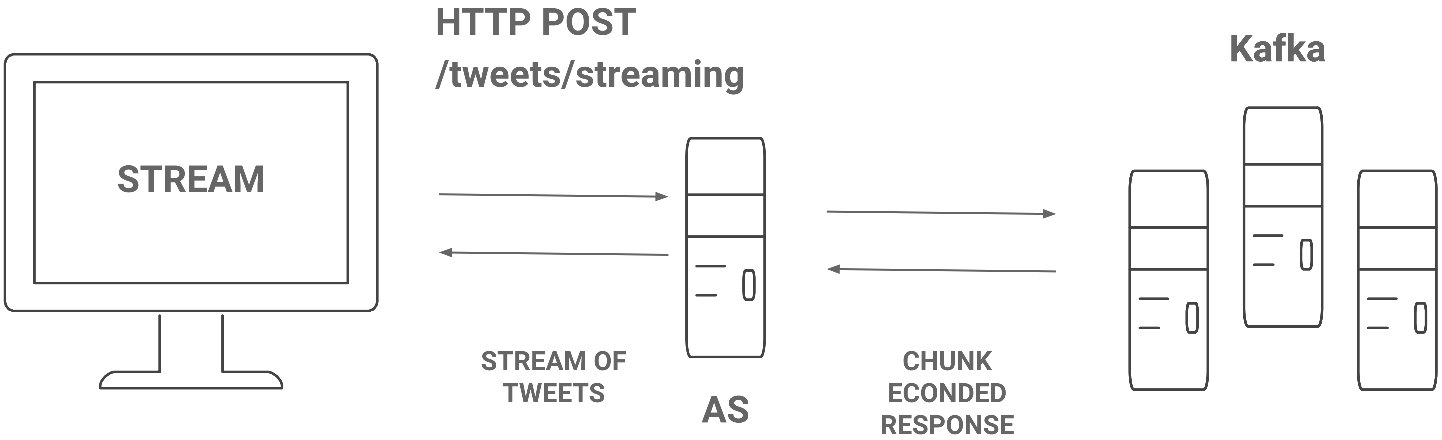
Batch Reading

- **AvroConsumer**, con diversi brokers
- Get latest N messages: adjust offset from the one of latest message to *(latest-N)*

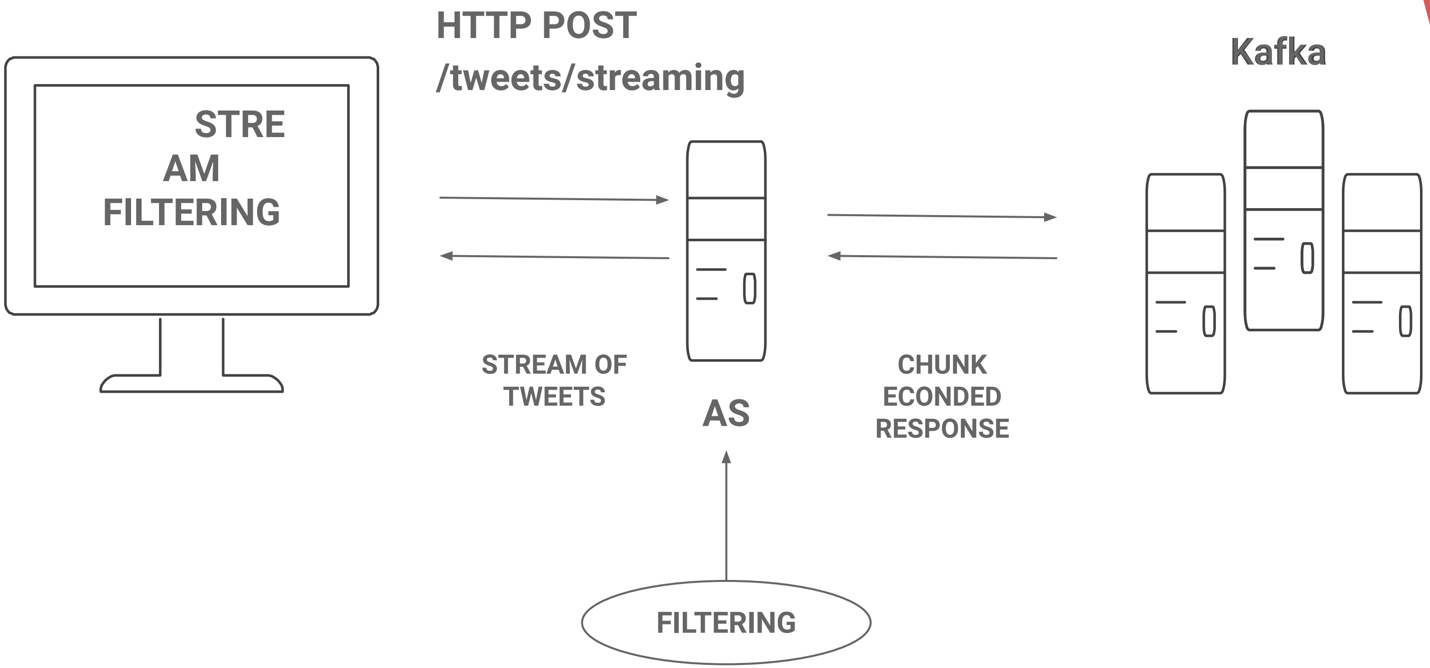


Streaming

- **5 min window** messages
- One request --> **Stream of tweets**



Filtering



HTTP POST
/tweets/streaming

STRE
AM
FILTERING

STREAM OF
TWEETS

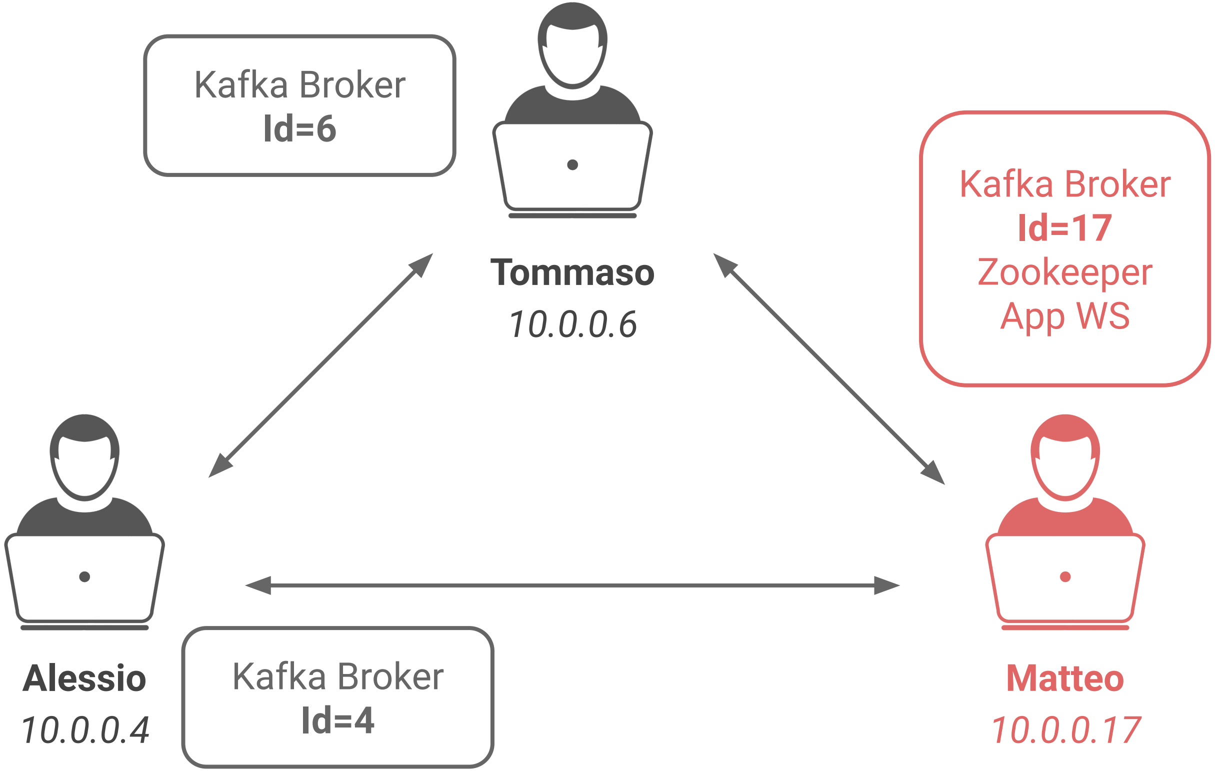
AS

CHUNK
ECONDED
RESPONSE

Kafka

FILTERING

Final Configuration



Kafka Broker
Id=6

Tommaso
10.0.0.6

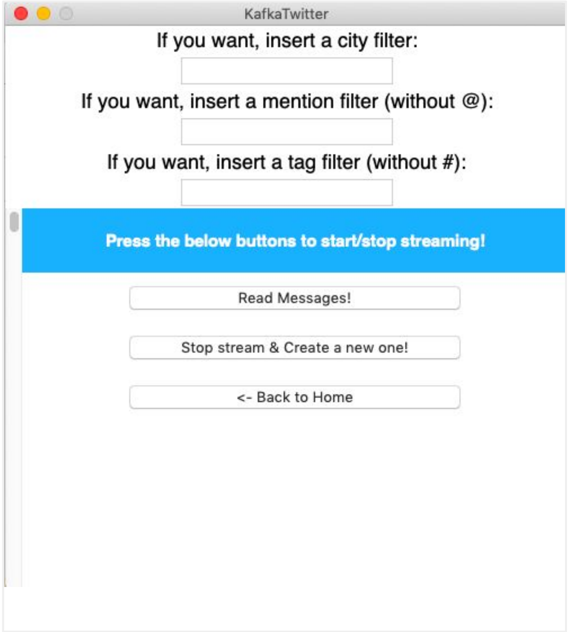
Kafka Broker
Id=17
Zookeeper
App WS

Alessio
10.0.0.4

Kafka Broker
Id=4

Matteo
10.0.0.17

Graphical User Interface

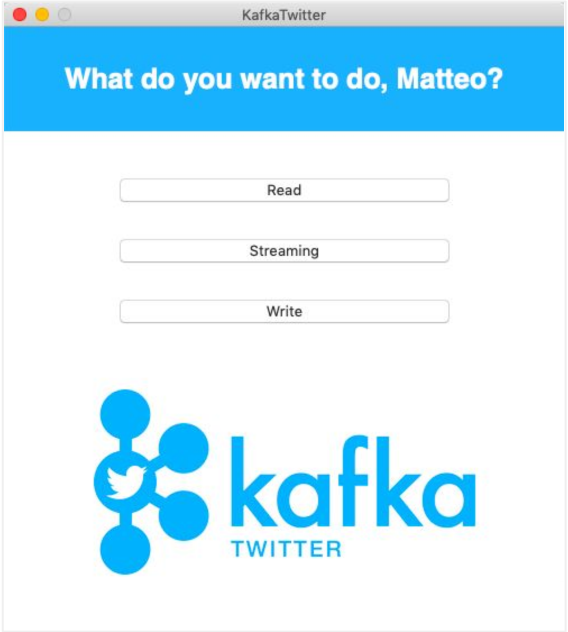

KafkaTwitter

If you want, insert a city filter:

If you want, insert a mention filter (without @):

If you want, insert a tag filter (without #):

Press the below buttons to start/stop streaming!


KafkaTwitter

What do you want to do, Matteo?

3.

BIG-DATA PLATFORM WITH AKKA

github.com/tmscarla/**akka-big-data**

Scope

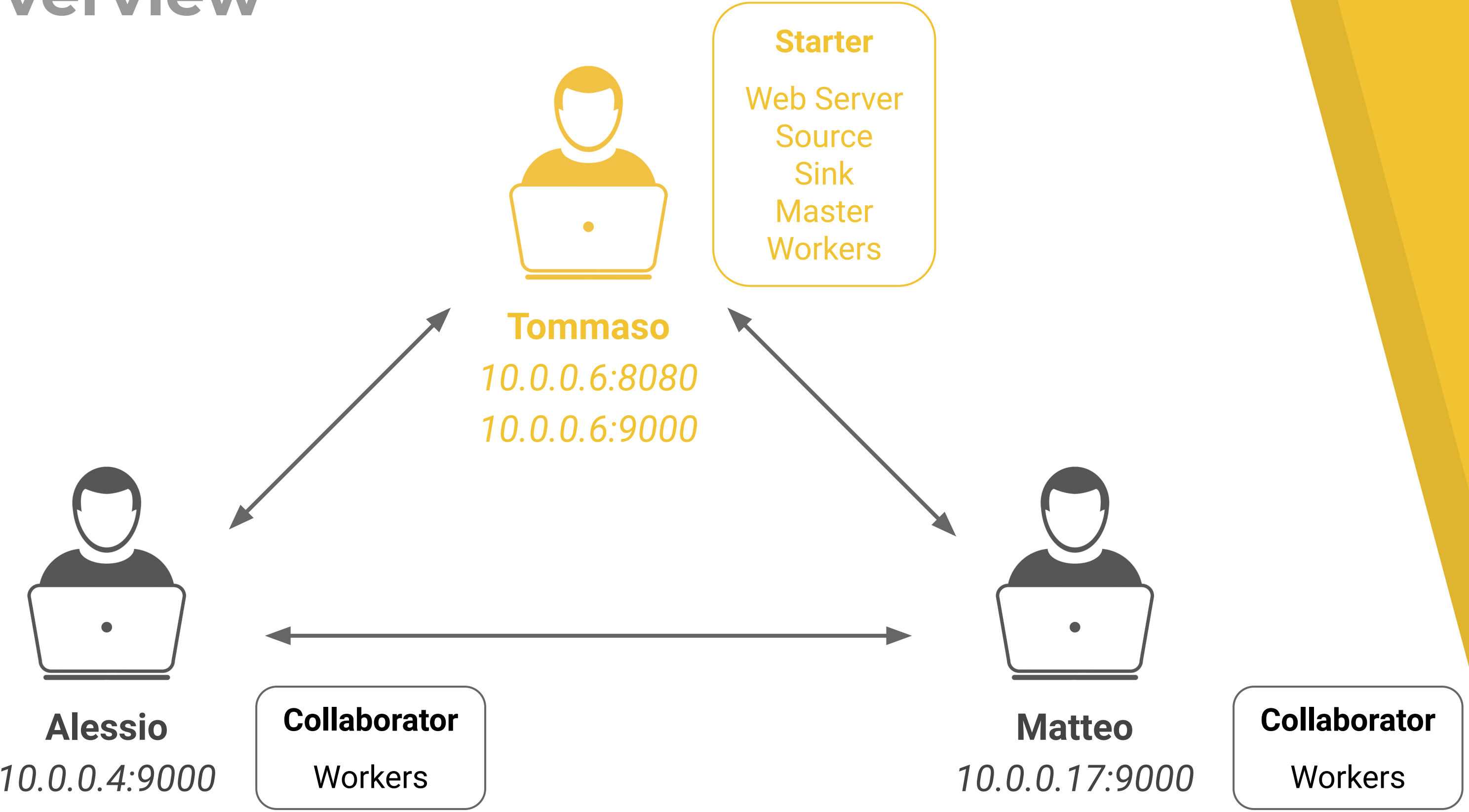
- Implement a Big Data (batch and stream) processing engine using Akka actors
- Engine accepts a sequence of Operators (**Job**) with user-defined functions
  - Source → Job → Sink
- Each operator is performed in parallel through multiple Workers allocated on different nodes

Workflow

- **Starter node:** main actors + a Worker for each operator
- **Collaborator node:** a Worker for each operator

1. Collaborator nodes are initialized
2. Starter node is launched
3. Actors instantiated across nodes
4. Computation begins

Overview



Starter

Web Server
Source
Sink
Master
Workers

Tommaso

10.0.0.6:8080

10.0.0.6:9000

Alessio

10.0.0.4:9000

Collaborator

Workers

Matteo

10.0.0.17:9000

Collaborator

Workers

Source

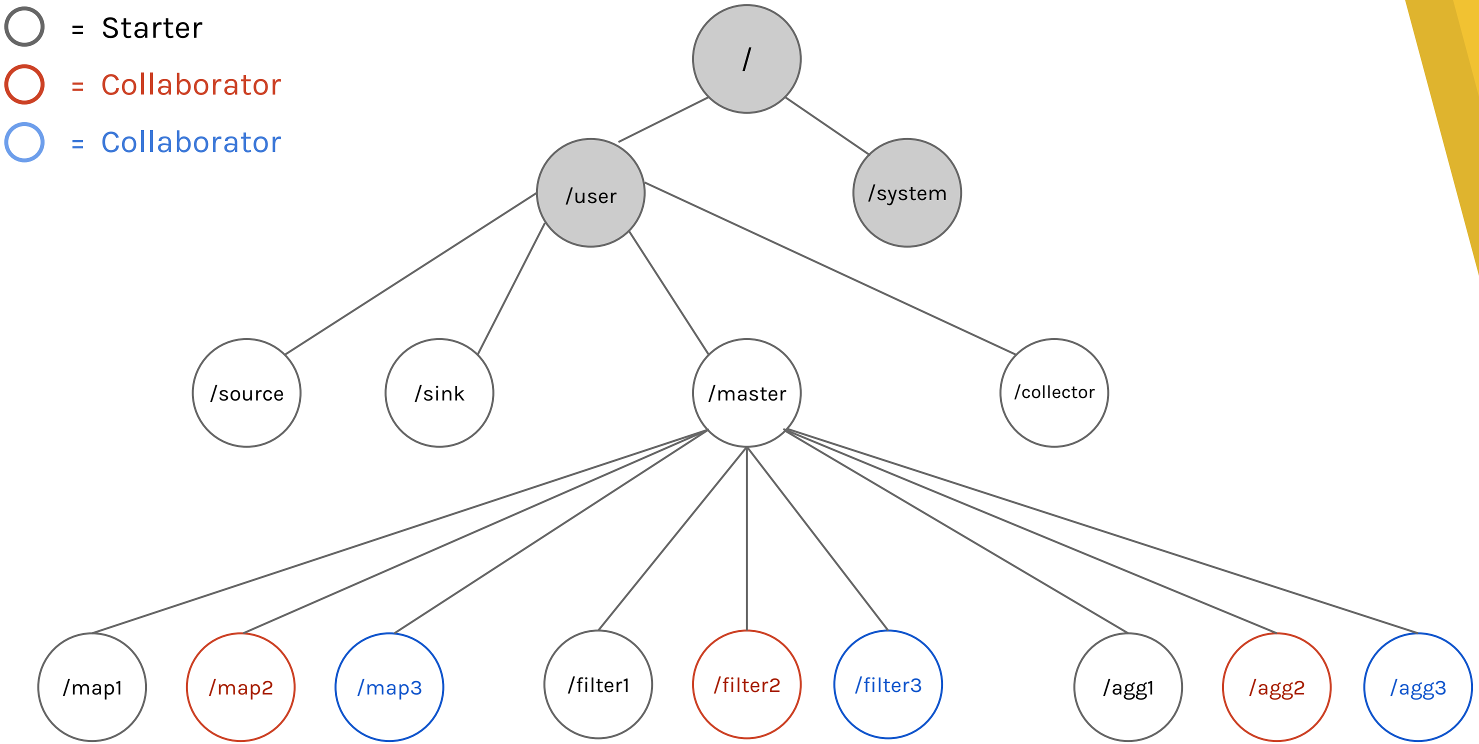
- Source node generate continuosly *<Key, Value>* messages with a specified frequency
- Can operate on two different modes:
    - **Random**: generate random messages within a `keySize` and `valueSize` range
    - **Read**: reads rows from a csv file with two columns `[‘Key’, ‘Value’]`

Actors hierarchy

○ = Starter

○ = Collaborator

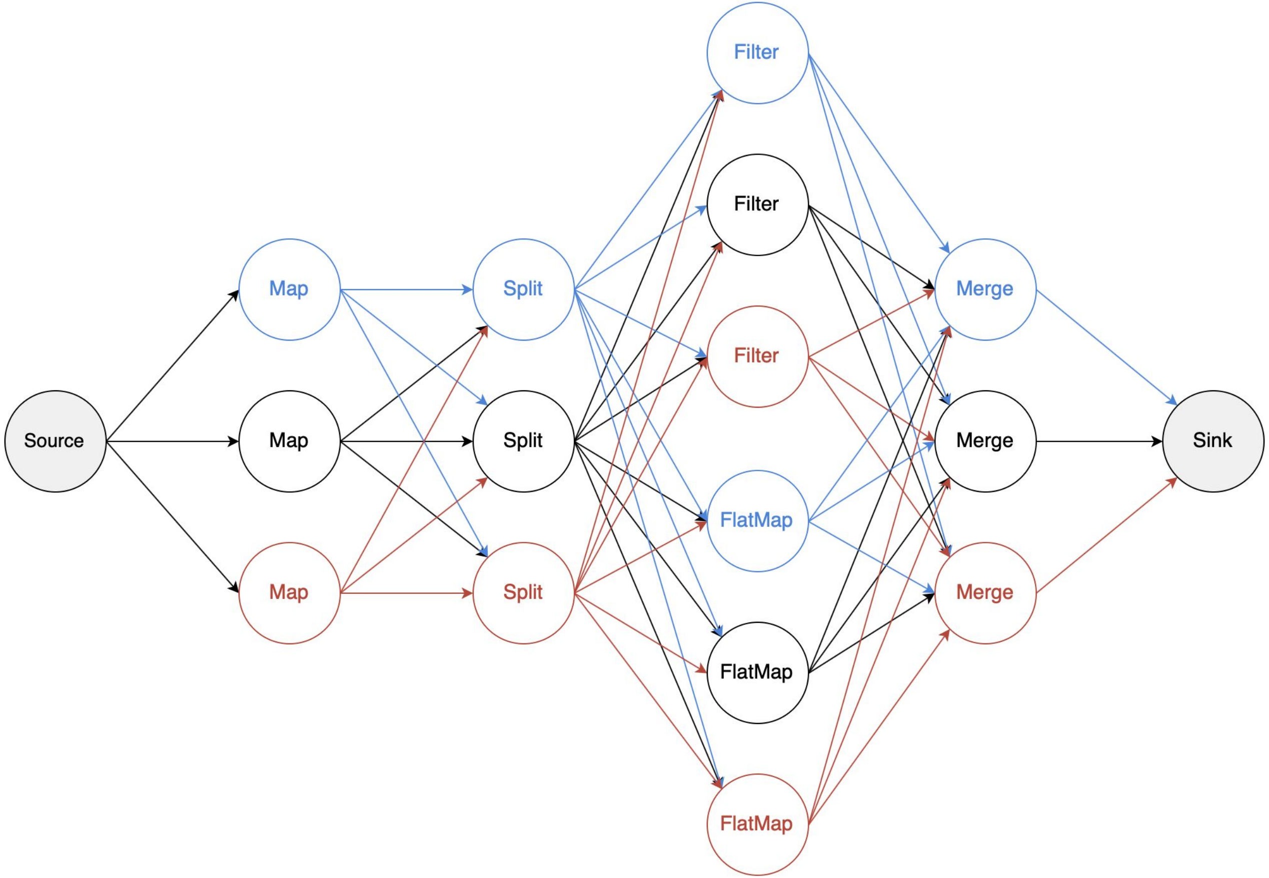
○ = Collaborator



Master node

- Is in charge of allocating all the Workers within its context. In this way is able to:
    - Choose local/remote deployment
    - Set dynamically the downstream of each Worker
    - Handle failures and perform recoveries

From Source to Sink



Batch vs Streaming

- The engine can operate on **two different modes**
- Each operator hold a *batchSize* parameter

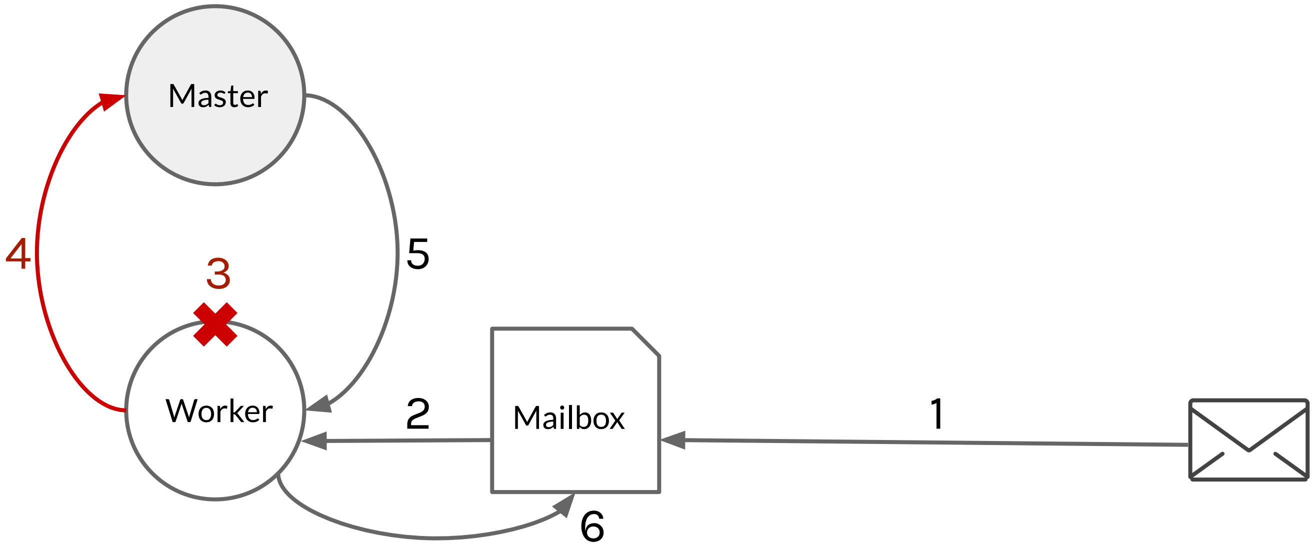
- **Streaming:** on receiving a message send it downstream as soon as it has been processed and it’s ready
- **Batch:** put messages to be sent out in a queue with a FIFO policy. When *batchSize* is filled, send the batch out.

Fault tolerance - What guaranteed delivery means?

1. The message is **sent out** on the network?
2. The message is **received** by the other host?
3. The message is put into the target actor’s **mailbox**?
4. The message is starting to be **processed** by the target actor?
5. The message is **processed successfully** by the target actor?

Fault tolerance mechanism

- Workers can crash during message processing
- Mailboxes configured as **priority queues**
- End-to-end **exactly once delivery**



REST API

- Starter node hosts an HTTP server which exposes a REST API with the following endpoints:
  - RANDOM SOURCE

    ```
    curl -d '{"keySize":"10", "valueSize":"50"}' -H "Content-Type:
    application/json" -X POST http://localhost:8080/source/random
    ```

  - SUBMIT JOB

    ```
    curl -d '{"id":"2"}' -H "Content-Type: application/json" -X
    POST http://localhost:8080 /job
    ```

  - STATISTICS

    ```
    curl -X GET http://localhost:8080 /stats
    ```

4.

PARALLEL K-MEANS WITH OPENMP AND MPI

github.com/tmscarla/**k-means-parallel**

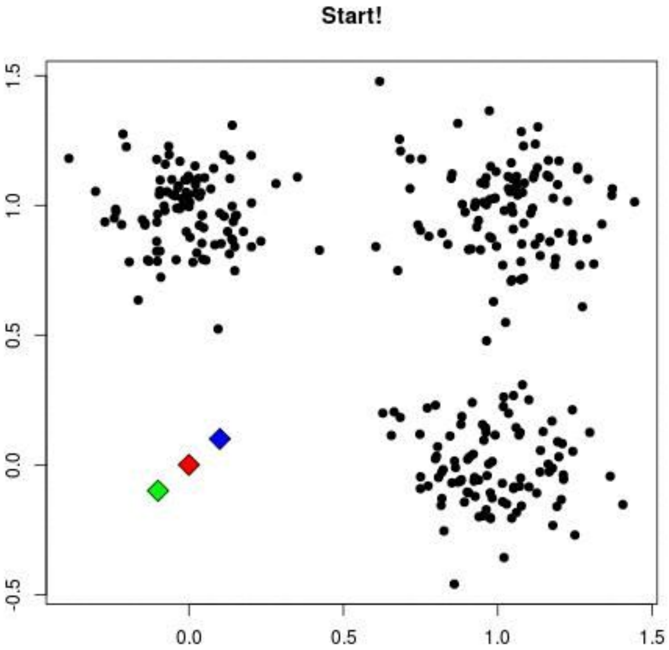
Scope

- Implement the **K-means clustering** algorithm exploiting:
  - Resources within one computing node with multiple processing cores (**OpenMP**)
  - Resources across computing nodes (**MPI**)
- Maximize the performances

Initial Configuration:

- *N* points represented in the space with a *M*-dim vector
- *P* processors defined as:
  - Node 0, Node 1, …, Node *P*-1
- *K* clusters to be considered
- *L* number of maximum iterations to be performed

K-means algorithm



Assumptions:

- Points are independent
- A point belongs to one and only one cluster

Termination:

- No changes in two adiacent iterations
    - Flag in each processor
- Number of iterations > *L*

Workflow

1) Node 0 **loads data** and assigns *N/P* points to each node. Remaining points are assigned one by one.

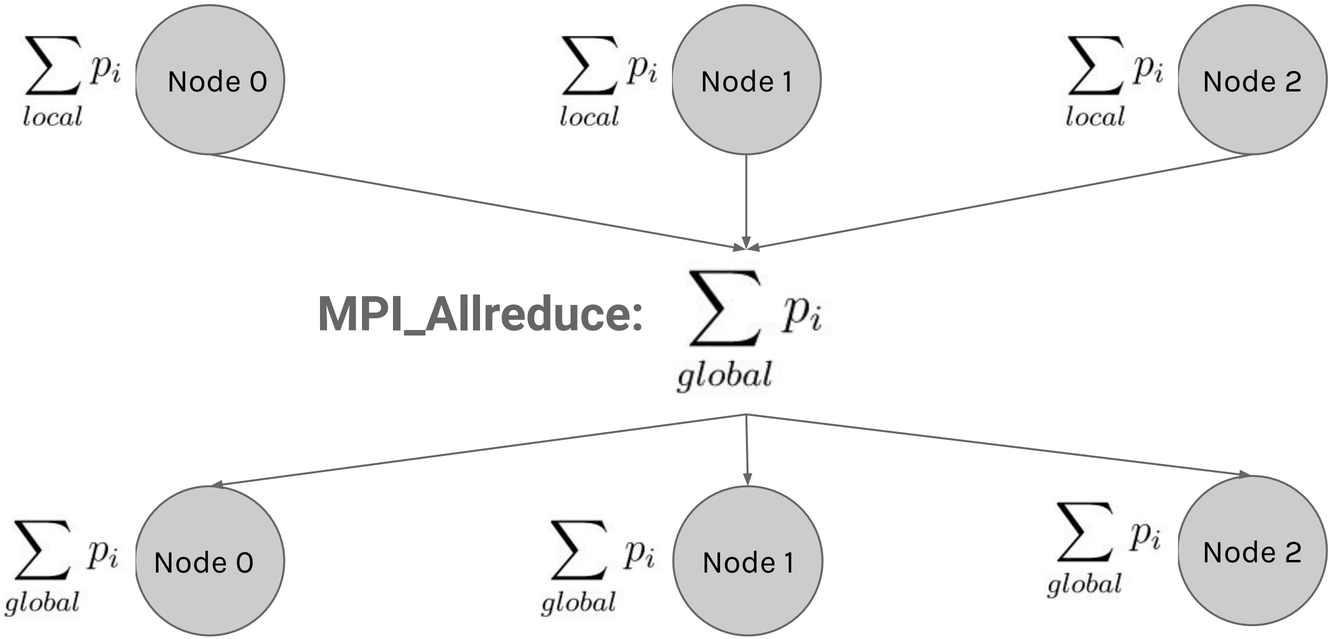
2) Node 0 reads and sets initial **configuration parameters**: *K, M, L*

3) Node 0 chooses *K* points as initial centroids and **broadcasts** them to the other nodes

Workflow:

4) Each node:
   - For each “local” point find the cluster membership among the *K* clusters
   - Distance calculation between points and centroids is performed in parallel using **OpenMP**
   - For each cluster, sum points dimensions values
5) After an ***MPI_Allreduce*** operation, each node knows the number of points and the sum of their values within each cluster. Compute new centroids
6) Go to point 4) and repeat until termination

Centroids recomputation:



$\sum_{local} p_i$ Node 0

$\sum_{local} p_i$ Node 1

$\sum_{local} p_i$ Node 2

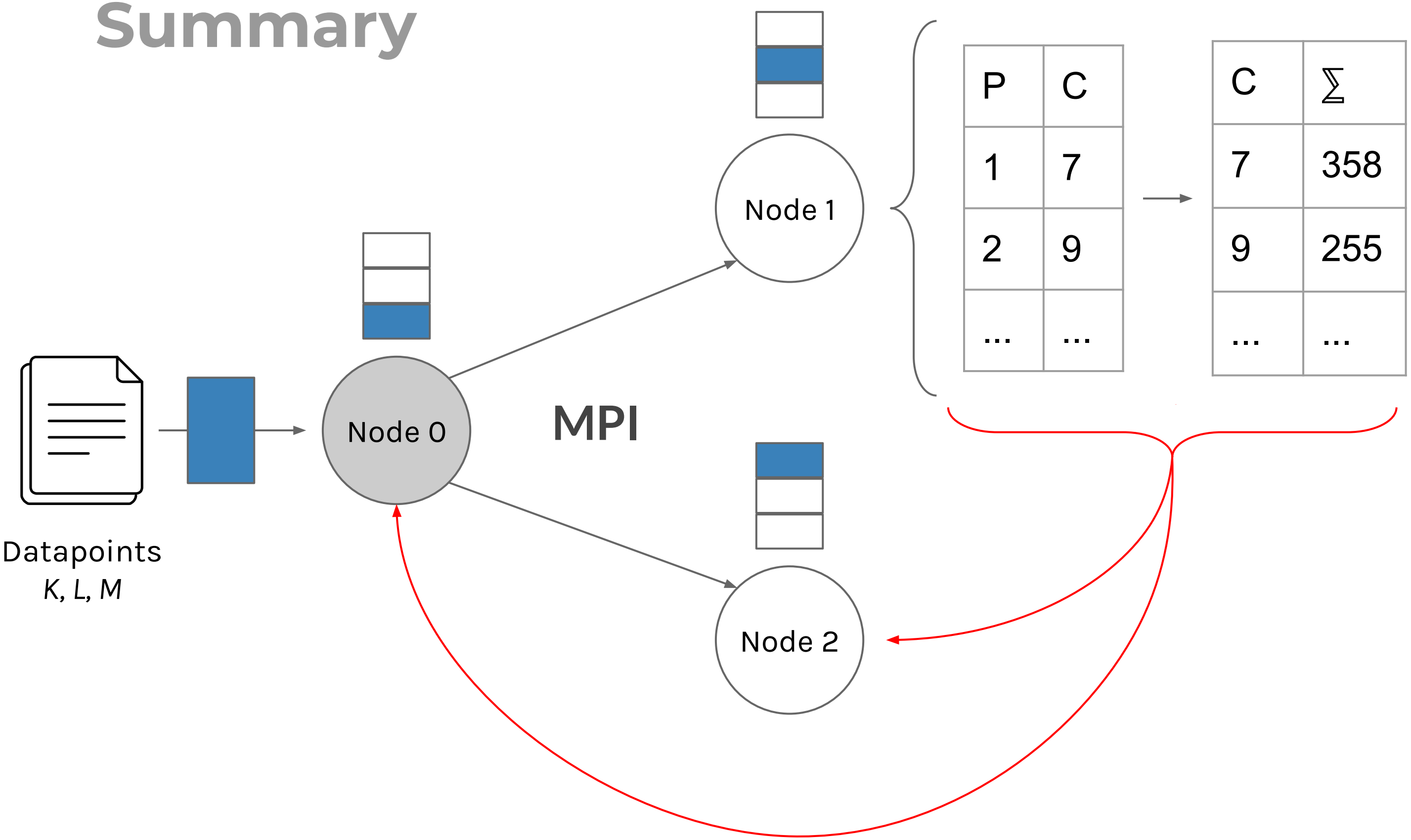
MPI_Allreduce: $\sum_{global} p_i$

$\sum_{global} p_i$ Node 0

$\sum_{global} p_i$ Node 1

$\sum_{global} p_i$ Node 2

Summary



P	C
1	7
2	9
...	...

C	∑
7	358
9	255
...	...

Distance metrics

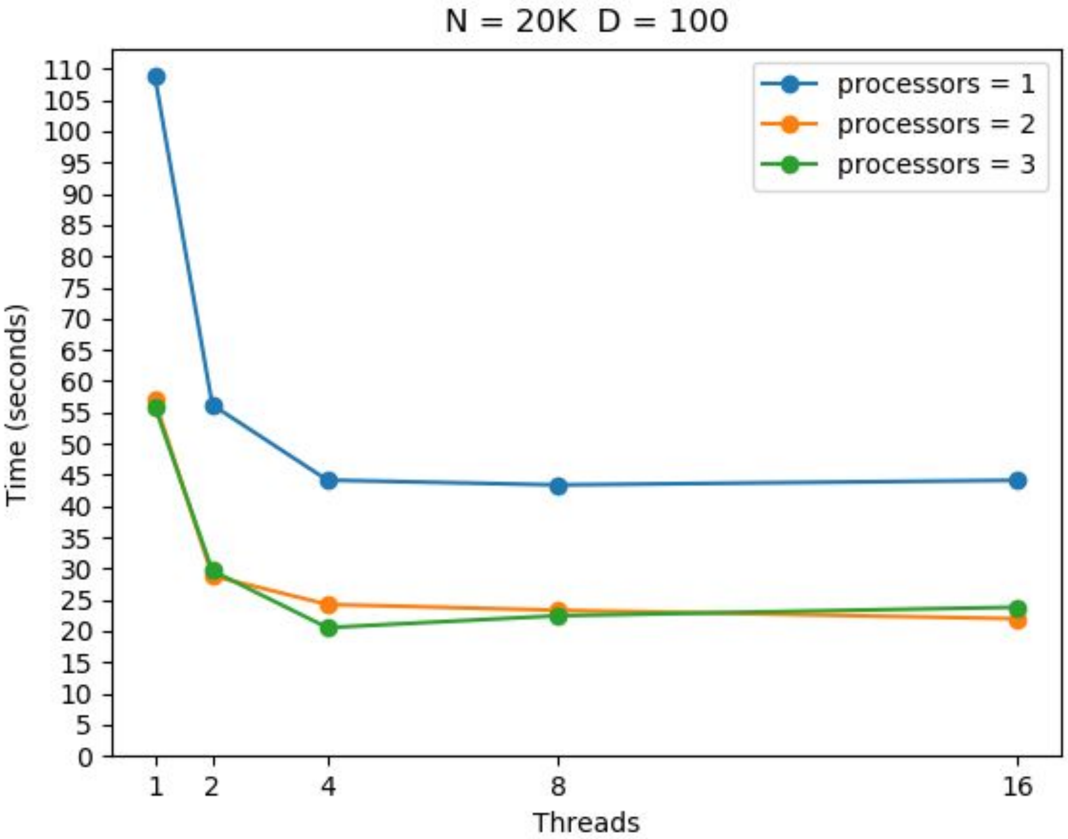
- Euclidean Distance:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

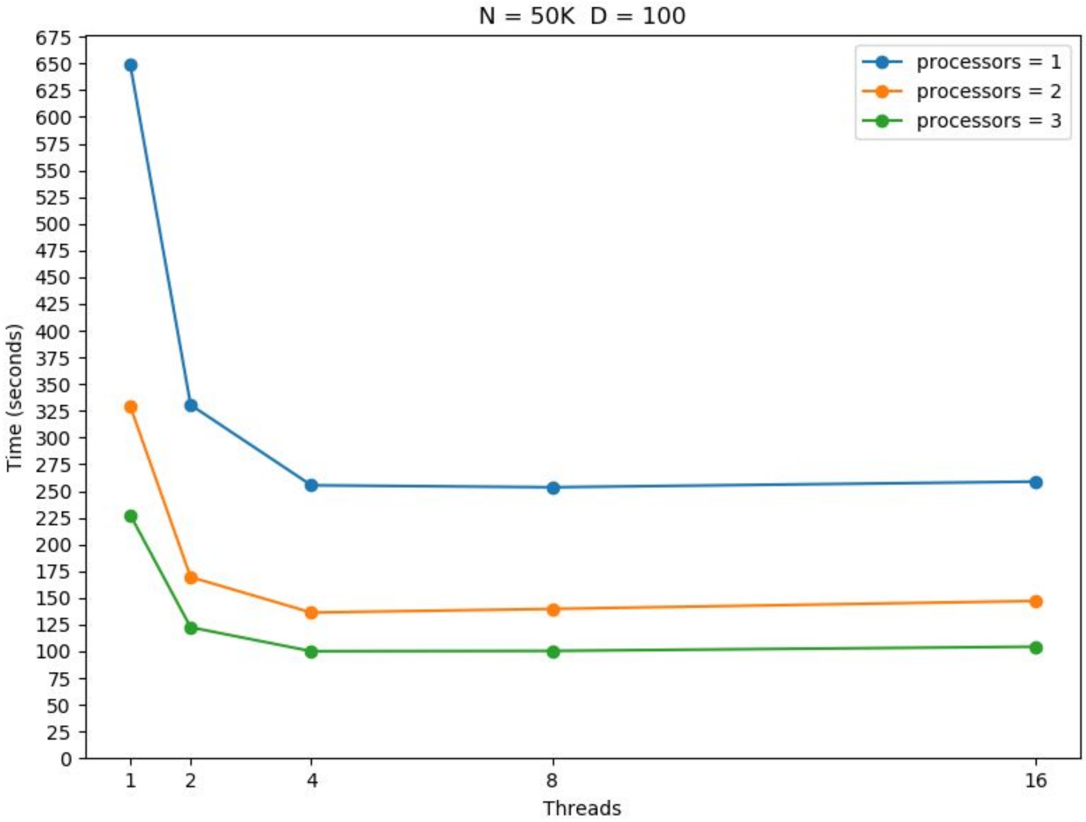
- Cosine Similarity:

$$Cosine(\mathbf{A}, \mathbf{B}) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Results (1/2)

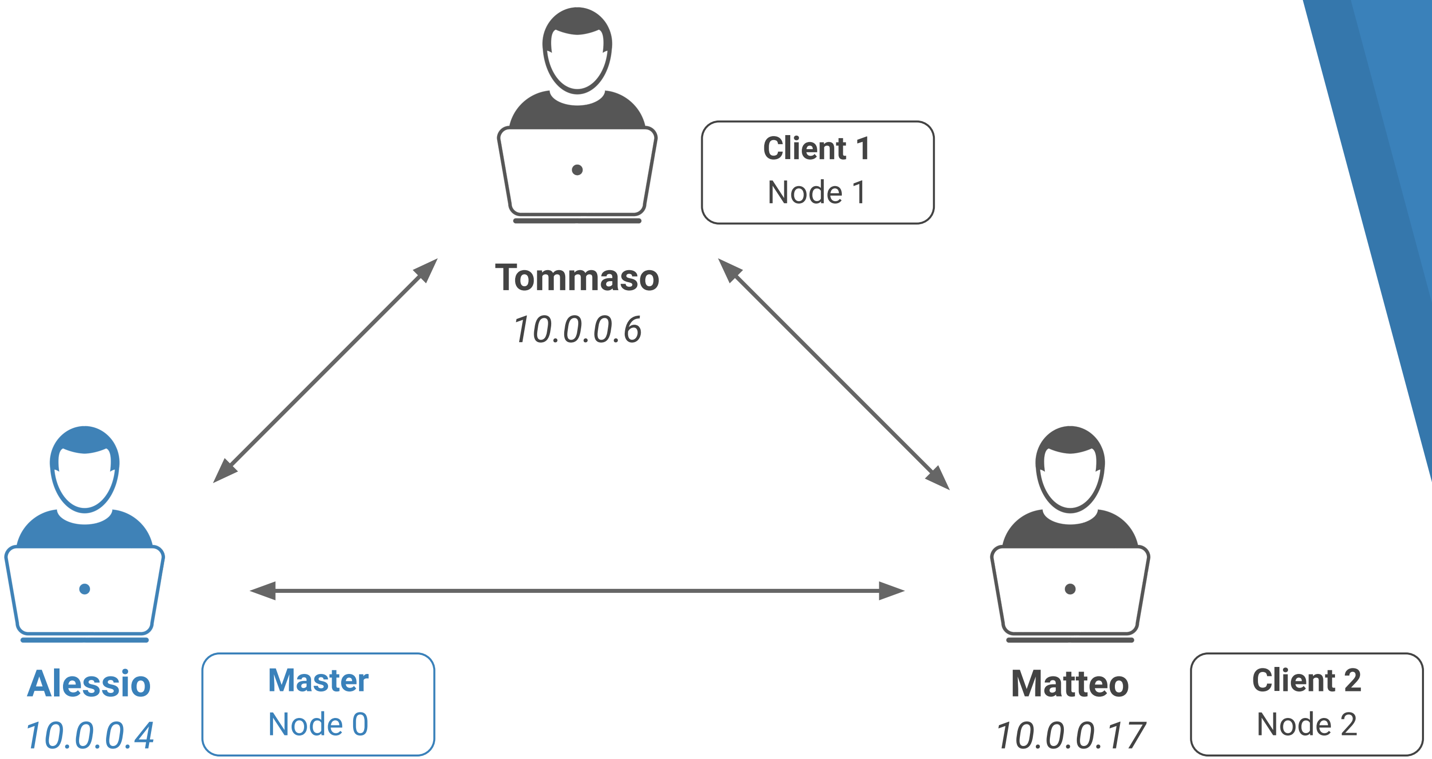

N = 20K D = 100
Time (seconds)
Threads
processors = 1
processors = 2
processors = 3
0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
1
2
4
8
16

Results (2/2)



N = 50K D = 100
processors = 1
processors = 2
processors = 3
Time (seconds)
0
25
50
75
100
125
150
175
200
225
250
275
300
325
350
375
400
425
450
475
500
525
550
575
600
625
650
675
Threads
1
2
4
8
16

Overview



Tommaso
10.0.0.6

Client 1
Node 1

Alessio
10.0.0.4

Master
Node 0

Matteo
10.0.0.17

Client 2
Node 2

5.

RESULTS AND CONCLUSION

LDA vs Doc2vec clustering

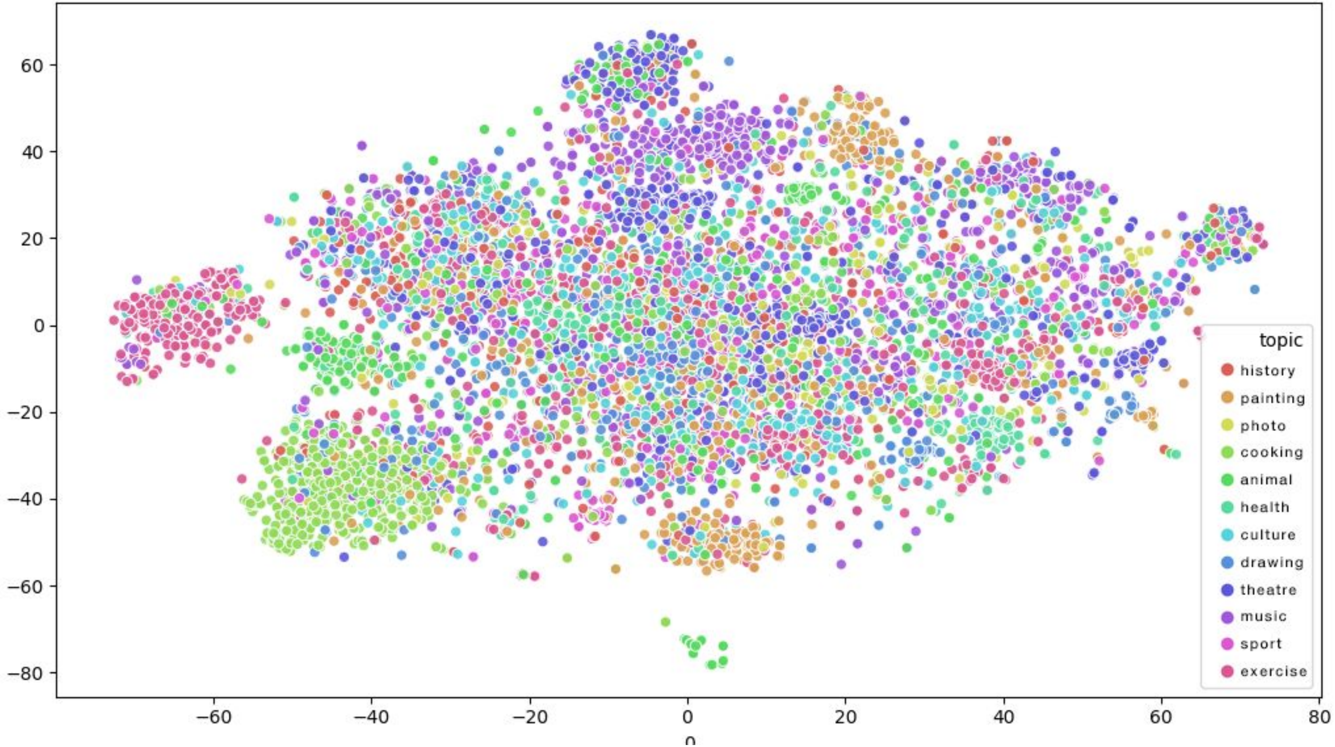
- Both **unsupervised** learning algorithms
- Both based on a **bag-of-words** (BoW) model
- Fuzzy vs disjoint clustering
  - LDA infers topics based on word counts
  - K-means leverages on the numeric representation of documents created by **doc2vec**

Two methods, one goal

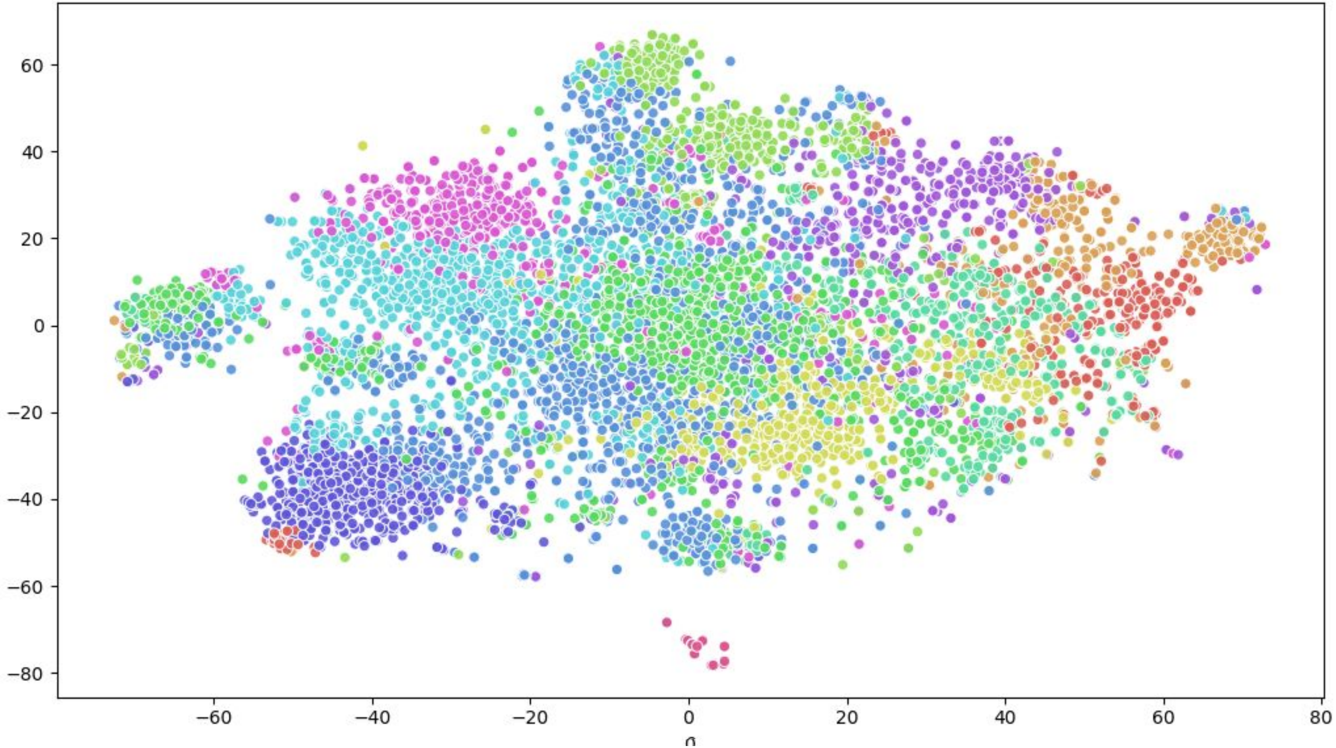
- Both methods are suited for **topic extraction** tasks
- Using **n_topics = n_clusters**, can we compare the two methods?
- In other words, can we quantify the similarity of the two approaches in labeling each tweet?

- We can **compare** the labels of the tweets within the same cluster and infer the topic!

Doc2vec - 2D (t-SNE)



K-means (cosine similarity)



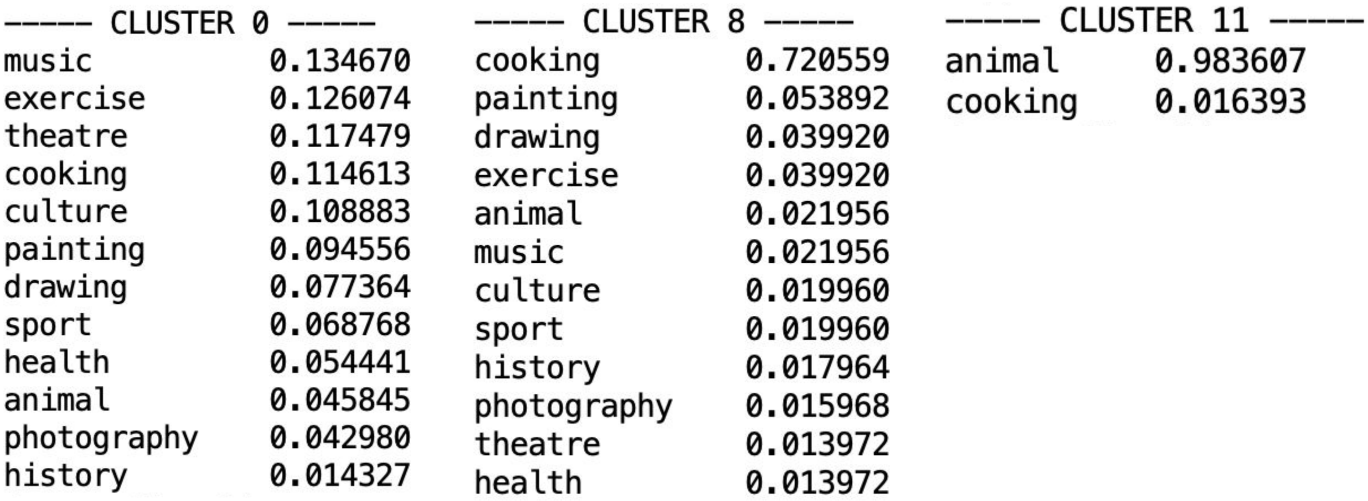
Evaluation metric

- **Purity:** a measure of the extent to which clusters contain a single class. How “pure” is respect to the dominant class:

$$purity = \frac{1}{N} \sum_{k} max_j |\omega_k \cap c_j|$$

- **N** number of tweets
- **k** number of clusters
- **ω_k** dominant class
- **c_j** real class

Results



```
----- CLUSTER 0 -----        ----- CLUSTER 8 -----        ----- CLUSTER 11 -----
music          0.134670      cooking          0.720559    animal     0.983607
exercise       0.126074      painting         0.053892    cooking    0.016393
theatre        0.117479      drawing          0.039920
cooking        0.114613      exercise         0.039920
culture        0.108883      animal           0.021956
painting       0.094556      music            0.021956
drawing        0.077364      culture          0.019960
sport          0.068768      sport            0.019960
health         0.054441      history          0.017964
animal         0.045845      photography      0.015968
photography    0.042980      theatre          0.013972
history        0.014327      health           0.013972
```

Purity = 0.21137750

Conclusions

- We explored three different technologies using three different programming languages
- We used **tweets** extracted from Twitter to create a **common thread** between the projects
- We tested our projects in a **really distributed scenario** using a wireless ad-hoc network between our three notebooks
- We compared **LDA** and **K-means** clustering of a **Doc2Vec** representation using purity as evaluation metric

THANK YOU!

Questions?

POLITECNICO
MILANO 1863

Matteo Moreschini
Alessio Russo Introito
Tommaso Scarlatti