In-memory Spatial-Aware Framework for Processing Proximity-Alike Queries in Big Spatial Data

Isam Mashhour Al Jawarneh, Paolo Bellavista, Antonio Corradi, **Luca Foschini**, Rebecca Montanari, Andrea Zanotti

DISI, University of Bologna, ITALY

{isam.aljawarneh3, paolo.bellavista, antonio.corradi, **luca.foschini,** rebecca.montanari}@unibo.it, andrea.zanotti@studio.unibo.it

***Monday, 17th September 2018*, IEEE CAMAD'18**

Quality of Service Aware Data Stream Processing for Highly Dynamic and Scalable Applications

Isam Mashhour Hasan Al Jawarneh
PhD Candidate, Cycle XXXII

DISI, University of Bologna, ITALY
isam.aljawarneh3@unibo.it,

Supervisor:
Prof. **Rebecca Montanari**

PhD program coordinator:
Prof. **Davide Sangiorgi**

PhD. Esame finale anno 2020, 2nd

Agenda

- **Smart City and Big Data Context**
  - Geolocated big data
  - MapReduce
  - Spark and GeoSpak
- **Spatial-Aware Big Data Management Strategies**
  - Self-Adaptable Spatial-Aware Partitioner (SASAP)
  - Spatial-aware query optimizations
- **Experimental Results**
  - Partitioning results
  - Proximity, containment and density based clustering query performance results
- **Conclusions & Ongoing Works**

Smart City and Big Data Context

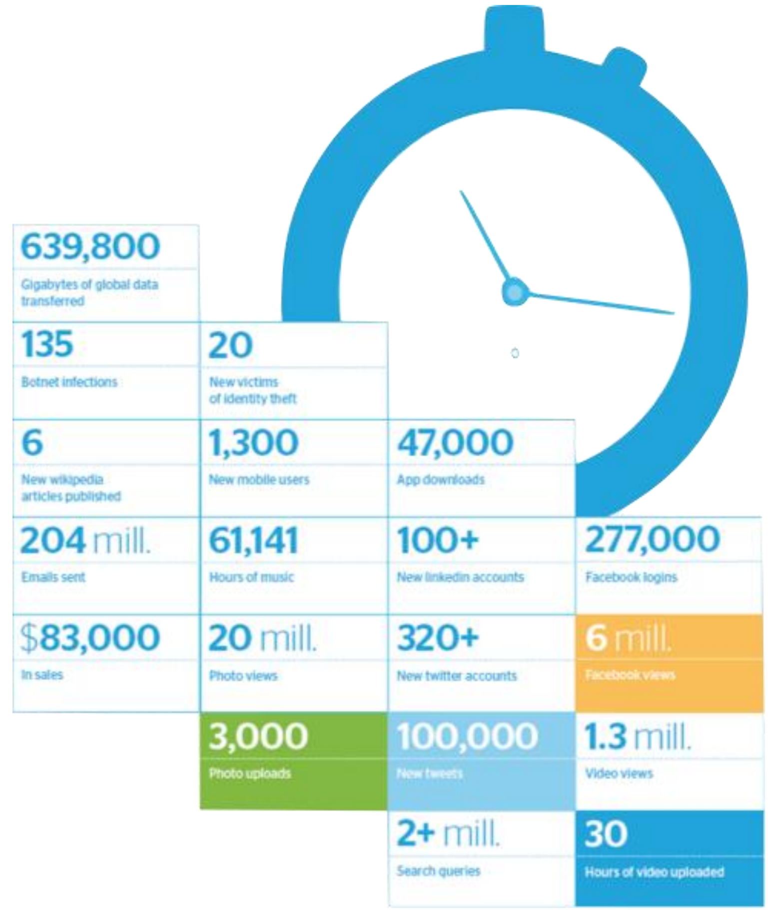
Smart City
Advanced technological services

Geographic Big Data
Huge amount of information

Mobile Sensing
Automatic collection of data

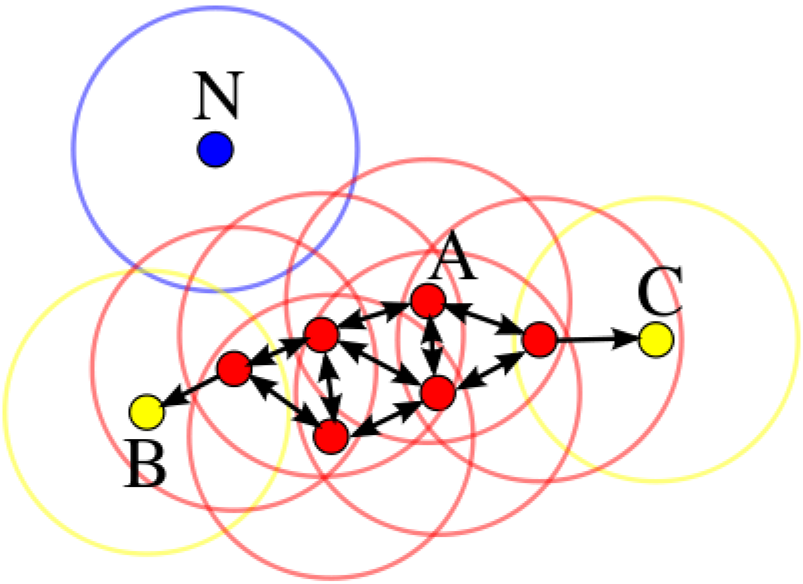
Geo-located Big Data

- Large amounts of geolocated data, **exceeding processing capability** of traditional database management systems
- Charateristics
  - **Volume**: amount of data
  - **Velocity**: streaming data
  - **Variety**: multiple sources, heterogeneous data
- Other charateristics
  - **Veracity**: uncertainty degree
  - **Variability**: possible inconsistency
  - **Complexity**: difficult to establish connection and relationships between data



Advanced Geospatial Queries & Clustering DBSCAN Algorithm

- **Eps (ε)** = max distance
- **MinPts** = minimum number of points inside radius ε



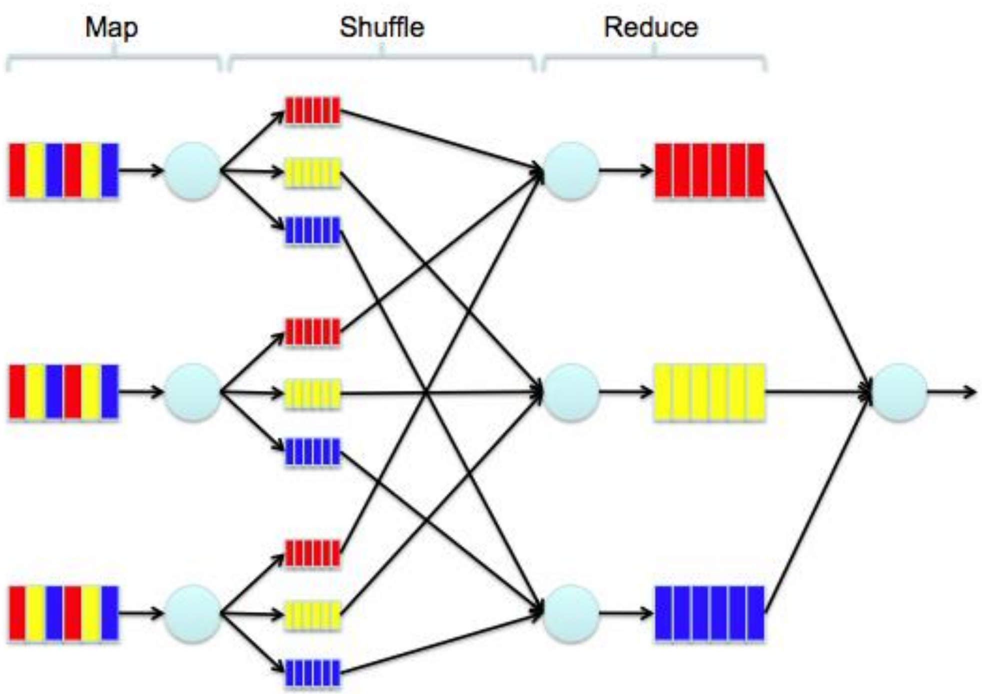
- **Core points (cp):** #(X: dist(X, Y) ≤ ε) ≥ MinPts
- **Cluster**: {X: dist(X, cp) ≤ ε} U {cp}

MapReduce

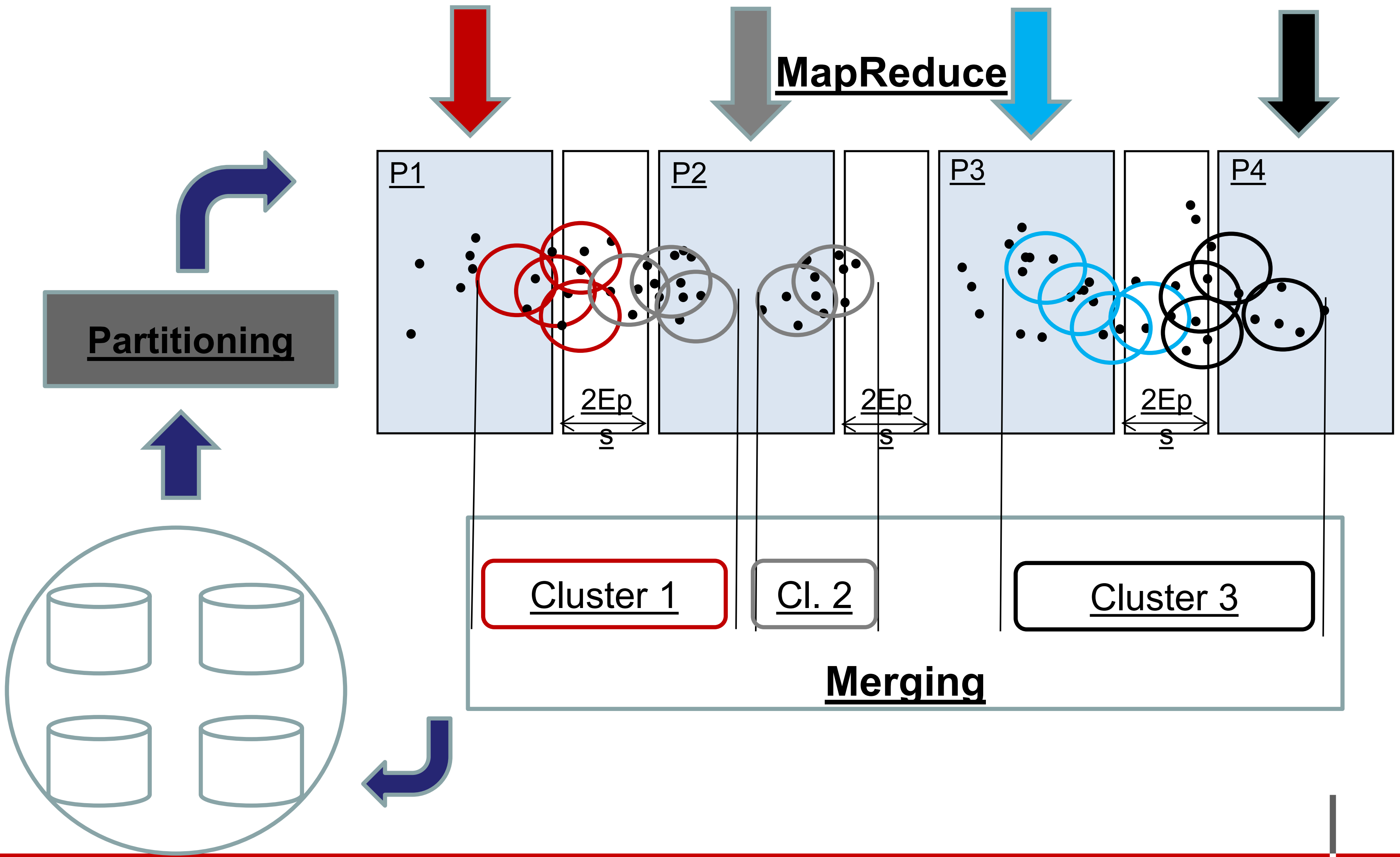
Programming paradigm for computing and aggregating **large amounts of data**

Functions:

- **Map**: for each input data, returns a **key-value pair**
- Intermediate **grouping** and **sorting** by key step
- **Reduce**: final aggregation step



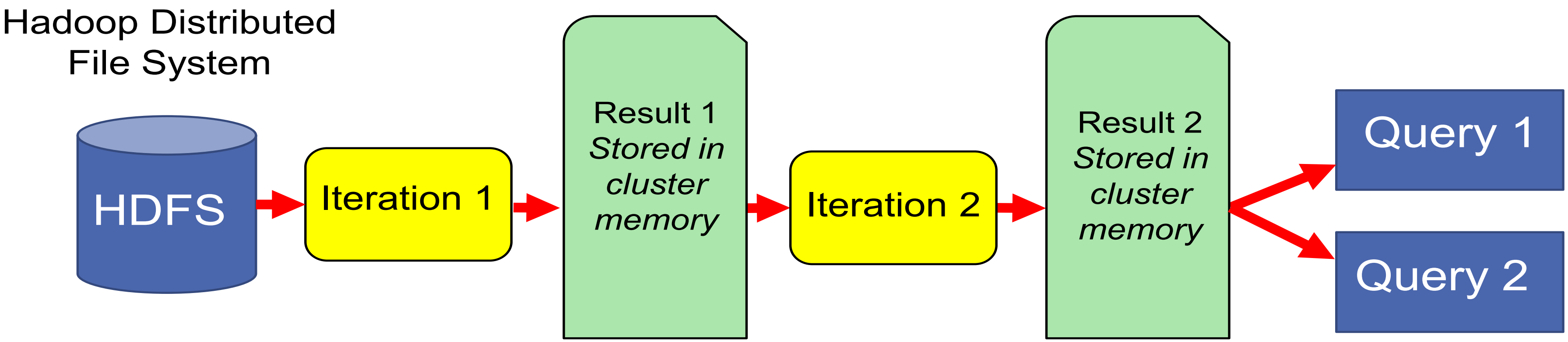
DBSCAN-MR

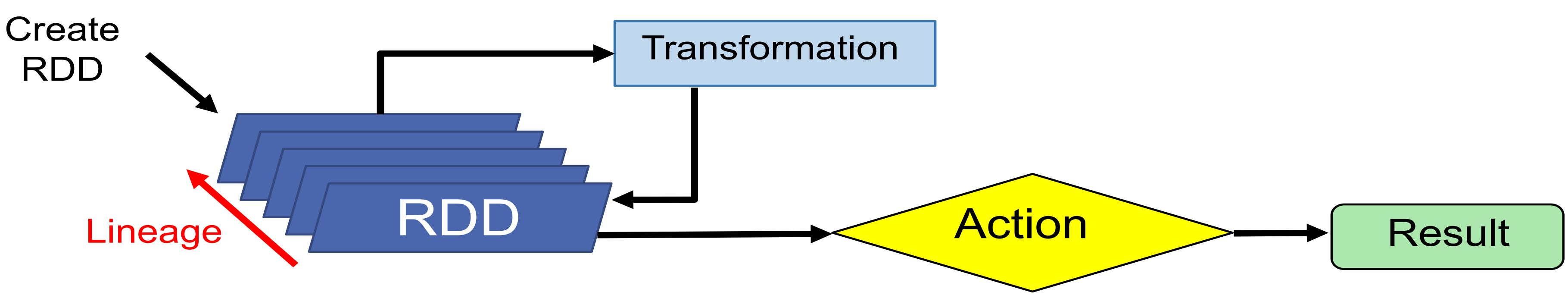


MapReduce
P1
P2
P3
P4
2Eps
2Eps
2Eps
Partitioning
Cluster 1
Cl. 2
Cluster 3
Merging

Apache Spark

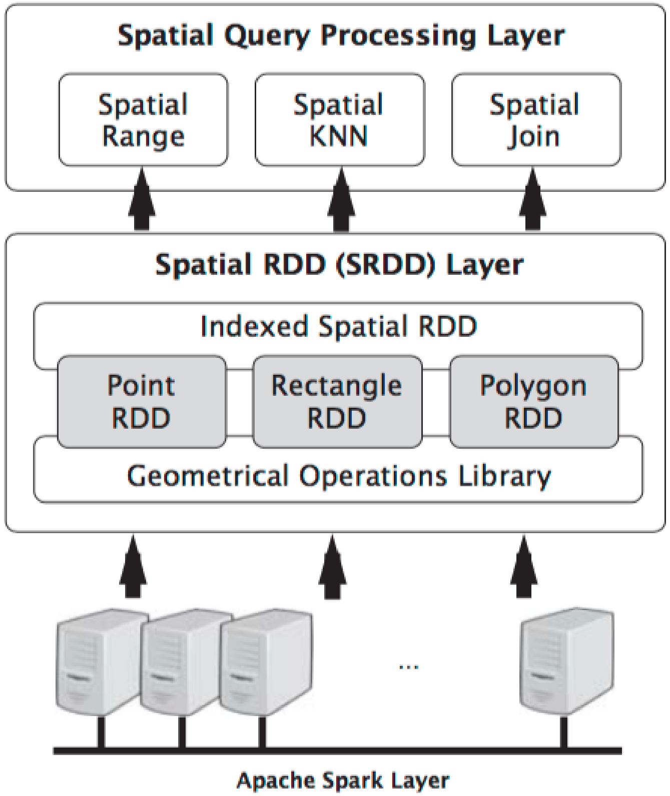
Data stored in memory: analyses on in-memory data and use of **Resilient Distributed Dataset** (RDD)





GeoSpark

- **Geospatial-Aware solutions**
  - ***GeoSpark*** : runs exactly as Spark, but with the awareness for geospatial data, consisting of three layers; Spark, spatial RDD and spatial query processing layers
  - Some other competitors are based on Hadoop like SpatialHadoop and Hadoop-GIS
- **Problem**
  - ***Does not include an integrated support for clustering methods***, or customizable modules for specific application requirements
  - Do not consider specific requirements like ***domain-specific data load balancing for optimizing clustering algorithm execution***



GeoSpark

Main Goals and Requirements

- Realize a **transparent partitioning** support of data, based upon **location information**
- Extend transparently the GeoSpark support
- Realize an effective support to proximity query in a partitioned architecture
- Provide optimizations for complex algorithms, such as the DBSCAN-MR density-based clustering algorithm

Spatial-aware optimizations Partitioning and advanced query optimizers
Geopark (including spatial representational support)
Spark Core

Layered Spatial-aware In-memory Processing Optimization Architecture

- Our support **does not require to modify GeoSpark** and is a transparent layer atop of it that hides implementation details from application layer
- Our support **includes spatial-aware partitioning strategies** better trading off three challenges, such as load balancing, containment query optimization, and spatial co-locality evauation
- Our support **includes a DBSCAN-MR implementation** (which belongs to the co-location data mining family) able to work atop GeoSpark

Self-Adaptation

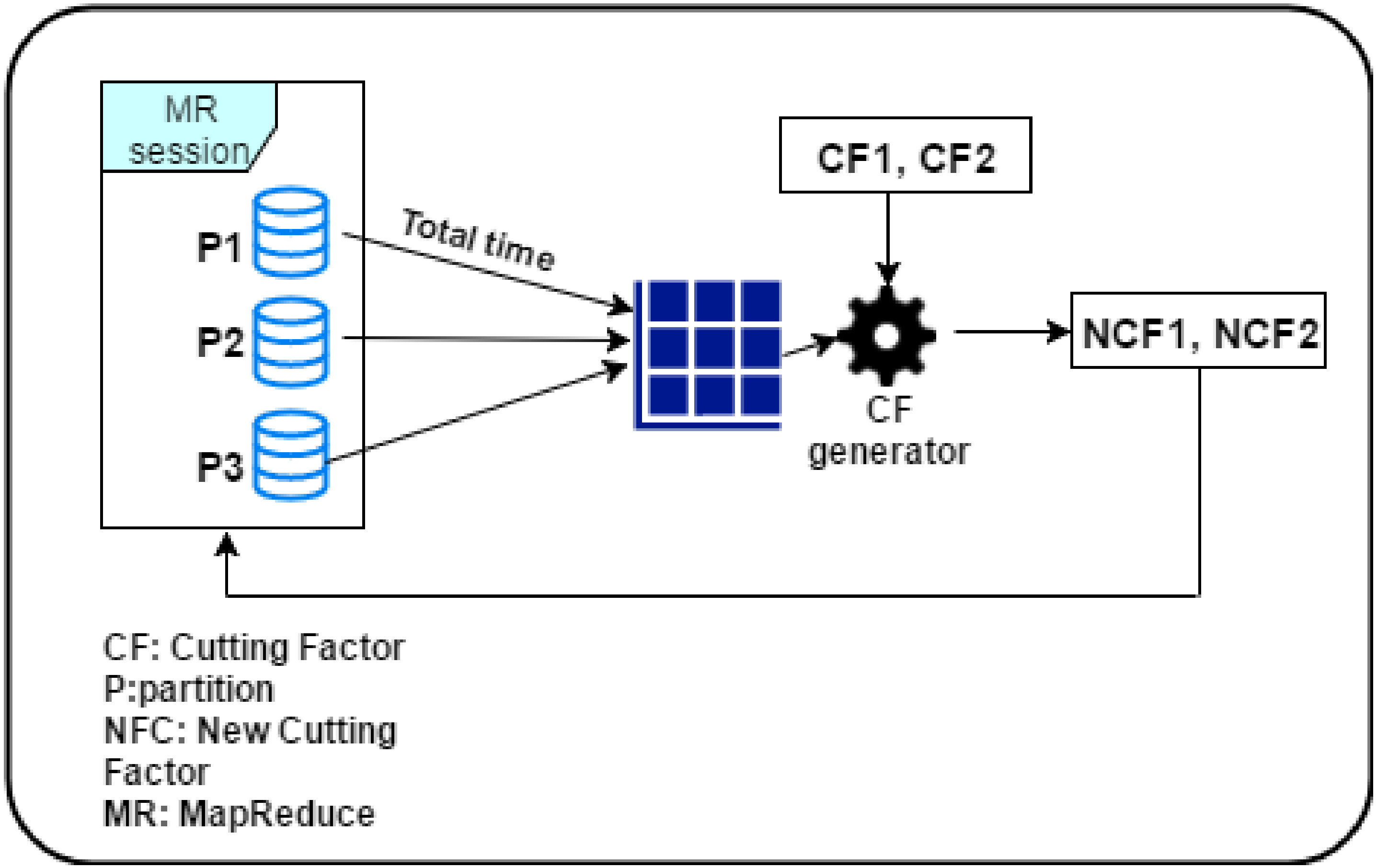
- ***Density-aware spatial data partitioning***, based on spatial object's distribution density
  - Useful with heavily skewed datasets
  - Roughly balances loads across computing resources

- For the execution of new clustering sessions, ***a self-tuning module optimizes cutting factors for subsequent sessions***, gaining performance improvement

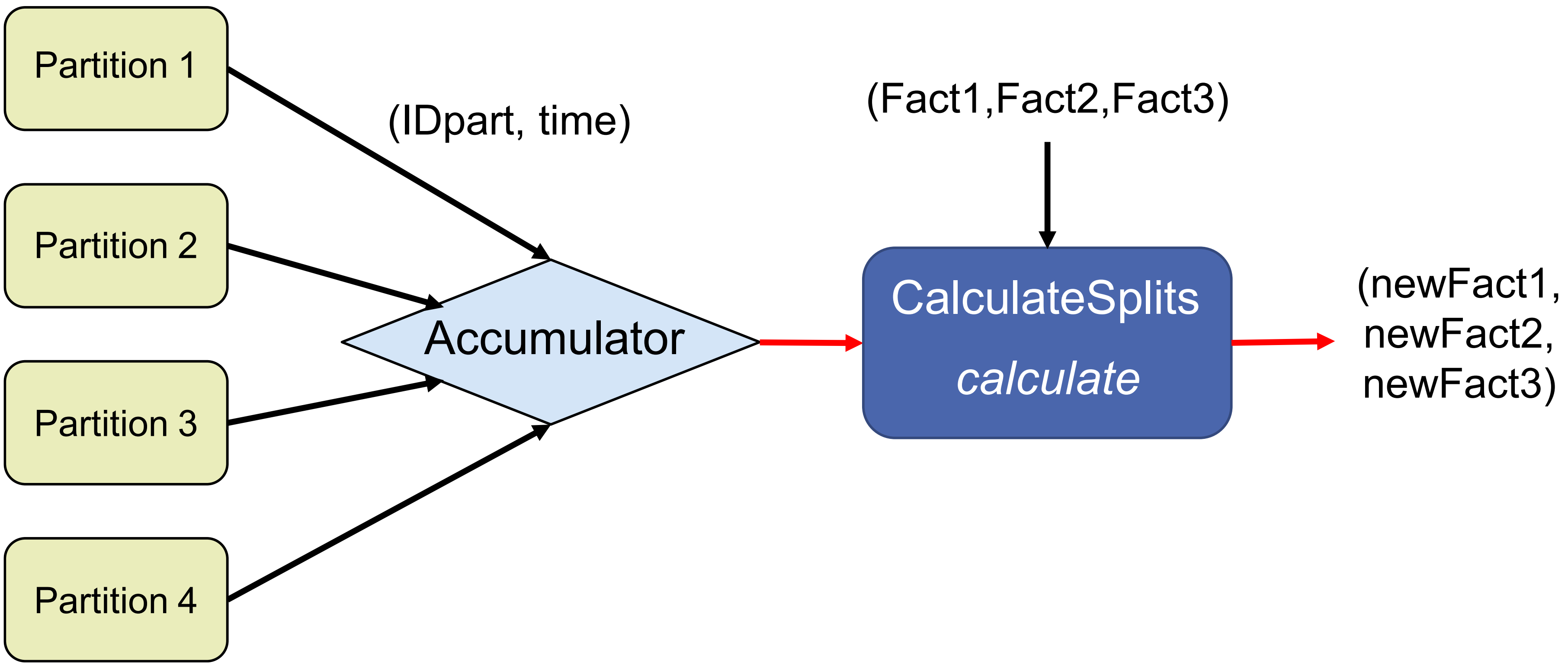
Data Partitioning

- Considering the earth flattened out, we can define ***DBSCAN-MR cutting factors as vertical partitioning lines*** in planar geometry
- ***Cutting factors configuration has an impact*** on partition's loads and spatial boundary objects

Self-Adaptable Spatial-Aware Partitioner (SASAP)



Self-adaptation of vertical cutting factors



- Monitors execution times at each data partition
- Uses a shared Accumulator variable
- CalculateSplits defines the new configuration, using ***threshold-based solutions to decide when to stop configuration tuning***

Experimental Setup

- Our experimental setup utilized ***Amazon AWS cloud's computing services***, specifically ***AWS EC2*** service
- 5 nodes have been used for deployment, one master and four processing
- On each node, spark 1.6.2 was installed, and ganglia 3.7.2 was used for performance analysis. our input database consisted of 250,000 spatial objects collected through ParticipAct project
- Comparison with a MongoDB-based Map-Reduce implementation (w/out using in memory RDDs)

Experimental Setup (cont'ed)

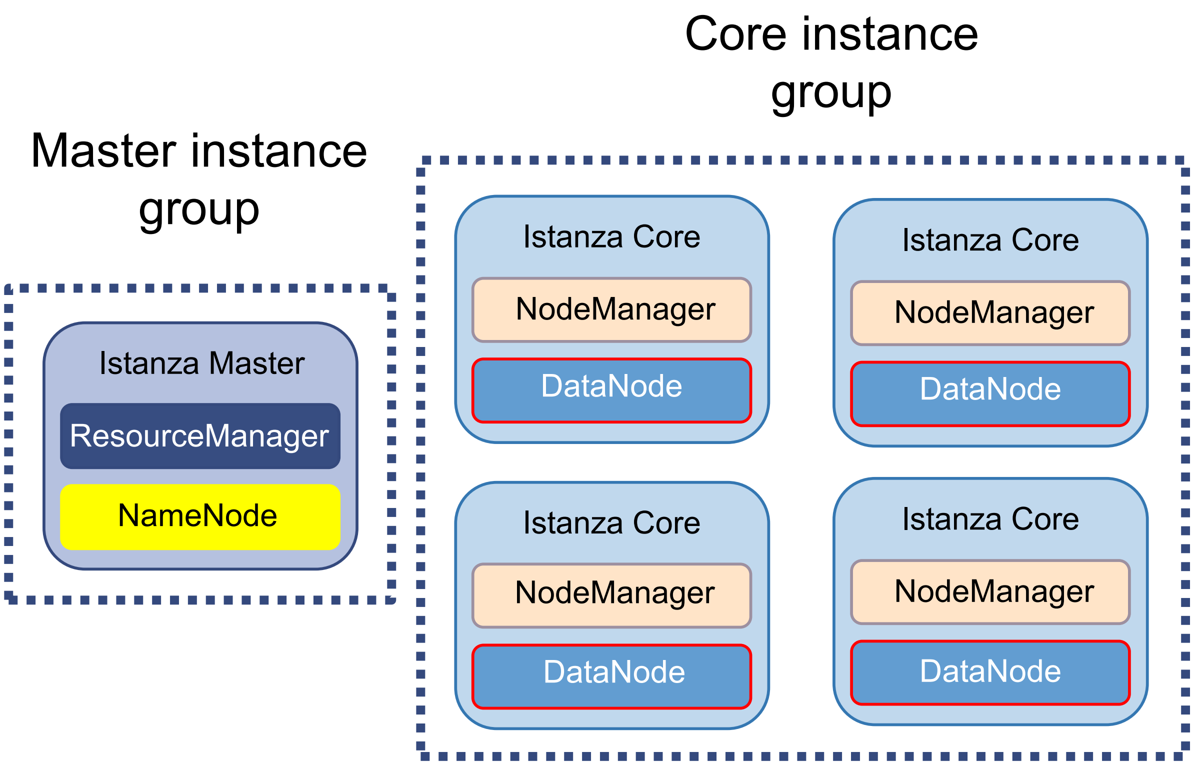
5 instances m3.xlarge using 1vCPU e 15GB RAM

1 Master:

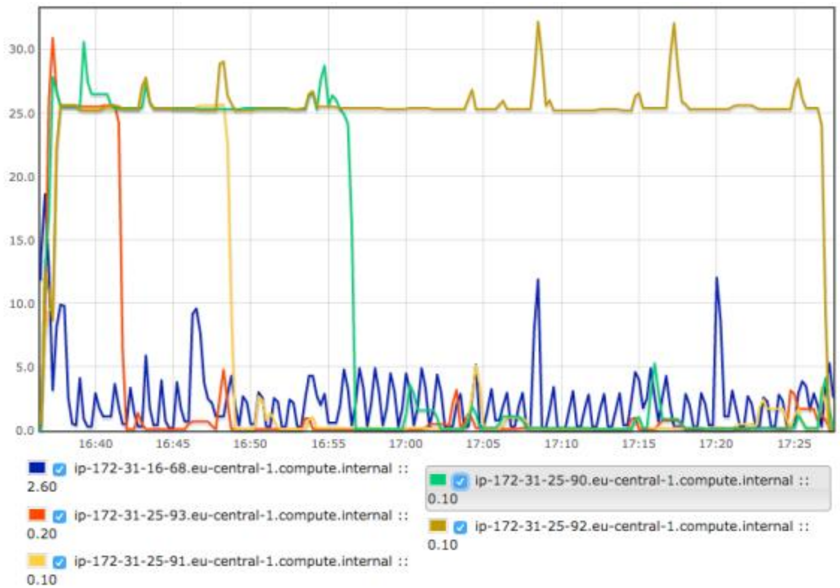
Driver Spark, ResourceManager YARN and NameNode HDFS

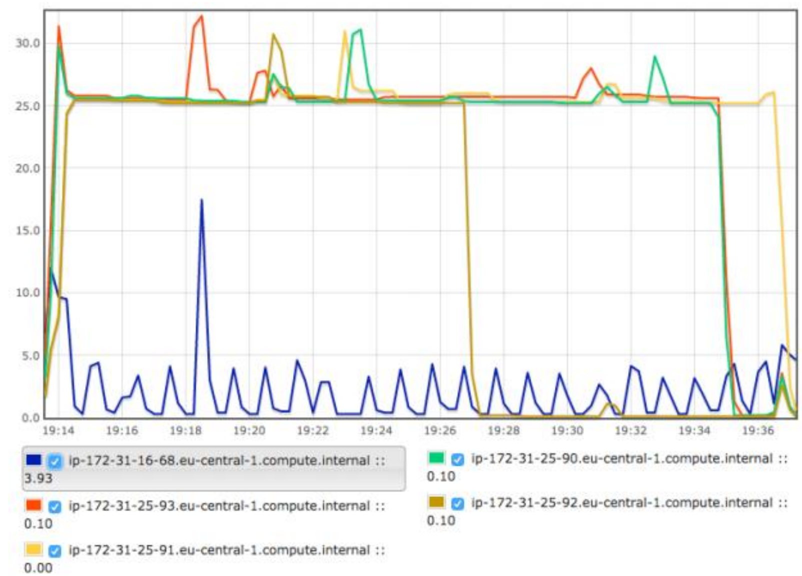
4 Slaves:

Executor Spark and DataNode HDFS



Self-Adaptable Load Balancing

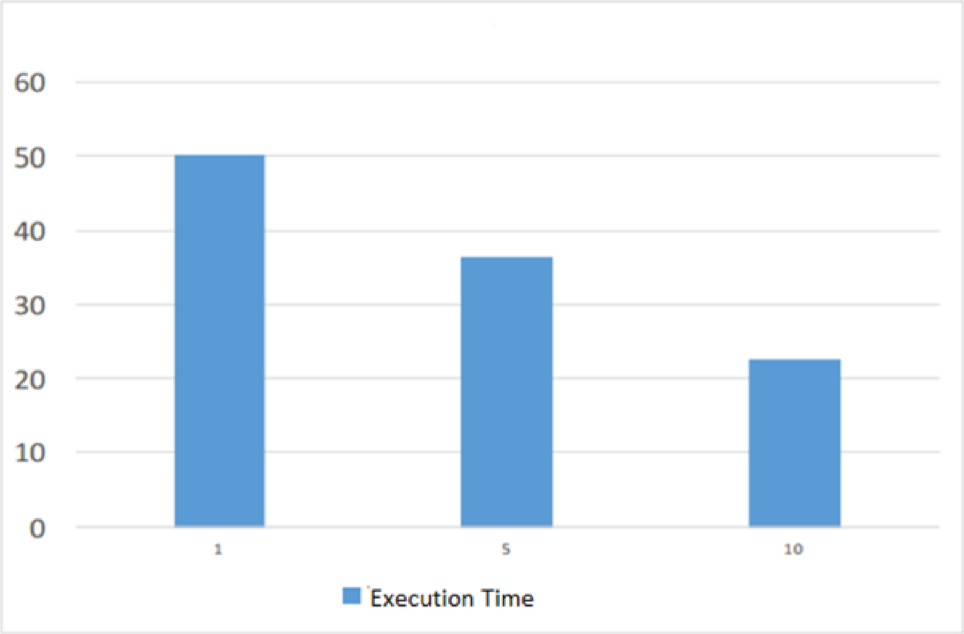




First running of DBSCAN-MR (≈ 50 mins) Tenth running of DBSCAN-MR (≈ 20 mins)

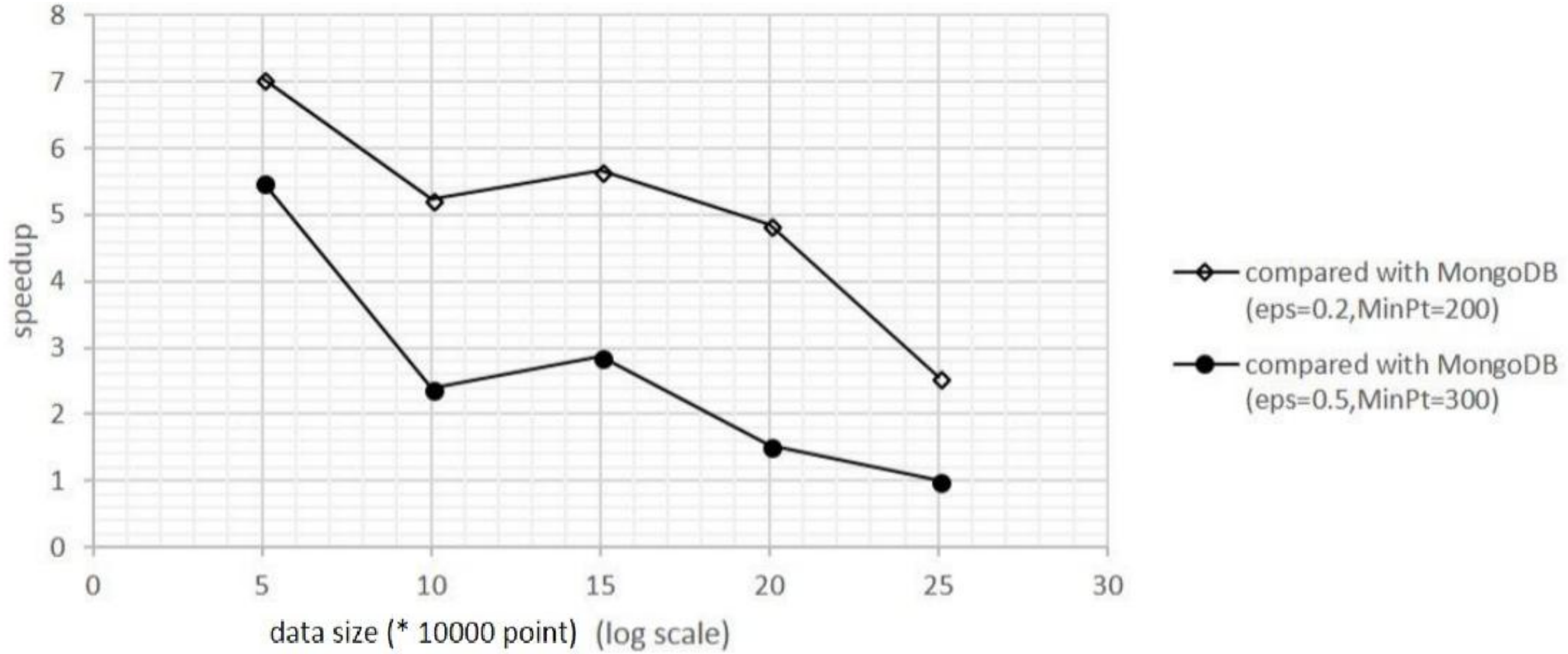
- CPU load during running sessions 1 and 10
- For 5 nodes configuration, and a dataset consisting of 250000 spatial entries

Query Performance Optimization



Execution time ***continuously improves*** along different processing rounds and goes **from 50.1 minutes** (first iteration) **to 22.7 minutes** (tenth iteration), obtaining a percentage of speed-up improvement equivalent to 54.7%

GeoSpark vs MongoDB



- Performance applying our GeoSpark support compared with DBSCAN-MR implementation over MongoDB
- Speedup gain degrades as we increase the data size,

Conclusions & Ongoing Works

- **Conclusions**
  - Current ***support of big data tools for compute-intensive geospatial big data*** sets is still rather ***poor***
  - Our framework efficiently supports **querying** and **analyzing big geospatial data** and was plugged on top of GeoSpark, with a motivation to optimize the performance of DBSCAN-MR clustering
- **Future works**
  - Incorporating additional services such as integrated geospatial-aware machine learning and data mining service
  - Extending our framework so to enable ***online processing of geospatial data streams*** in ***STARK***

Thank you!
In-memory Spatial-Aware Framework for Processing Proximity-Alike Queries in Big Spatial Data

Luca Foschini, Assistant Professor
DISI - University of Bologna
Viale del Risorgimento, 2 - 40136 Bologna - ITALY
Email: luca.foschin@unibo.it
Web: https://www.unibo.it/sitoweb/luca.foschini

Contacts

http://lia.deis.unibo.it/Staff/LucaFoschini/
http://middleware.unibo.it/

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
DIPARTIMENTO DI INFORMATICA - SCIENZA E INGEGNERIA