Simba: Efficient In-Memory Spatial Analytics.

Dong Xie, Feifei Li, Bin Yao, Gefei Li, Liang Zhou and Minyi Guo

SIGMOD'16.

Andres Calderon

November 9, 2016

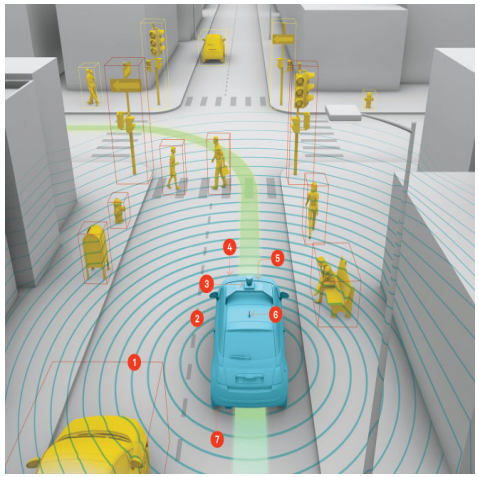
Agenda

1. Background
2. Simba Architecture Overview
   - Programming Interface
   - Indexing
   - Spatial Operations
   - Optimization
3. Experiments
4. Conclusions

Agenda

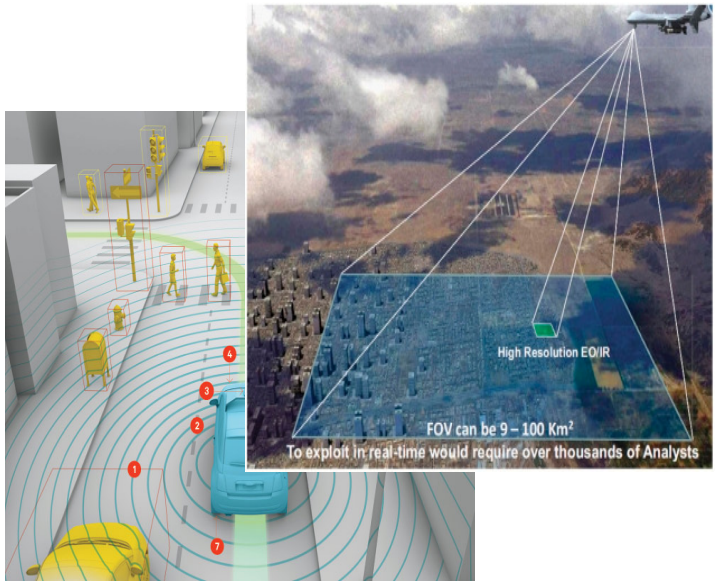
Introduction

- There has been an explosion in the amount of spatial data in recent years...



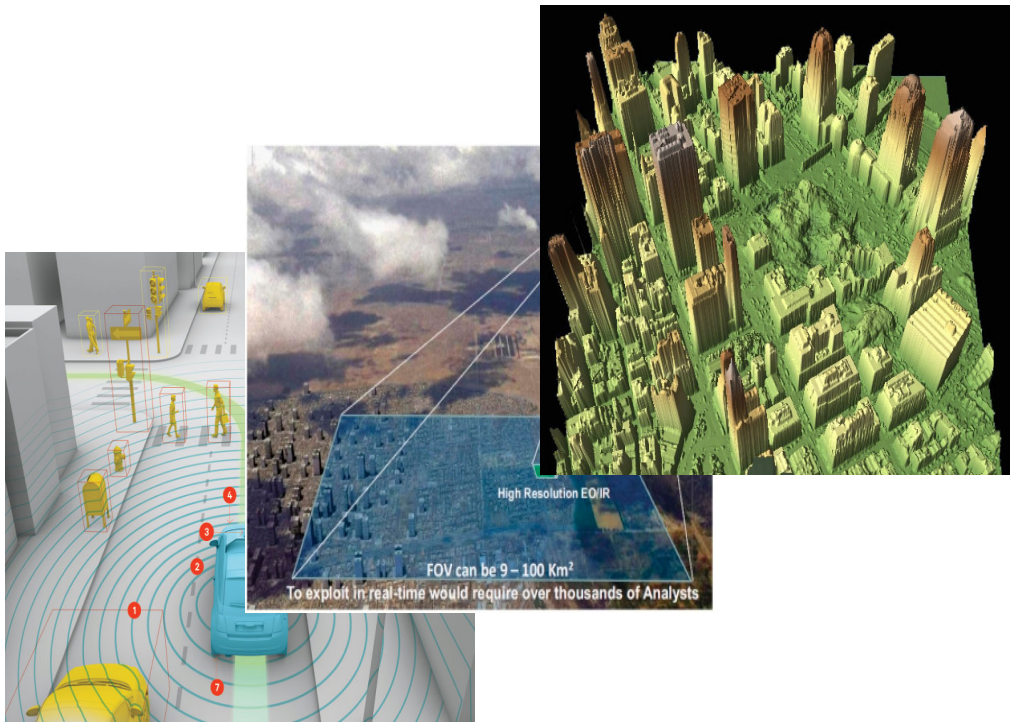
Introduction

- There has been an explosion in the amount of spatial data in recent years...



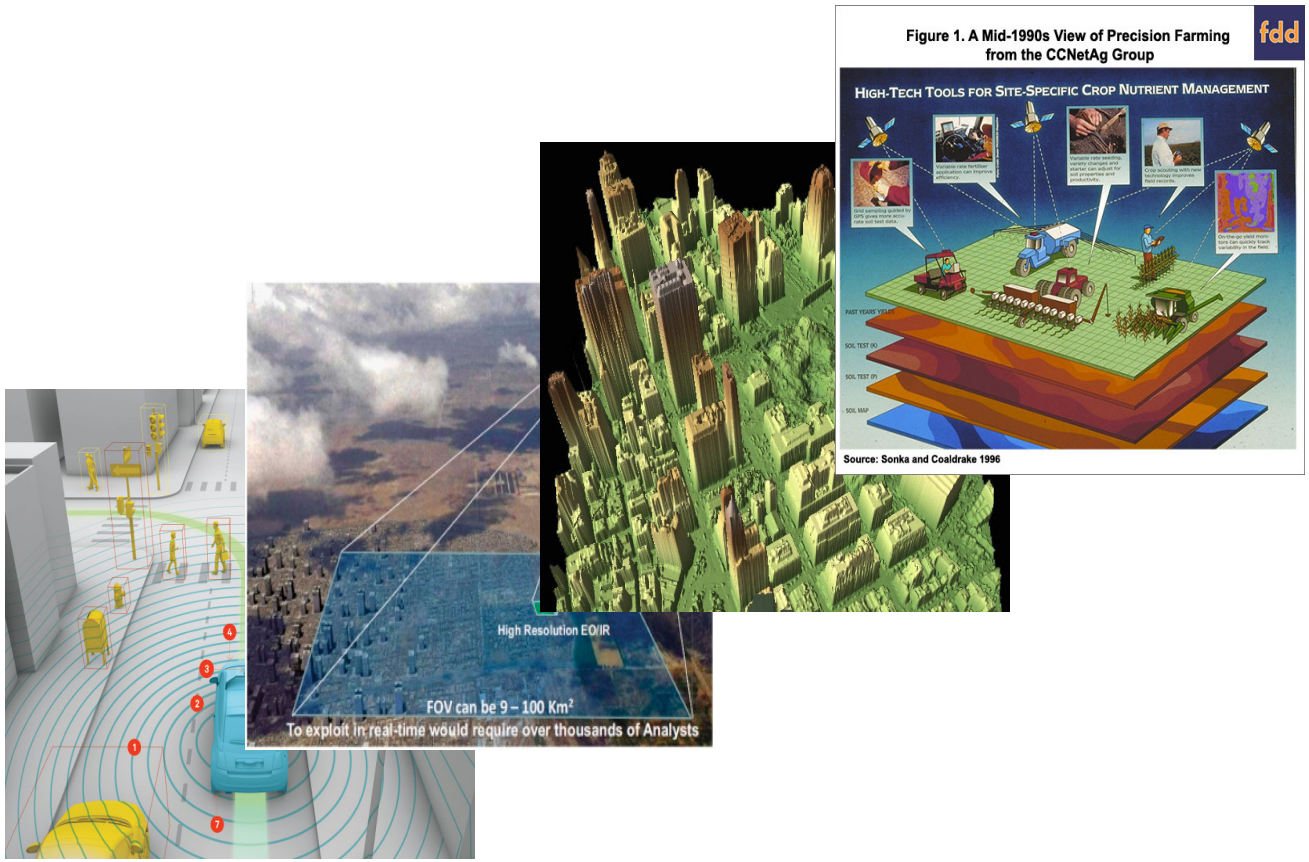
Introduction

- There has been an explosion in the amount of spatial data in recent years...



Introduction

- There has been an explosion in the amount of spatial data in recent years...



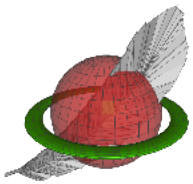
Introduction

- The applications and commercial interest is clear...

Introduction

- But remember that "Spatial is Special"...

ORACLE SPATIAL



MySQL

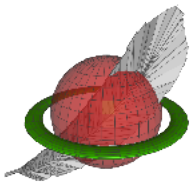
ArcGIS
ESRI

Microsoft SQL Server

Introduction

- But remember that "Spatial is Special"...

ORACLE SPATIAL



MySQL

ArcGIS
ESRI

Microsoft SQL Server

Hadoop-GIS
Spatial Big Data Solutions

MD-Hbase

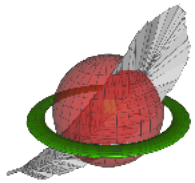
SECONDO

geomesa

Introduction

- But remember that "Spatial is Special"...

ORACLE SPATIAL



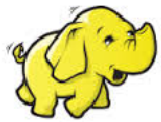
MySQL

ArcGIS
ESRI

Microsoft SQL Server

Hadoop-GIS
Spatial Big Data Solutions

MD-Hbase



SECONDO

geomesa

GeoSpark

GeoTrellis

Spark SQL

SpatialSpark

MEMSQL

Introduction

- Why do we need a new tool???

Introduction

- Problems of Existing Systems...
  - Single node database (low scalability)
    ArcGIS, PostGIS, Oracle Spatial.
  - Disk-oriented cluster computation (low performance)
    Hadoop-GIS, SpatialHadoop, GeoMesa.
  - No native support for spatial operators
    Spark SQL, MemSQL
  - No sophisticated query planner and optimizer
    SpatialSpark, GeoSpark

Introduction

- Simba: **S**patial **I**n **M**emory **B**ig data **A**nalytics.
  1. Extends Spark SQL to support spatial queries and offers simple APIs for both SQL and DataFrame.
  2. Support two-layer spatial indexing over RDDs (low latency).
  3. Designs a SQL context to run important spatial operations in parallel (high throughput).
  4. Introduces spatial-aware and cost-based optimizations to select good spatial plans.

Introduction

- Simba: **S**patial **I**n **M**emory **B**ig data **A**nalytics.
  1. Extends Spark SQL to support spatial queries and offers simple APIs for both SQL and DataFrame.
  2. Support two-layer spatial indexing over RDDs (low latency).
  3. Designs a SQL context to run important spatial operations in parallel (high throughput).
  4. Introduces spatial-aware and cost-based optimizations to select good spatial plans.

Introduction

- Simba: **S**patial **I**n **M**emory **B**ig data **A**nalytics.
  1. Extends Spark SQL to support spatial queries and offers simple APIs for both SQL and DataFrame.
  2. Support two-layer spatial indexing over RDDs (low latency).
  3. Designs a SQL context to run important spatial operations in parallel (high throughput).
  4. Introduces spatial-aware and cost-based optimizations to select good spatial plans.

Introduction

- Simba: **S**patial **I**n **M**emory **B**ig data **A**nalytics.
  1. Extends Spark SQL to support spatial queries and offers simple APIs for both SQL and DataFrame.
  2. Support two-layer spatial indexing over RDDs (low latency).
  3. Designs a SQL context to run important spatial operations in parallel (high throughput).
  4. Introduces spatial-aware and cost-based optimizations to select good spatial plans.

Introduction

Core Features	Simba	GeoSpark	SpatialSpark	SpatialHadoop	Hadoop GIS
Data dimensions	multiple	$d \leq 2$	$d \leq 2$	$d \leq 2$	$d \leq 2$
SQL	✓	×	×	Pigeon	×
DataFrame API	✓	×	×	×	×
Spatial indexing	R-tree	R-/quad-tree	grid/kd-tree	grid/R-tree	SATO
In-memory	✓	✓	✓	×	×
Query planner	✓	×	×	✓	×
Query optimizer	✓	×	×	×	×
Concurrent query execution	thread pool in query engine	user-level process	user-level process	user-level process	user-level process
query operation support					
Box range query	✓	✓	✓	✓	✓
Circle range query	✓	✓	✓	×	×
k nearest neighbor	✓	✓	only 1NN	✓	×
Distance join	✓	✓	✓	via spatial join	✓
kNN join	✓	×	×	×	×
Geometric object	×[1]	✓	✓	✓	✓
Compound query	✓	×	×	✓	×

Table 1: Comparing Simba against other systems.

Agenda

1. Background
2. Simba Architecture Overview
   - Programming Interface
   - Indexing
   - Spatial Operations
   - Optimization
3. Experiments
4. Conclusions

Spark SQL Overview

Spark SQL is Apache Spark's module for working with structured data.

- Seamlessly mixes SQL queries with Spark programs.
- Connects to any data source the same way.
- Includes a highly extensible cost-based optimizer (*Catalyst*).
- Spark SQL is a full-fledged query engine based on the underlying Spark core.

Spark SQL Overview

Spark SQL is Apache Spark's module for working with structured data.

- Seamlessly mixes SQL queries with Spark programs.
- Connects to any data source the same way.
- Includes a highly extensible cost-based optimizer (*Catalyst*).
- Spark SQL is a full-fledged query engine based on the underlying Spark core.

Spark SQL Overview

Spark SQL is Apache Spark's module for working with structured data.

- Seamlessly mixes SQL queries with Spark programs.
- Connects to any data source the same way.
- Includes a highly extensible cost-based optimizer (*Catalyst*).
- Spark SQL is a full-fledged query engine based on the underlying Spark core.

Spark SQL Overview

Spark SQL is Apache Spark's module for working with structured data.

- Seamlessly mixes SQL queries with Spark programs.
- Connects to any data source the same way.
- Includes a highly extensible cost-based optimizer (*Catalyst*).
- Spark SQL is a full-fledged query engine based on the underlying Spark core.

Spark SQL Overview

Spark SQL is Apache Spark's module for working with structured data.

- Seamlessly mixes SQL queries with Spark programs.
- Connects to any data source the same way.
- Includes a highly extensible cost-based optimizer (*Catalyst*).
- Spark SQL is a full-fledged query engine based on the underlying Spark core.

Spark SQL Overview

```
# Apply functions to results of SQL queries.
context = HiveContext(sc)
results = context.sql("""
                        SELECT
                              *
                        FROM
                              people""")
names = results.map(lambda p: p.name)
# Query and join different data sources.
context.jsonFile("s3n://...").registerTempTable("json")
results = context.sql("""
                        SELECT
                             *
                        FROM
                             people
                        JOIN
                             json ...""")
```

Simba Architecture

Simba is an extension of Spark SQL across the system stack.

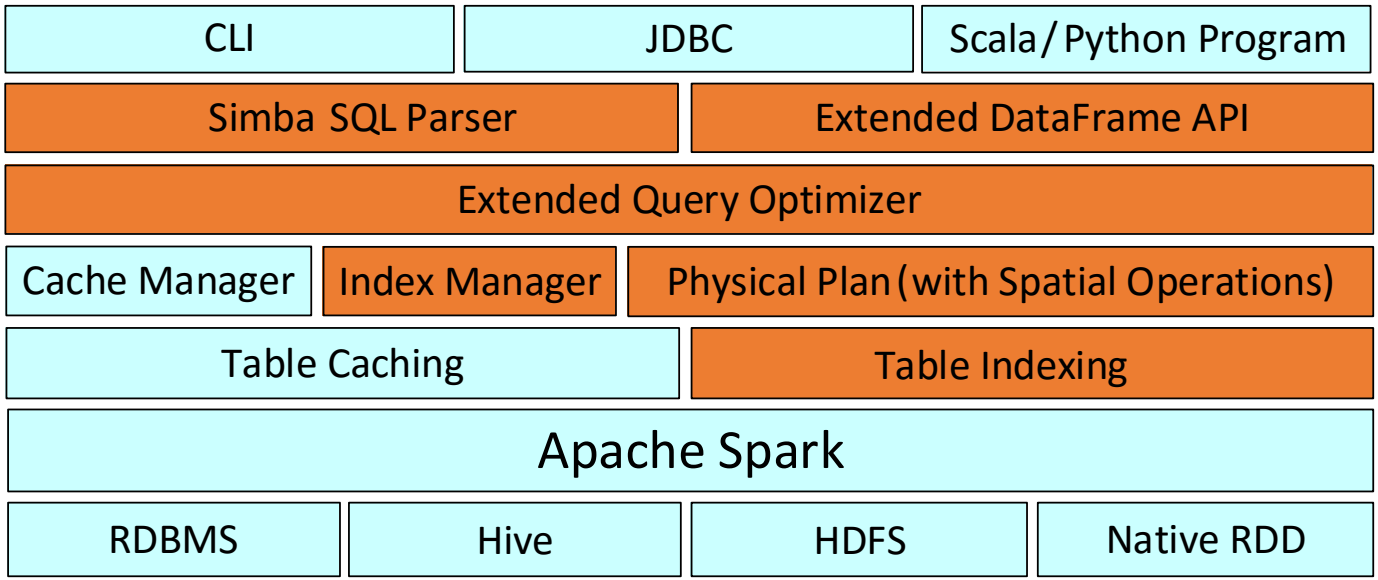


Figure 1: Simba architecture.

Simba Architecture

Simba is an extension of Spark SQL across the system stack.

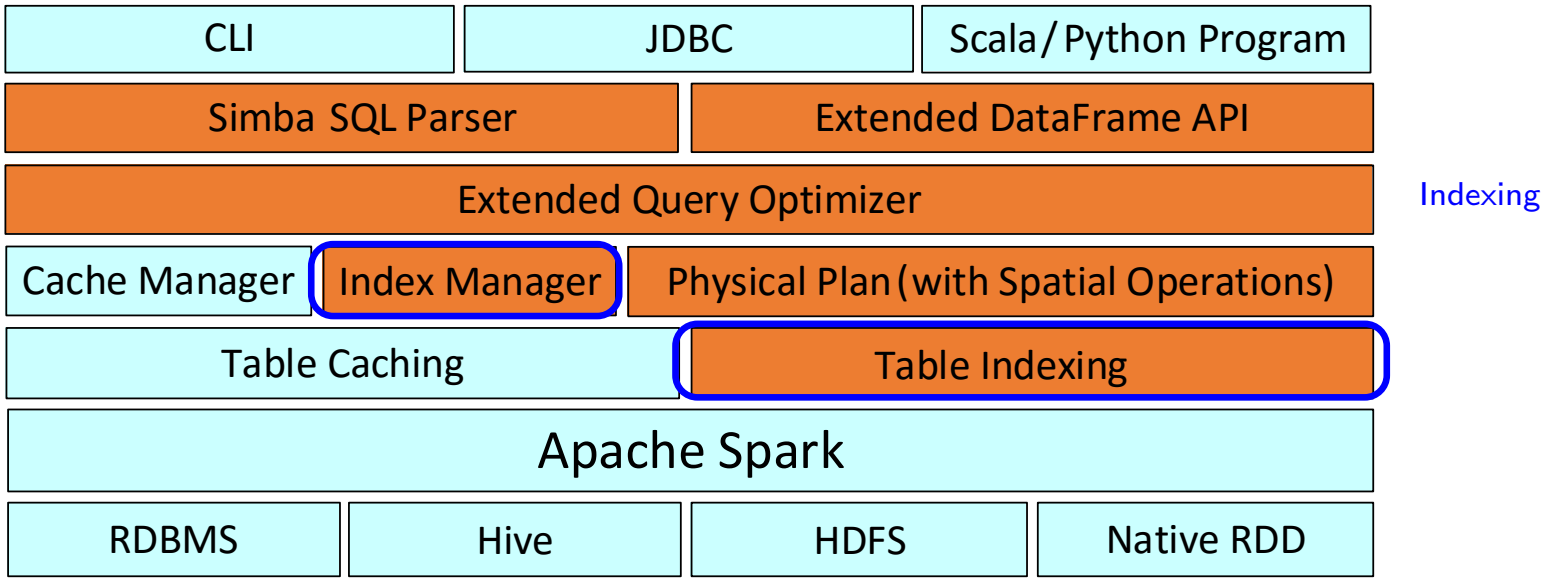


Figure 1: Simba architecture.

Simba Architecture

Simba is an extension of Spark SQL across the system stack.

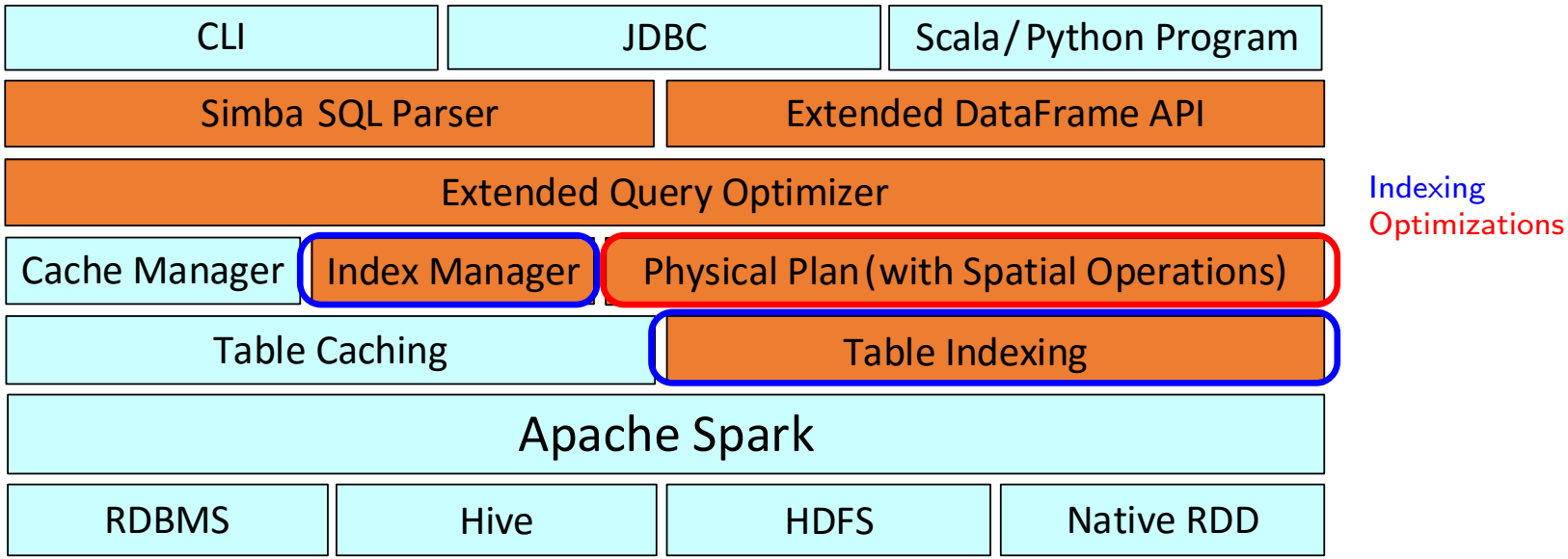


Figure 1: Simba architecture.

Simba Architecture

Simba is an extension of Spark SQL across the system stack.

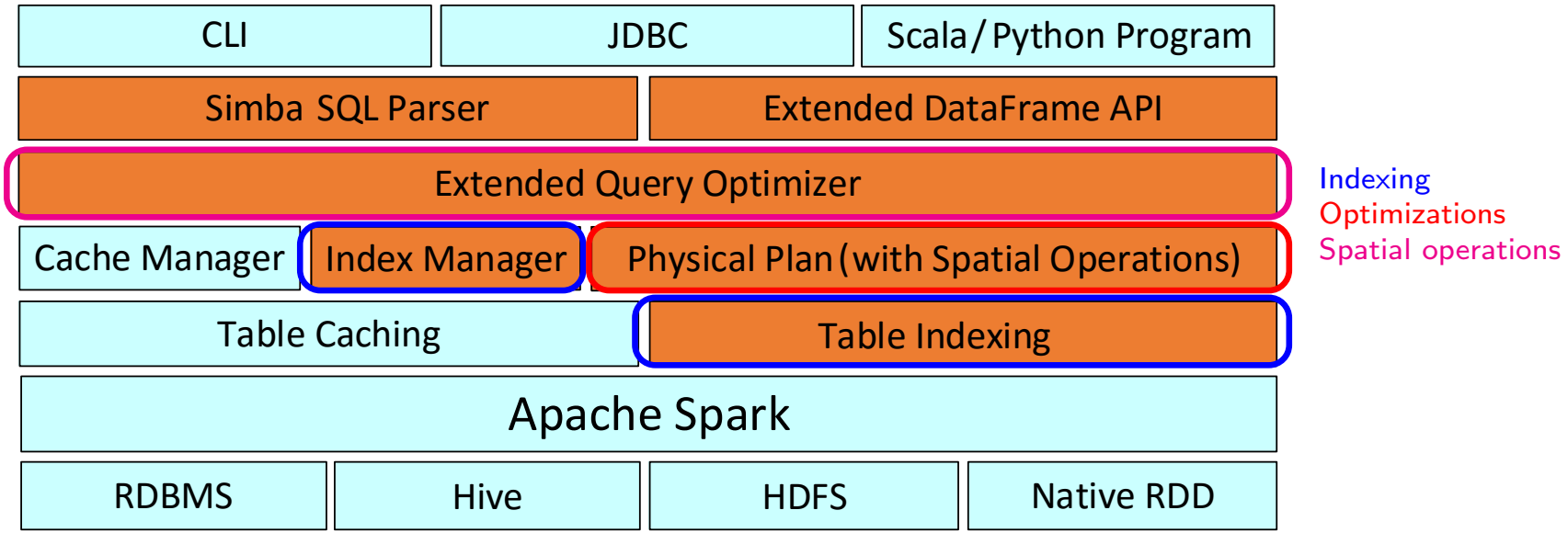


Figure 1: Simba architecture.

Simba Architecture

Simba is an extension of Spark SQL across the system stack.

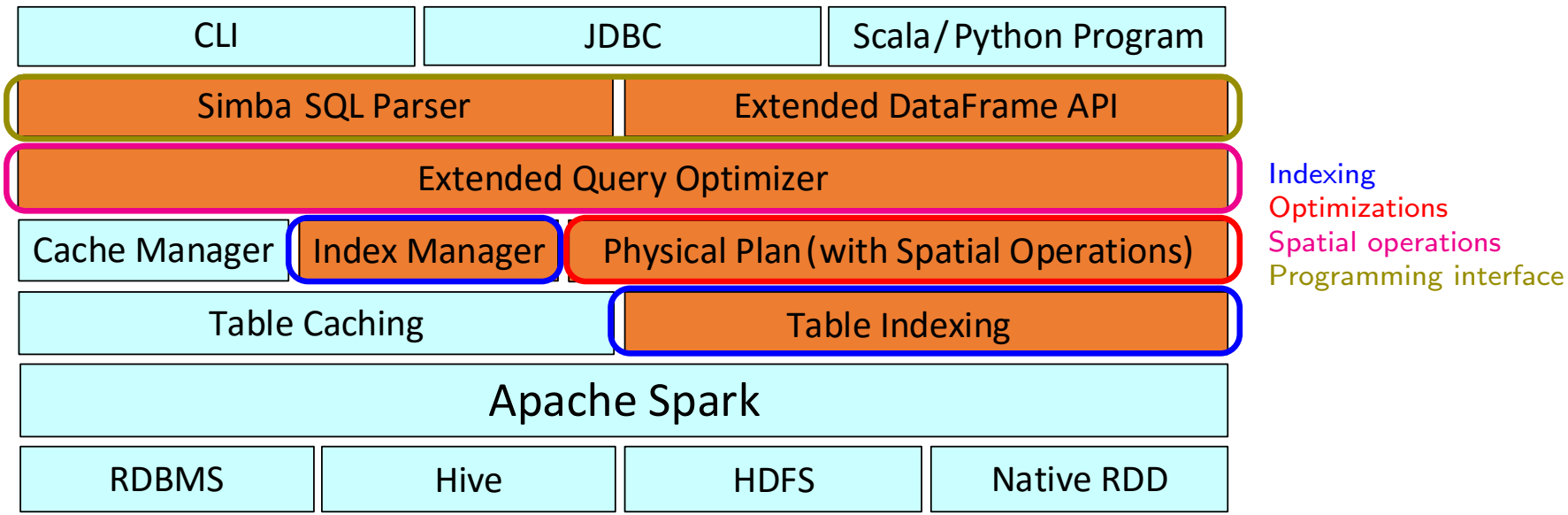


Figure 1: Simba architecture.

Query Load in Simba

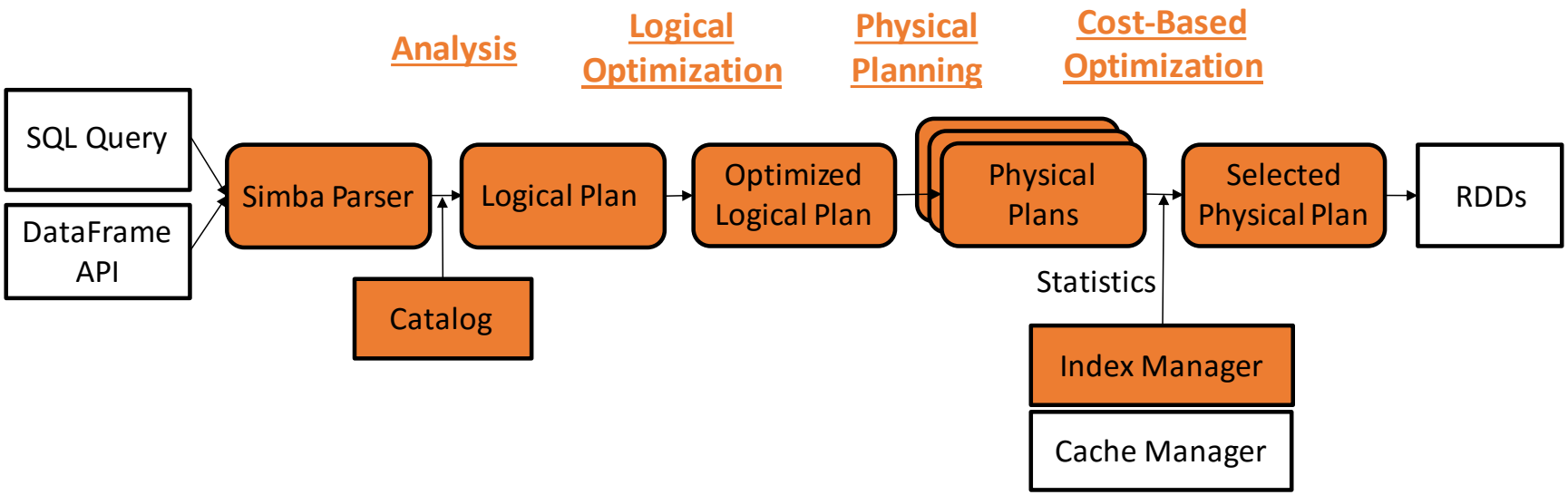


Figure 2: Query processing workflow in Simba.

Agenda

Programming Interface

Support rich query types natively in the kernel...

```
SELECT
      *
FROM
      points
SORT BY
      (x - 2) * (x - 2) +
      (y - 3) * (y - 3)
LIMIT
      5
```

$\Longrightarrow$

```
SELECT
      *
FROM
      points
WHERE
      POINT(x, y) IN
            KNN(POINT(2, 3), 5)
```

Programming Interface

Achieve something that is impossible in Spark SQL...

```
SELECT
      *
FROM
      queries q
KNN JOIN
      pois p
ON
      POINT(p.x, p.y) IN KNN(POINT(q.x, q.y), 3)
```

Programming Interface

Fully compatible with standard SQL operators...

```
SELECT
      p.id, count(*) AS n
FROM
      pois AS p
DISTANCE JOIN
      restaurants AS r
ON
      POINT(r.lat, r.lng) IN CIRCLERANGE(POINT(p.lat, p.lng), 200.0)
GROUP BY
      p.id
ORDER BY
      n
```

Programming Interface

Same level of flexibility for DataFrames...

```
pois.distanceJoin(restaurants, Point(pois("lat"), pois("lng")),
                  Point(restaurants("lat"), restaurants("lng")), 3.0)
      .groupBy(pois("id"))
      .agg(count("*").as("n"))
      .sort("n").show()
```

Agenda

Table Indexing

- In Spark SQL:
  - `Record` $\rightarrow$ `Row`
  - `Table` $\rightarrow$ `RDD[Row]`
- Spark SQL makes a full scan of RDDs.
  - Inefficient for spatial queries!!!
- Solution: native **two-level** indexing over RDDs

Table Indexing

- IndexRDD
  - Pack all Row objects within a RDD partition into an array (O(1) cost for access).
  - IPartition data structure:
    - `case class IPartition[Type](Data: Array[Type], I: Index)`
    - Index can be HashMap, TreeMap or RTree.
  - So, by using Type=Row:
    - `type IndexRDD[Row] = RDD[IPartition[Row]]`

Two-level indexing strategy

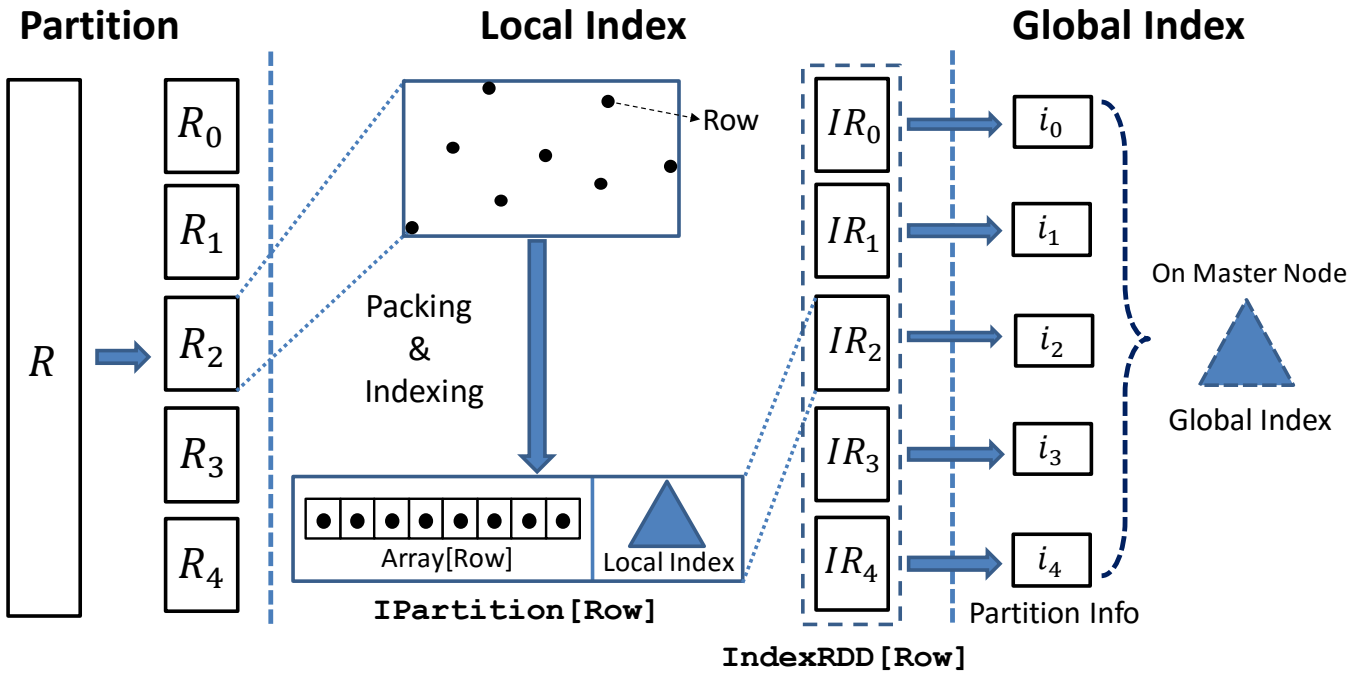


Figure 4: Two-level indexing strategy in Simba.

```
CREATE INDEX idx_name ON R(x_1, ..., x_m) USE idx_type
DROP INDEX idx_name ON table_name
```

Agenda

Agenda

Agenda

Agenda

Conclusions

Future ideas

Thank you!!!

Do you have any question?