# FINAL YEAR PROJECT

## SECOND EVALUATION

2019 - 2020

DEVAL MUDIA (BT16CSE060)
AYUSH SINGH (BT16CSE098)
CHAITYA CHHEDA (BT16CSE016)
SADNEYA PUSALKAR (BT16CSE076)

UNDER THE GUIDANCE OF
Dr. S.R. SATHE

# Parallelised Recommendation System using Spark and OpenMP

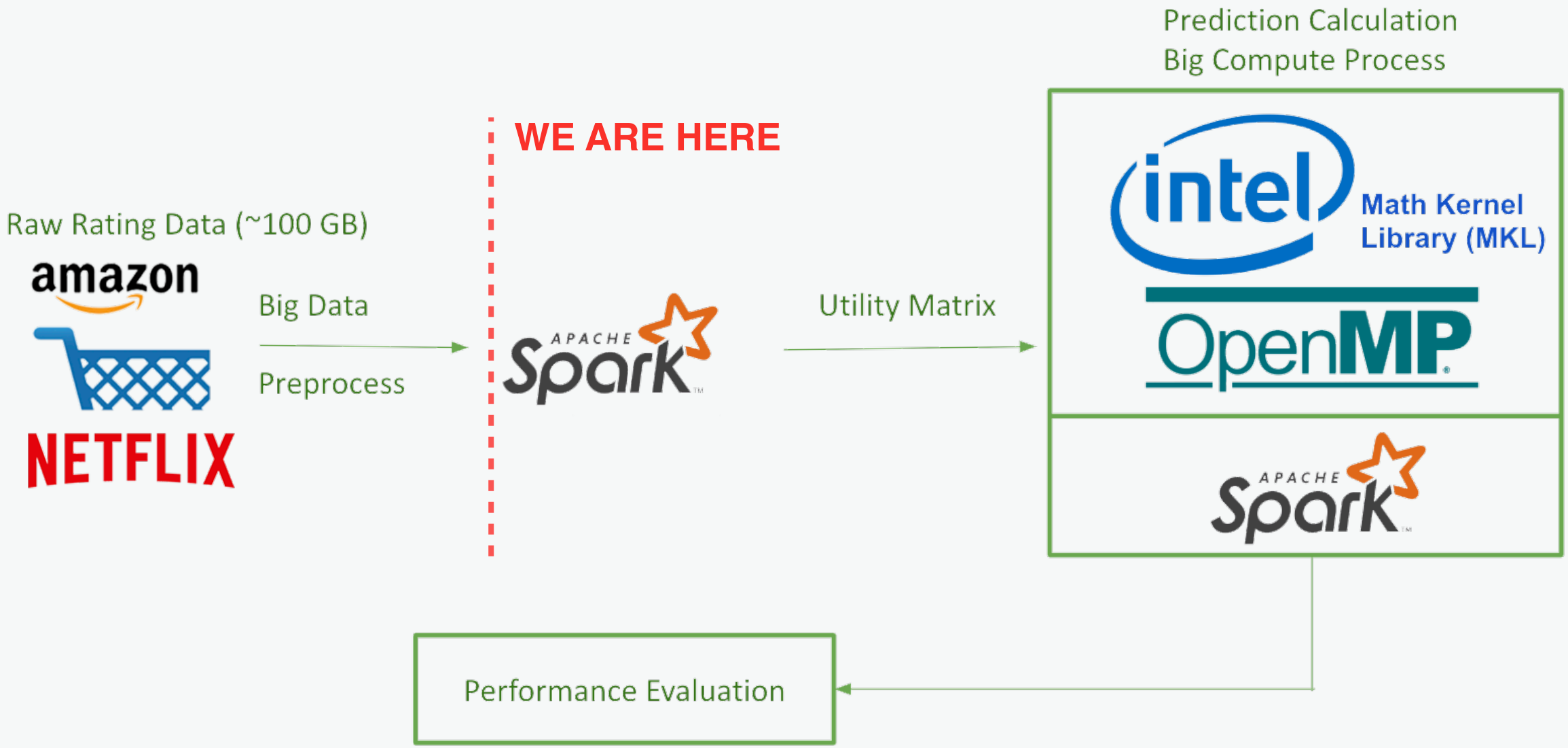
## Big Data

Dealing with a large, unstructured dataset and in order to process it into a matrix, we would need to make use of **big data processing** solutions such as **Spark.**

## Big Compute

In order to compute similarity scores and generate predictions, ***matrix or vector operations*** can be made parallel through **big compute** using multi-threading to speed up these computations.

# PLATFORM AND INFRASTRUCTURE



# DATASET

**Amazon Product Data (UCSD)** : http://jmcauley.ucsd.edu/data/amazon/

- Book reviews - 5 Crore (8,898,041 reviews) : **7GB**

## Raw JSON Data


```
{
  "reviewerID": "A2SUAM1J3GNN3B",
  "asin": "0000013714",
  "reviewerName": "J. McDonald",
  "helpful": [2, 3],
  "reviewText": "Great purchase!",
  "overall": 5.0,
  "summary": "Heavenly Highway Hymns",
  "unixReviewTime": 1252800000,
  "reviewTime": "09 13, 2009"
}
```


## Goal

| | P1 | P2 | P3 | P4 | P5 | P6 |
|---|---|---|---|---|---|---|
| U1 | 3 | | 4 | 5 | | |
| U2 | 1 | | | | 5 | |
| U3 | | | 4 | | | 3 |
| U4 | | 5 | 2 | | 5 | |
| U5 | 3 | | | 5 | | 4 |

# DATA PREPROCESSING

## Raw JSON Data


```
{
  "reviewerID": "A2SUAM1J3GNN3B",
  "asin": "0000013714",
  "reviewerName": "J. McDonald",
  "helpful": [2, 3],
  "reviewText": "It looks good and solve my problems",
  "overall": 5.0,
  "summary": "Heavenly Highway Hymns",
  "unixReviewTime": 1252800000,
  "reviewTime": "09 13, 2009"
}
```


## Index Matrix

| asin | id |
|---|---|
| | |
| | |

| reviewer | id |
|---|---|
| | |
| | |

## Utility Matrix

| | Product 1 | Product 2 |
|---|---|---|
| Reviewer 1 | 4.0 | 5.0 |
| Reviewer 2 | 5.0 | 1.0 |
| Reviewer 3 | 3.0 | 4.0 |

# IMPLEMENTATION

## System Requirements

**SCALA**
Scala is a general-purpose programming language providing support for functional programming and a strong static type system.

**Install Scala**

1. Java 8 JDK (also known as 1.8) should be installed.

2. Install sbt

# IMPLEMENTATION

## APACHE SPARK

Apache Spark is a fast and general-purpose cluster computing system that supports a rich set of higher-level tools including Spark SQL for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming.

### Install Spark

1. Choose the Spark Release 2.4.4
2. Download a pre-built for Hadoop 2.7 version of Spark (preferably Spark 2.0 or later).

   *spark-2.4.4-bin-hadoop2.7.tgz*

3. Use the Spark Shell to load .scala scripts.

   *./spark-shell*

# IMPLEMENTATION

```
Miguels-MacBook-Pro:~ migscabral$ scala -version
Scala code runner version 2.12.4 -- Copyright 2002-2017, LAMP/EPFL and Lightbend, Inc.
Miguels-MacBook-Pro:~ migscabral$ spark-shell --version
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.2.0
      /_/

Using Scala version 2.11.8, Java HotSpot(TM) 64-Bit Server VM, 1.8.0_152
Branch
Compiled by user jenkins on 2017-06-30T22:58:04Z
Revision
Url
Type --help for more information.
Miguels-MacBook-Pro:~ migscabral$
```

# WHY SCALA WITH SPARK

- Apache Spark is written in Scala and because of its scalability on JVM - Scala programming is most prominently used programming language
- Syntax for Scala is less intimidating and complex when compared to JAVA or C++
- Scala has excellent built-in concurrency support and libraries like Akka which makes it easy to build a truly scalable application.
- Scala has well-designed libraries for scientific computing, linear algebra and random number generation.

# IMPLEMENTATION

## Spark Standalone Mode

Right now we are using Spark in simple standalone deploy mode.

Start a standalone master server by executing:

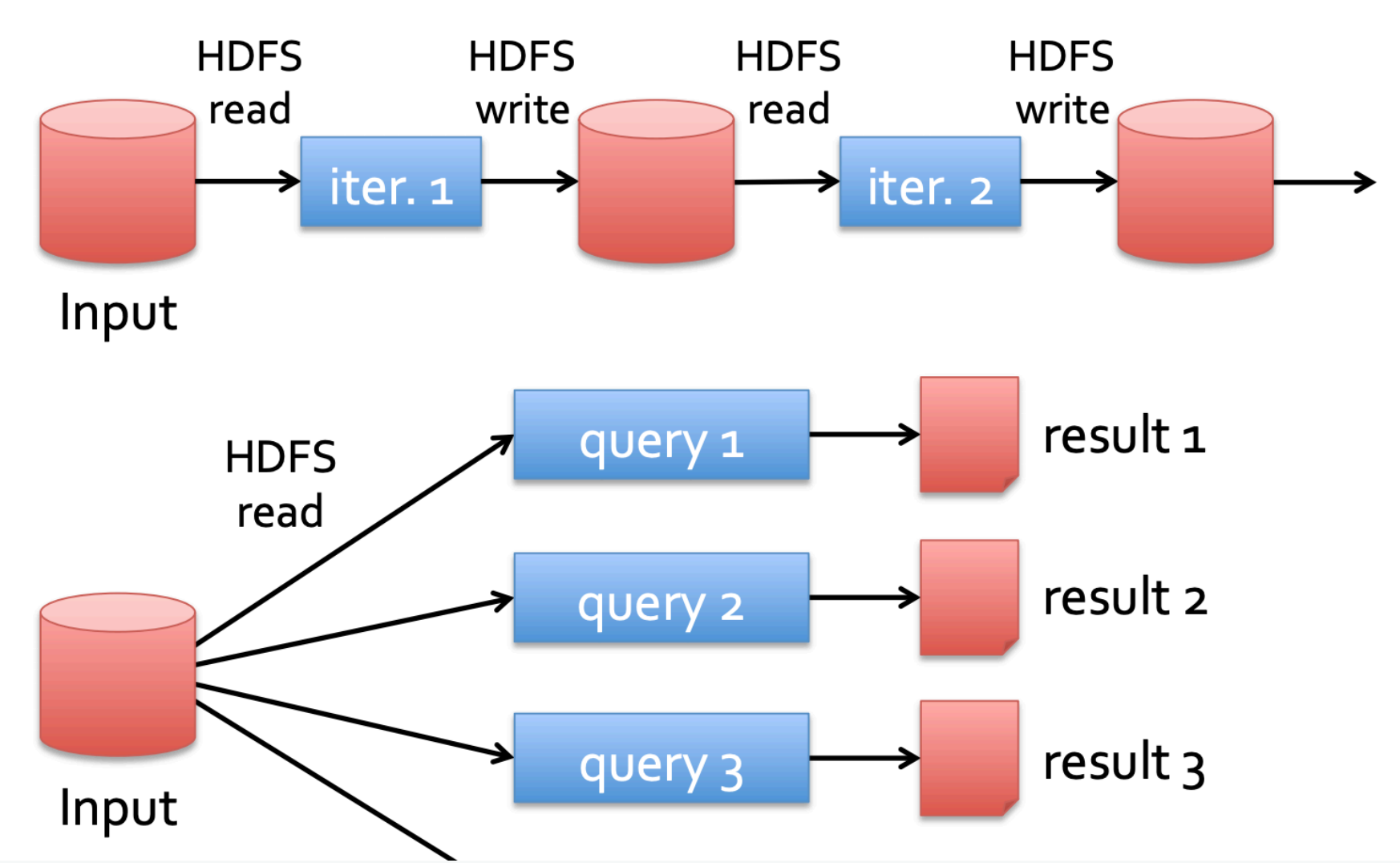
```
./sbin/start-master.sh
```

## RDD : Resilient Distributed Dataset

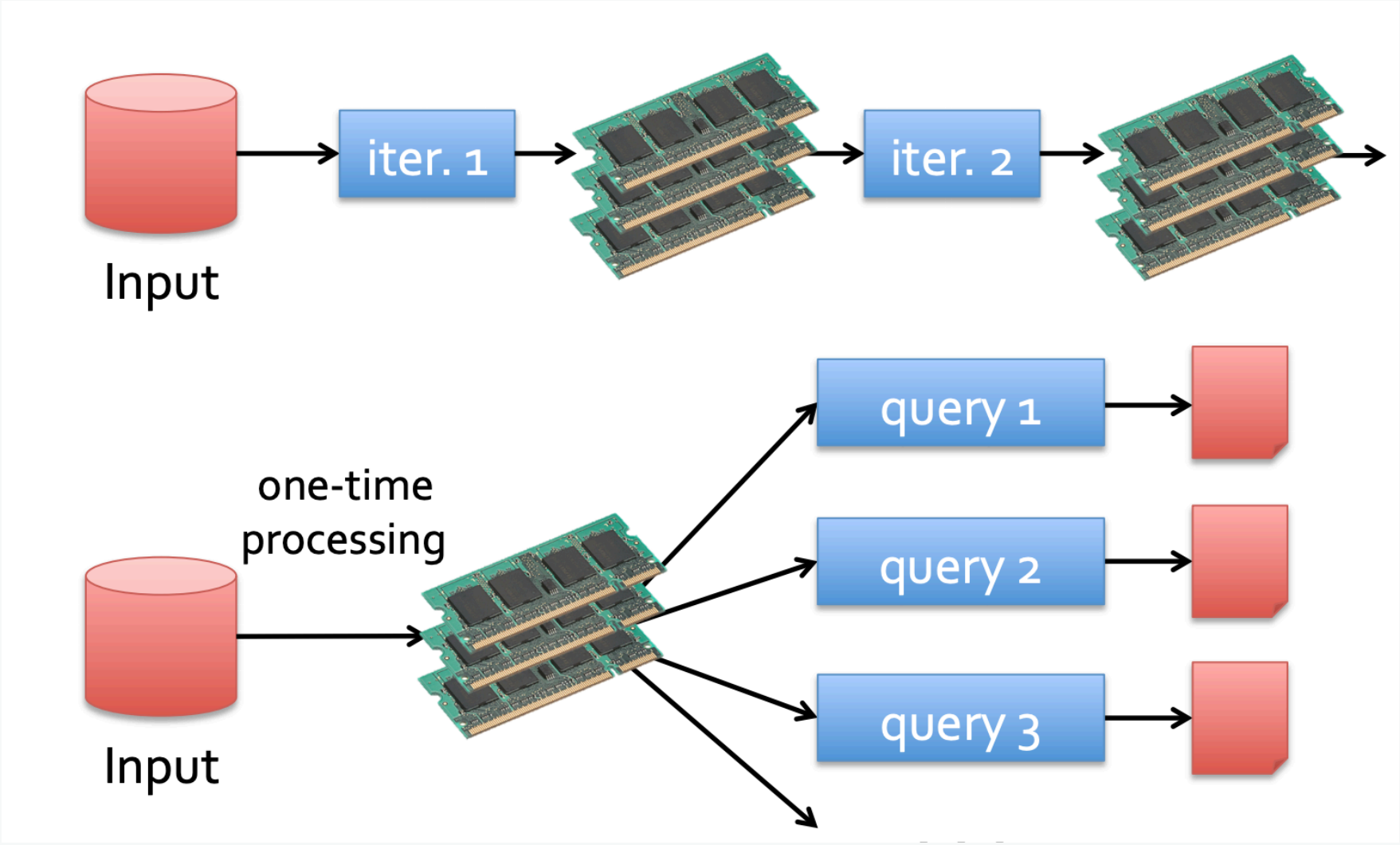
Fundamental data structure of Apache Spark. RDD is an immutable collection of objects which computes on the different node of the cluster.

Each and every dataset in RDD is logically partitioned across many servers so that they can be computed on different nodes of the cluster.

# HADOOP


HDFS read
HDFS write
HDFS read
HDFS write
iter. 1
iter. 2
Input
HDFS read
query 1
query 2
query 3
result 1
result 2
result 3
Input

# SPARK : RDD


iter. 1
iter. 2
Input
one-time
processing
query 1
query 2
query 3
Input

# TIMELINE

## Complete Data Preprocessing

The input of this data pipeline is the raw `JSON` file containing all the metadata for a given product.

The output of this data pipeline is the utility matrix mentioned above.

## Explore OpenMP

Understand the API for Writing Multithreaded Applications

## Complete Stage 1 of Project

# THANK YOU