RDD – Overview

(Resilient Distributed Datasets*)

Nov 1st 2014
Oakland CA
By Taposh Dutta Roy

* Source: https://www.cs.berkeley.edu/~matei/papers/2012/nsdi_spark.pdf

Contents

- What is RDD
- Motivation Behind RDD
- Use Cases for RDD
- Challenges for RDD
- RDD: Solve

What is RDD

"RDDs are fault tolerant, parallel data structures that let users explicitly persist intermediate results in memory, control their partitioning to optimize data placement, and manipulate them using a rich set of operations. "

In a nutshell RDDs are a level of abstraction that enable efficient data reuse in a broad range of applications

Motivation behind RDD

Current frameworks like MapReduce & Dyrad provide a numerous abstractions for accessing a cluster's computational resources but lack abstractions for leveraging the distributed memory !!!

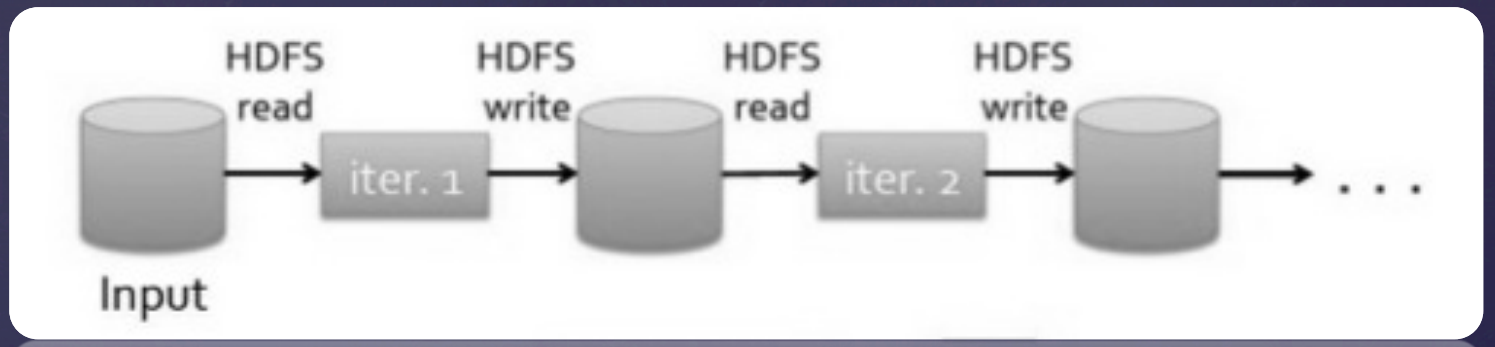
Data reuse is common in many iterative machine learning algorithms such as – Page Rank, K-means Clustering & Logistic Regression.

Motivation behind RDD

Another use case is when an user runs multiple adhoc queries on the same subset of data.
Unfortunately in current frameworks, the only way to reuse data between computations i.e between two jobs is to write to an external storage system e.g. a distributed file system such as Amazon S3.

Use cases for RDD

1. Solving Iterative problems



Existing Solution – Slow, needs high I/O

RDD - Fast, in memory

Use cases for RDD

Example: Suppose I have to look at the webserver access logs and look for an error_code or certain text.

```
TODAY=$(date +"%Y-%m-%d")

/home/ec2-user/elastic-mapreduce-cli/elastic-mapreduce \
--create \
--visible-to-all-users \
--name "Prod_data_issue_${BUILD_TAG}" \
--log-uri s3n://mypath.myserver.com/data_analysis \
--instance-group master --instance-type m1.small --instance-count 1 \
--instance-group core --instance-type m1.small --instance-count 2 \
--stream \
--input s3://mypath.myserver.com/test_data/test \
--mapper /bin/cat \
--reducer "/bin/grep '${TEXT}'" \
--output s3://mypath.myserver.com/test_data/test/${DATE} \
--with-termination-protection
```

Use cases for RDD

Example (cont'd) : I run the above code on server which returns a set of files with the words looked for grepped, closes the cluster and puts the file into an Amazon S3 location specified in the script.

Now we look at the result files and need to extract some other text from this file, we will need to write or use another set of map-reduce code. This might take extra time to fetch the files, process and provide the results.

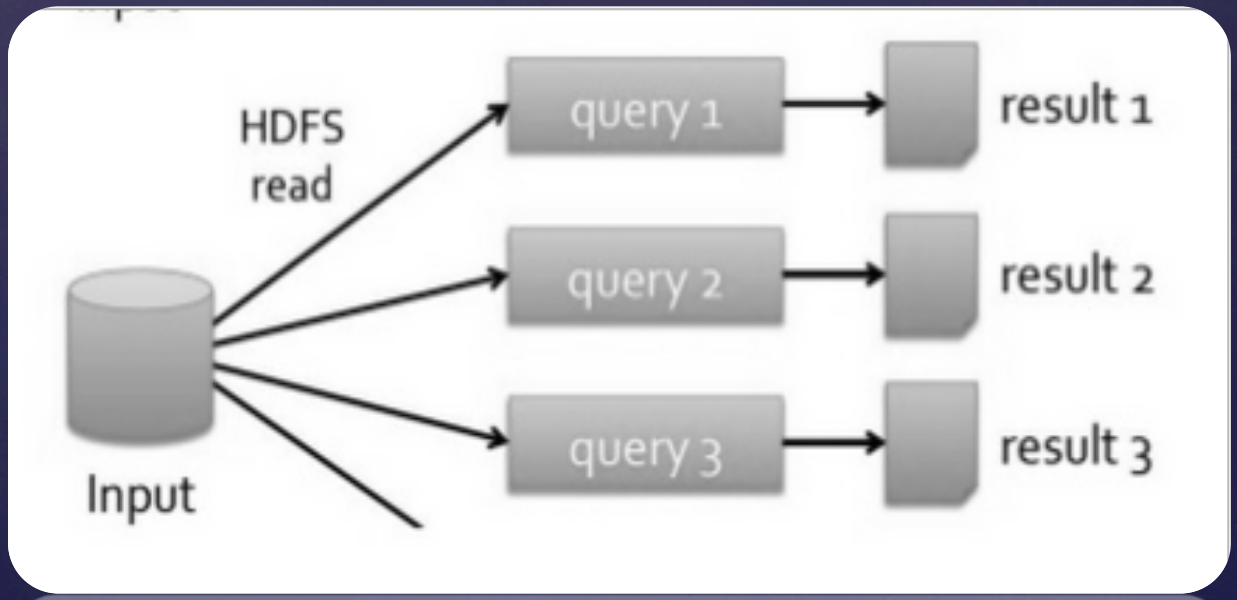
Use cases for RDD

RDD solves this problem by storing the data in memory and providing a ability for the user to requery the subset.

Use cases for RDD

2. Solving Interactive Problems

The second use case is its usage in interactive algorithms such as logistic regression which need the data to be re-used.



Challenge for RDD

The main challenge in designing RDD is defining a programming interface that can provide *fault tolerance* efficiently.

Challenges for RDD

Existing solutions such as distributed shared memory, key value stores, & databases offer an interface based on *fine-grained updates*.

With such systems, the only way to get fault tolerance is to *replicate the data across machines or to log updates across machines*. Both of these approaches are data intensive. They need high bandwidth to move the data over the cluster network and large storage.

RDD: Solve

RDD solves these probems by providing an interface based on *coarse grained* transformations such as *map, filter and join*.
These transformations apply the same operations to many data items.
This allows them to efficiently provide fault tolerance by logging the transformations used to build a dataset (i.e. lineage) rather than actual data. If a partition of RDD is lost, the RDD has enough information about how it ..

RDD: Solve

(Cont'd) was derived from other RDD to recompute just that partition. The lost data can be recovered quickly, without costly replication.

Applications not suitable : RDD

RDDs would be less suitable for applications that make asynchronous fine grained updates to shared state, such as a storage system for a web application or an incremental web crawller. For such applications traditional update logging and data checkpointing such as databases.

Conclusion RDD

RDD's goal is to provide an efficient programming model for batch analytics.

RDD has been implemented in a system called SPARK.