

COMP9313: Big Data Management

Introduction to MapReduce
and Spark

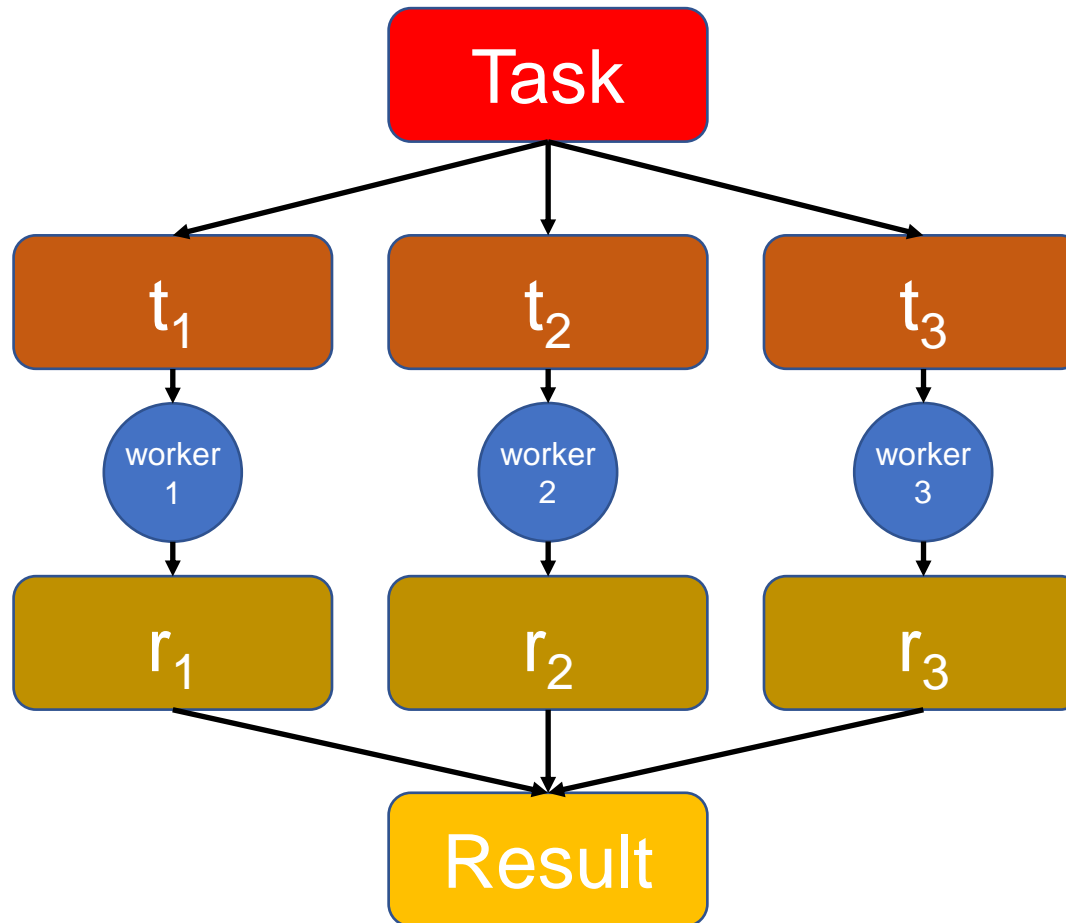
Motivation of MapReduce

- Word count
 - output the number of occurrence for each word in the dataset.
- Naïve solution:

```
word_count(D):  
    H = new dict  
    For each w in D:  
        H[w] += 1  
    For each w in H:  
        print (w, H[w])
```
- How to speed up?

Motivation of MapReduce

- Make use of multiple workers



There are some problems...

- Data reliability
 - Equal split of data
 - Delay of worker
 - Failure of worker
 - Aggregation the result
-
- **We need to handle them all!** In traditional way of parallel and distributed processing.

MapReduce

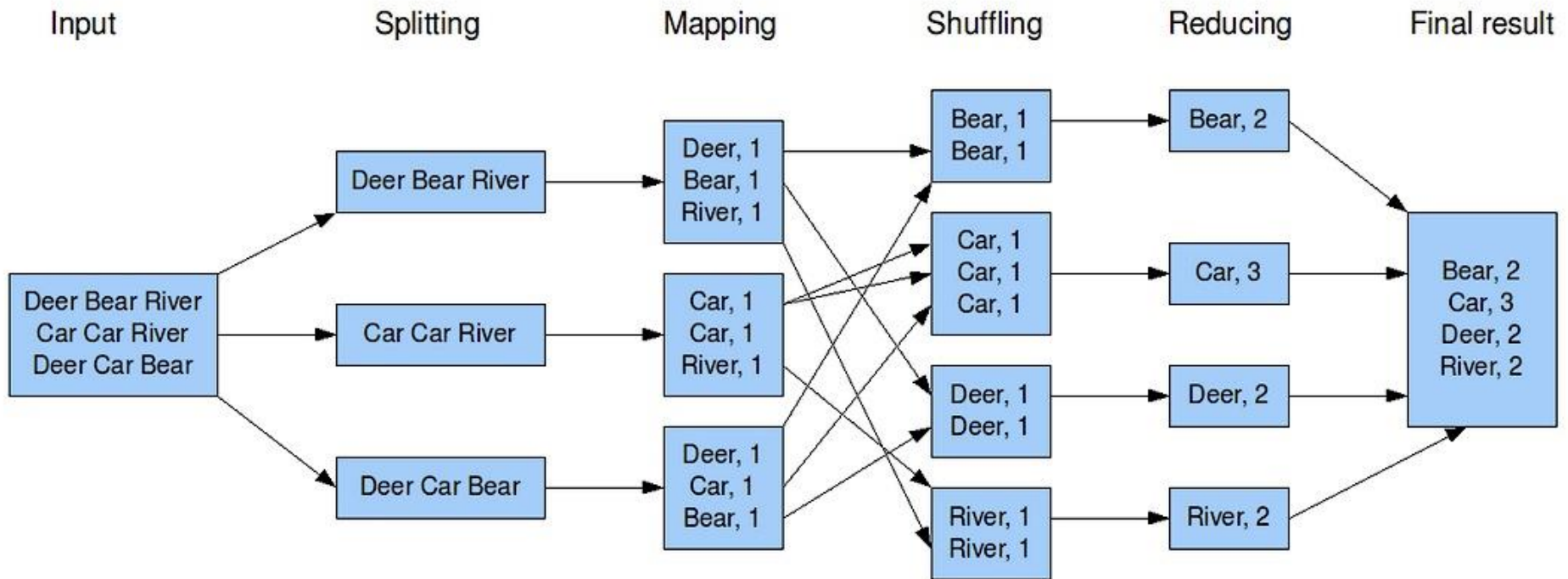
- MapReduce is a programming framework that
 - allows us to perform distributed and parallel processing on large data sets in a distributed environment
 - no need to bother about the issues like reliability, fault tolerance etc
 - offers the flexibility to write code logic without caring about the design issues of the system

Map Reduce

- MapReduce consists of Map and Reduce
- Map
 - Reads a block of data
 - Produces key-value pairs as intermediate outputs
- Reduce
 - Receive key-value pairs from multiple map jobs
 - aggregates the intermediate data tuples to the final output

A Simple MapReduce Example

The overall MapReduce word count process



Pseudo Code of Word Count

Map(D):

```
    for each w in D:  
        emit(w, 1)
```

Reduce(t, counts): # e.g., bear, [1, 1]

```
    sum = 0  
    for c in counts:  
        sum = sum + c  
    emit (t, sum)
```


Advantages of MapReduce

- Parallel processing
 - Jobs are divided to multiple nodes
 - Nodes work simultaneously
 - Processing time reduced
- Data locality
 - Moving processing to the data
 - Opposite from traditional way

We will discuss more on
MapReduce, but not now...

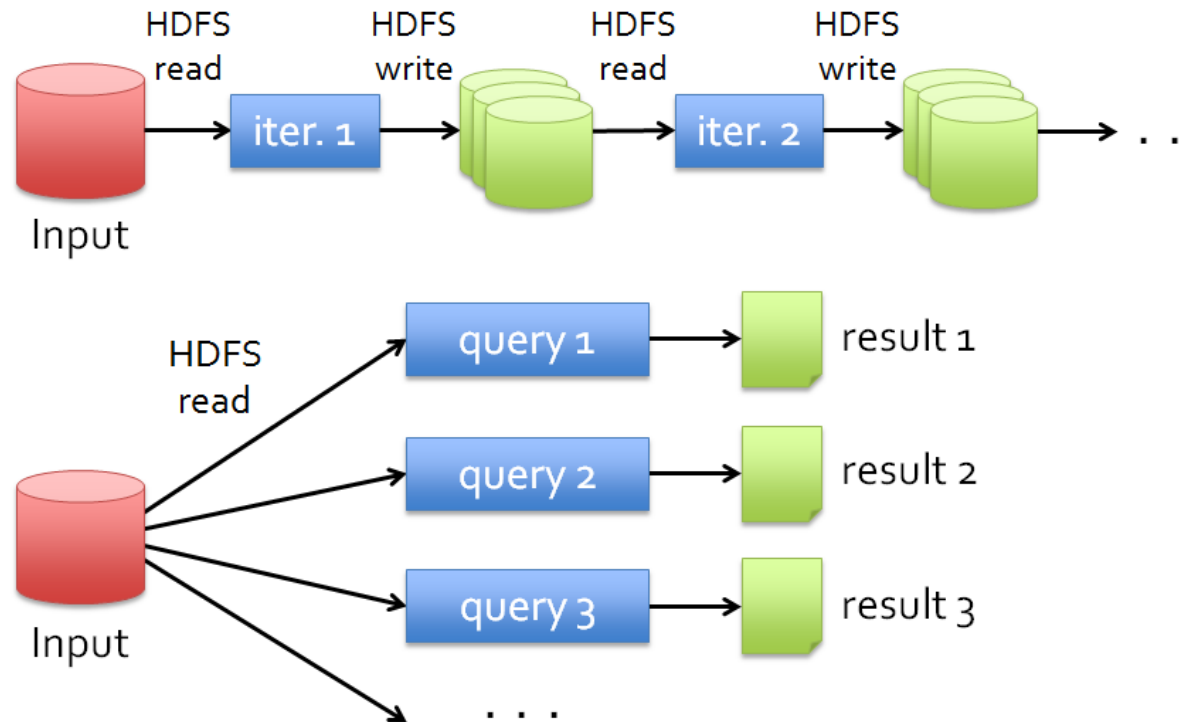
Motivation of Spark

- MapReduce greatly simplified big data analysis on large, unreliable clusters. It is great at one-pass computation.
- But as soon as it got popular, users wanted more:
 - more **complex**, multi-pass analytics (e.g. ML, graph)
 - more **interactive** ad-hoc queries
 - more **real-time** stream processing

Limitations of MapReduce

- As a general programming model:
 - more suitable for one-pass computation on a large dataset
 - hard to compose and nest multiple operations
 - no means of expressing iterative operations
- As implemented in Hadoop
 - all datasets are read from disk, then stored back on to disk
 - all data is (usually) triple-replicated for reliability

Data Sharing in Hadoop MapReduce



- **Slow** due to replication, serialization, and disk IO
- Complex apps, streaming, and interactive queries all need one thing that MapReduce lacks:
 - Efficient primitives for **data sharing**

What is Spark?

- Apache Spark is an open-source cluster computing framework for real-time processing.
- Spark provides an interface for programming entire clusters with
 - implicit data parallelism
 - fault-tolerance
- Built on top of Hadoop MapReduce
 - extends the MapReduce model to efficiently use more types of computations

Spark Features

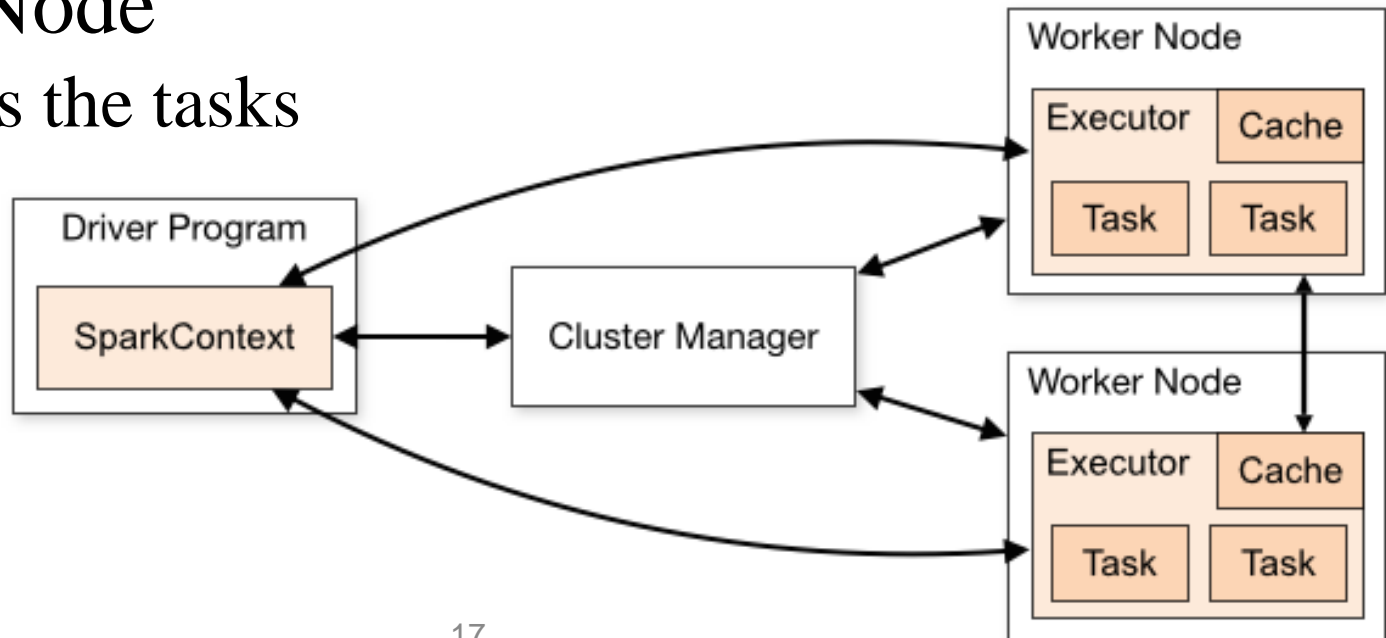
- Polyglot
- Speed
- Multiple formats
- Lazy evaluation
- Real time computation
- Hadoop integration
- Machine learning

Spark Eco-System



Spark Architecture

- Master Node
 - takes care of the job execution within the cluster
- Cluster Manager
 - allocates resources across applications
- Worker Node
 - executes the tasks



弹性分布式数据集

Resilient Distributed Dataset (RDD)

初始阶段可以理解为RDD就是python中的list

- RDD is where the data stays 存储数据
- RDD is the fundamental data structure of Apache Spark 基本数据结构
 - is a collection of elements
 - Dataset
 - can be operated on in parallel 并行运行
 - Distributed 分布式；分散式
 - fault tolerant 容错法
 - Resilient 有弹性的；弹回的；有弹力的

Features of Spark RDD

- ✓ • In memory computation 内存计算
- Partitioning 划分；分割
- ✓ • Fault tolerance 容错功能
- Immutability 不可变性
- Persistence 坚持；持续存在。持续性
- Coarse-grained operations 粗粒度操作
- Location-stickiness 位置粘性
- ✓ Lazy Evaluation

Create RDDs

在你的驱动程序中并行一个现有的集合

- Parallelizing an existing collection in your driver program

通常, Spark会尝试根据您的集群自动设置分区数量

- Normally, Spark tries to set the number of partitions automatically based on your cluster
- Referencing a dataset in an external storage system 引用外部存储系统中的数据集
 - HDFS, HBase, or any data source offering a Hadoop InputFormat
 - By default, Spark creates one ^{分割, 分区} partition for each block of the file

RDD Operations

转变；变形

- Transformations

- functions that take an RDD as the input and produce one or many RDDs as the output
- Narrow Transformation
- Wide Transformation

- Actions

- RDD operations that produce non-RDD values.
- returns final result of RDD computations

计算

窄

宽

转变

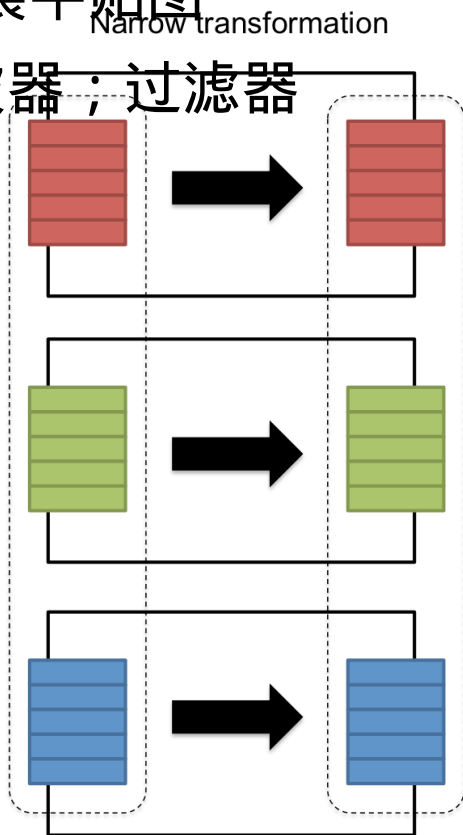
Narrow and Wide Transformations

转变就是从 一个 RDD 到 另外一个 RDD

Narrow transformation
involves no data shuffling

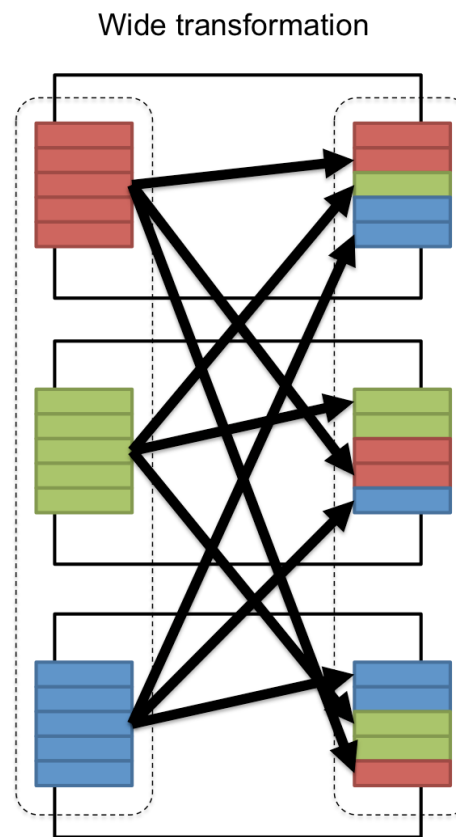
数据改组

- map
- flatMap 展平贴图
- filter 滤波器；过滤器
- sample



Wide transformation
involves data shuffling

- sortByKey
- reduceByKey
- groupByKey
- join



Action

- Actions are the operations which are applied on an RDD to instruct Apache Spark to apply computation and pass the result back to the driver
 - collect
 - take
 - reduce
 - foreach
 - sample
 - count
 - save

Lineage 宗族；谱系

所有祖先的图

- RDD lineage is the graph of all the ancestor RDDs of an RDD
 - Also called RDD operator graph or RDD dependency graph 依赖关系图
- Nodes: RDDs
- Edges: dependencies between RDDs

容错性

Fault tolerance of RDD

从容错数据生成的所有RD都是容错的

- All the RDDs generated from fault tolerant data are fault tolerant
- If a worker fails, and any ^{分区}partition of an RDD is lost
 - 可以使用沿袭从原始的容错数据集中重新计算分区
 - the partition can be re-computed from the original fault-tolerant dataset using the lineage
 - task will be assigned to another worker

有向无环图

DAG in Spark

- DAG is a direct graph with no cycle
 - Node: RDDs, results
 - Edge: Operations to be applied on RDD
- On the calling of Action, the created DAG submits to DAG Scheduler which further splits the graph into the stages of the task
- DAG operations can do better ^{更好的全局优化} global optimization than other systems like MapReduce

DAG, Stages and Tasks

- DAG Scheduler splits the graph into multiple stages
- Stages are created based on transformations ^{转变}
 - The narrow transformations will be grouped together into a single stage
 - Wide transformation define the boundary of 2 stages
- DAG scheduler will then submit the stages into the task scheduler
 - Number of tasks depends on the number of partitions
 - The stages that are not interdependent may be submitted to the cluster for execution in parallel
集群并行执行

宗系 ; 家系

Lineage vs. DAG in Spark

- They are both DAG (data structure)
- Different end nodes
- Different roles in Spark