Outline

Module 1：大數據簡介

Module 2：Hadoop Ecosystem介紹

Module 3：Hadoop 平台安裝

Module 4：Hadoop 分散式檔案系統（HDFS）

Module 5：Hadoop MapReduce

Module 6：Apache Hive

Module 7：Sqoop與Flume

Module 8：Apache Spark

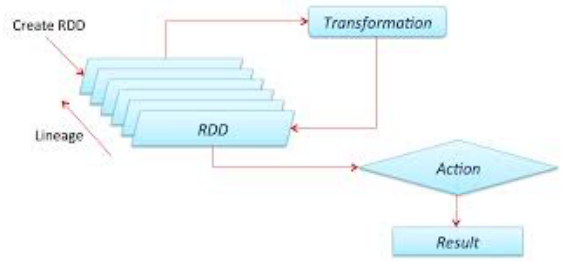
Module 9：Spark 平台安裝

Module 10：RDD – Resilient distributed dataset

Module 11：Scala 程式開發基礎

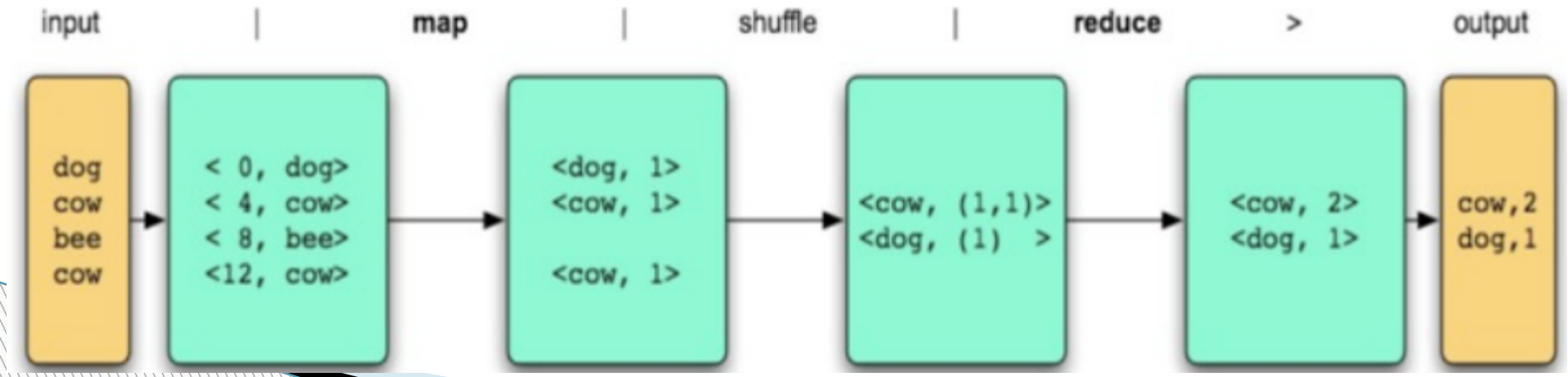
Module 12：Spark SQL 及 DataFrame

Module 13：Spark 機器學習函式庫(MLlib)



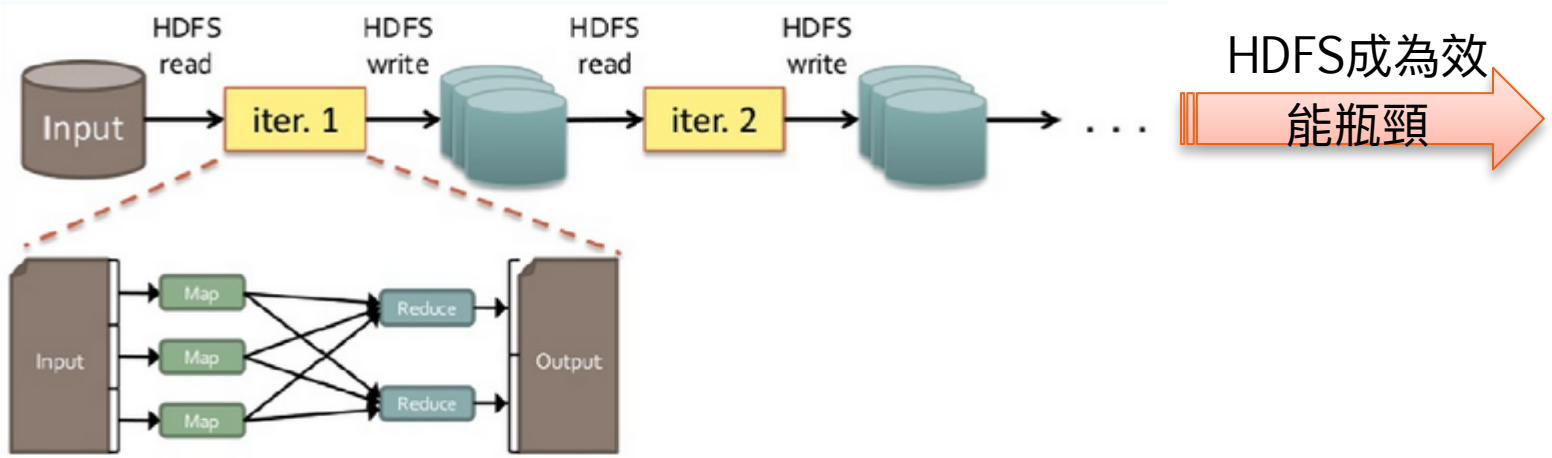
RDD之前，先談MapReduce

- 由Google提出，在電腦叢集上執行分散式運算的軟體架構。
- 開發人員只需專注於定義Map及Reduce的執行內容。
- MapReduce函式示意
  - Map　　(K1, V1) → list(K2, V2)
  - Reduce　(K2, list(V2))→list(K3, V3)
- 運作流程範例(以Word Count為例)



Hadoop之短 …

- MapReduce on Hadoop成功在分散式環境處理大量資料，但人們要的不只是Word Count …
  - 在需要多個iteration、且iteration間需共享資料(如機器學習)的處理情境下效能表現較不理想 …



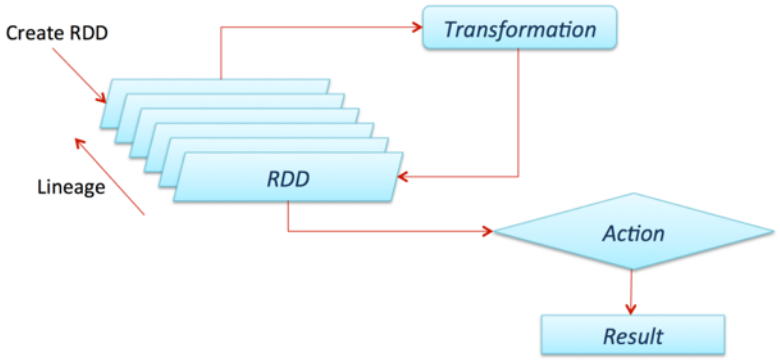
Spark之長 …

- Spark的解決方法 – RDD(Resilient Distribute Datasets)
  - In-Memory Data Processing and Sharing
  - 高容錯(tolerant)、高效能(efficient)的結構
- 容錯
  - 血統關係(lineage) – 描述RDD間之繼承關係
  - 透過lineage的記錄回復運算狀態
- 運算
  - Transformations: In memory、lazy，建立lineage及新的RDD
  - Action: 執行一個運算並return結果或是存到Storage裡
  - Persistence: 將RDD「持久化」在記憶體中做為後續使用，以加快執行效能

比喻: 1+2+3+4+5 = 15

Transformation

Action



常見的RDD運算

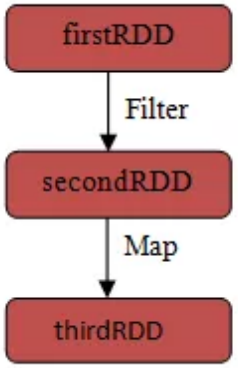
Transformations	*map*(*f* : T ⇒ U)	:	RDD[T] ⇒ RDD[U]
	filter(*f* : T ⇒ Bool)	:	RDD[T] ⇒ RDD[T]
	flatMap(*f* : T ⇒ Seq[U])	:	RDD[T] ⇒ RDD[U]
	sample(*fraction* : Float)	:	RDD[T] ⇒ RDD[T] (Deterministic sampling)
	groupByKey()	:	RDD[(K, V)] ⇒ RDD[(K, Seq[V])]
	reduceByKey(*f* : (V, V) ⇒ V)	:	RDD[(K, V)] ⇒ RDD[(K, V)]
	union()	:	(RDD[T], RDD[T]) ⇒ RDD[T]
	join()	:	(RDD[(K, V)], RDD[(K, W)]) ⇒ RDD[(K, (V, W))]
	cogroup()	:	(RDD[(K, V)], RDD[(K, W)]) ⇒ RDD[(K, (Seq[V], Seq[W]))]
	crossProduct()	:	(RDD[T], RDD[U]) ⇒ RDD[(T, U)]
	mapValues(*f* : V ⇒ W)	:	RDD[(K, V)] ⇒ RDD[(K, W)] (Preserves partitioning)
	sort(*c* : Comparator[K])	:	RDD[(K, V)] ⇒ RDD[(K, V)]
	partitionBy(*p* : Partitioner[K])	:	RDD[(K, V)] ⇒ RDD[(K, V)]
Actions	*count*()	:	RDD[T] ⇒ Long
	collect()	:	RDD[T] ⇒ Seq[T]
	reduce(*f* : (T, T) ⇒ T)	:	RDD[T] ⇒ T
	lookup(*k* : K)	:	RDD[(K, V)] ⇒ Seq[V] (On hash/range partitioned RDDs)
	save(*path* : String)	:	Outputs RDD to a storage system, *e.g.*, HDFS

Table 2: Transformations and actions available on RDDs in Spark. Seq[T] denotes a sequence of elements of type T.

RDD Ref: http://spark.apache.org/docs/latest/programming-guide.html#transformations

開發常用的RDD操作指令

- SparkContext.textFile – 讀取檔案建立RDD
- map: 由現有RDD內容建立新的RDD
- filter: 取出現有RDD中符合特定條件的資料，建立現有RDD子集
- reduceByKey: 針對RDD中相同Key的所有資料進行運算後建立新的RDD，常用於計算相同Key的資料總和、個數、或取得最大最小值。
- groupByKey: 匯集RDD中相同Key的所有資料，建立新的RDD
- join、cogroup: 整合兩個RDD中相同Key的所有資料，建立新的RDD
- sortBy、reverse: 依據RDD內容進行排序
- take(N): 取出RDD中前N筆資料建立新的RDD
- saveAsTextFile: 將RDD內容輸出至檔案



除錯常用的指令

- count: 計算RDD中資料筆數
- collect: 將RDD中所有資料轉成Collection(Seq物件)
- head(N): 顯示RDD中前N筆資料內容
- mkString: 將Collection內容用指定符號串成字串

[Tips]

- 除錯常用指令對效能影響甚鉅，非必要不要使用
- 儘量使用Transformation指令完成作業

RDD操作指令演練

- [Exercise]在spark-shell中輸入以下指令，觀察輸出結果
  - val intRDD = sc.parallelize(List(1,2,3,4,5,6,7,8,9,0))
  - intRDD.map(x => x + 1).collect()
  - intRDD.filter(x => x > 5).collect()
  - intRDD.stats
  - val mapRDD=intRDD.map{x=>("g"+(x%3), x)}
  - mapRDD.groupByKey.foreach{x=>println("key: %s, vals=%s".format(x._1, x._2.mkString(",")))}
  - mapRDD.reduceByKey(_+_).foreach(println)
  - mapRDD.reduceByKey{case(a,b) => a+b}.foreach(println)

RDD實戰演練(Word Count一下)

- [Exercise]讀取蓋茲堡宣言(The Gettysburg Address)，計算文中每個字的出現次數
  - 下載蓋茲堡宣言(The Gettysburg Address)(https://docs.google.com/file/d/0B5ioqs2Bs0AnZ1Z1TWJET2NuQlU/view)，另存為gettysburg.txt
  - 讀取gettysburg.txt，以空格( )切開字組 提示：sc.textFile、flatMap、split
    - 不分大小寫，不計入空白 提示：toLowerCase, filter
  - 計算每個字組在文中出現的次數 提示：reduceByKey
  - 依字組出現次數由大到小排序 提示：sortBy、foreach
  - 印出前五個 提示：take(5)、foreach

參考解答：https://github.com/yclee0418/sparkTeach/blob/master/codeSample/exercise/WordCount_Rdd.txt