

第7章  
Hadoop和SparkRDD 的操作



### Hadoop操作密技

* 完整指令集請參考：

<https://hadoop.apache.org/docs/r2.6.4/hadoop-project-dist/hadoop-hdfs/HDFSCommands.html>

### 管理者的指令

|  |  |
| --- | --- |
| 指令 | 說明 |
| hdfs namenode -format | 格式化NameNode |
| hdfs dfsadmin –report | 回報基本的檔案系統資訊和統計 |
| hdfs dfsadmin -getDatanodeInfo slave1:50020 | 查看slave1的資訊 |

### 使用者的指令

|  |  |
| --- | --- |
| 指令 | 說明 |
| hdfs dfs -ls /user/mis | 列出/user/mis目錄下的目錄和檔案 |
| hdfs dfs -ls -R /user/mis | 加上-R選項，列出所有子目錄細節 |
| hdfs dfs -mkdir /hadoop/data | 建立/hadoop/data這樣目錄結構 |
| hdfs dfs -mkdir -p /hadoop/data | 加上 -p選項，連同原本不存在的上層父目錄一同建立 |
| hdfs dfs -touchz /hadoop/data/input.txt | 在/user/hadoop/data之下建立一個空檔案input.txt |
| hdfs dfs -appendToFile /etc/hosts /hadoop/data/input.txt | 使用-appendToFile將本地的檔案附加到某個檔案之後 |
| hdfs dfs -copyFromLocal /home/mis/.bashrc /hadoop/mis.sh | 把使用者mis的登入腳本檔拷貝到hdfs上的/hadoop目錄下，並且儲存為mis.sh |
| hdfs dfs -copyFromLocal /usr/local/hadoop/etc/hadoop/\*.xml /hadoop/data | 將本地Hadoop設定檔目錄下的所有 .xml檔案拷貝到/hadoop/data目錄下 |
| hdfs dfs -copyToLocal /hadoop/mis.sh /home/mis | 將hdfs上的/hadoop/mis.sh拷貝放置本地/home/mis目錄下 |
| hdfs dfs -getmerge /hadoop/data all.xml | 將hdfs上的目錄下所有檔案合併至一個本地檔案 |
| hdfs dfs -cat /hadoop/mis.sh | 顯示前例中所產生的mis.sh檔案內容 |
| hdfs dfs -tail /hadoop/mis.sh | 顯示檔案最後1KB的資料 |
| hdfs dfs -cp /hadoop/mis.sh /hadoop/data | 將/hadoop/mis.sh檔案拷貝到/hadoop/data之下 |
| hdfs dfs -cp /hadoop/data / | 將/hadoop/data整個目錄，拷貝到hdfs的跟目錄下 |
| hdfs dfs -cp /hadoop/mis.sh file:///home/mis/mis2.txt | 從hdfs拷貝/hadoop/mis.sh至本地的/home/mis目錄，檔名取為mis2.txt |
| hdfs dfs -mv /data /hadoop/data2 | 將/data移動到/hadoop下，並將目錄名稱改為data2 |
| hdfs dfs -rm /hadoop/data/input.txt  hdfs dfs -rm -r /hadoop/data2 | 分別刪除檔案/hadoop/data/input.txt和目錄/hadoop/data2 |
| hdfs dfs -setrep 3 /hadoop/data/mis.sh | 將/hadoop/data/mis.sh的存檔份數從1份提升到3份 |
| hdfs dfs -du /hadoop | 查看/hadoop目錄下所佔總空間大小（bytes） |
| hdfs dfs -du /hadoop/data | 查看/hadoop/data目錄下所佔總空間大小（bytes） |
| hdfs dfs -count /hadoop | 查看/hadoop目錄下的目錄數、檔案數和所佔總空間大小（bytes） |
| hdfs fsck /hadoop | 檢查/hadoop下檔案架構的健康狀態 |

### SparkRDD密技

* 完整RDD操作請參考：

1. https://spark.apache.org/docs/1.6.2/api/scala/#org.apache.spark.rdd.RDD
2. https://spark.apache.org/docs/1.6.2/api/scala/#org.apache.spark.rdd.PairRDDFunctions

#### 轉換類

|  |  |
| --- | --- |
| 指令 | 說明 |
| 啟動spark shell | |
| spark-shell | 啟動spark shell |
| RDD轉換運算—建立 | |
| val numRDD = sc.makeRDD(1 to 9)  numRDD.foreach(println)  numRDD.partitions.size | 建立RDD，並查看RDD內容和分割數 |
| val numRDD = sc.parallelize(1 to 9)  numRDD.foreach(println) | 使用parallelize建立RDD，功能與makeRDD相同 |
| val numRDD2 = sc.makeRDD(1 to 9, 3)  numRDD2.partitions.size | 建立RDD，設定分割數為3 |
| val strRDD = sc.makeRDD(Array("一", "二", "三", "四", "五", "六", "日"))  strRDD.foreach(println)  strRDD.collect() | 建立字串RDD，使用二種不同的輸出方式。注意collect會將資料轉成Array |
| val numRDD = sc.makeRDD(Array(1,2,3,4,5,6,7,8,9))  numRDD.foreach(println)  numRDD.collect() | Array建立數字1～9 RDD，也可以 |
| 讀入檔案 | |
| val fileRDD = sc.textFile("file:///etc/network/interfaces")  fileRDD.foreach(println) | 讀入本地檔案/etc/network/interfaces建立RDD，並用foreach(println)列印出來 |
| RDD轉換運算—基礎 | |
| map運算 | |
| val numRDD = sc.makeRDD(1 to 9)  numRDD.foreach(println)  val n2RDD = numRDD.map(x => x\*2)  n2RDD.foreach(println) | 找出九九乘法表中乘數2的列表 |
| val n2RDD = numRDD.map(\_\*2)  n2RDD.foreach(println) | 使用佔位字元（placeholder）的寫法 |
| val f2RDD = numRDD.map(\_.toFloat)  f2RDD.foreach(println) | 整數元素轉變成浮點數 |
| val equRDD = f2RDD.map(x => 3\*math.sqrt(x) + 1)  equRDD.foreach(println) | 產生y=3\*√x+1的結果 |
| val strRDD = sc.makeRDD(Array("一", "二", "三", "四", "五", "六", "日"))  val weekRDD = strRDD.map(x => "星期" + x)  weekRDD.foreach(println) | 將每一個中文數字前面「黏」上 “星期” 二字 |
| flatMap運算 | |
| val xRDD = sc.makeRDD(Array("Hello how are you"))  xRDD.foreach(println)  val yRDD = xRDD.map(x => x.split(" "))  yRDD.collect  val zRDD = xRDD.flatMap(x => x.split(" "))  zRDD.foreach(println) | 將一段文字 “Hello how are you” 放入xRDD中，用字串方法split切開空白，形成獨立的單字。首先用map來執行，其結果會被包含在陣列中（yRDD），改用flatMap來執行便可以得到獨立的單字（zRDD） |
| filter運算 | |
| val numRDD = sc.makeRDD(1 to 9)  numRDD.foreach(println)  val xRDD = numRDD.filter(x => x>4)  xRDD.foreach(println) | 留下大於4的元素 |
| val yRDD = numRDD.filter(x => x % 2 == 0)  yRDD.foreach(println) | 挑出偶數元素 |
| val nameRDD=sc.makeRDD(Array("John", "Mary", "Jack", "Nancy", "Mark", "Maggie", "Melissa"))  nameRDD.foreach(println)  nameRDD.filter(x => x=="John").foreach(println)  nameRDD.filter(x => x.contains("J")).foreach(println)  nameRDD.filter(x => x.contains("J") || x.contains("k")).foreach(println) | 找出RDD中是否有John、有無名字包含大寫J、有無名字包含大寫J或小寫k |
| groupBy運算 | |
| val numRDD = sc.makeRDD(1 to 9)  numRDD.collect()  val xRDD=numRDD.groupBy(x => x % 2 == 0) //基數分一組，偶數分一組  xRDD.foreach(println) //印出來，有二組資料  xRDD.collect() //轉成陣列看看  xRDD.collect()(0) //查看基數那一組  xRDD.collect()(1) //查看偶數那一組 | 將數組分偶數一組，基數一組 |
| val xRDD=numRDD.groupBy(x => { if (x % 2 == 0) "even" else "odd"}) //基偶分組另一個寫法 | 基偶分組另一個寫法 |
| val yRDD=numRDD.groupBy(x => x > 5) //大於5的分一組，小於等於5的分一組  yRDD.foreach(println) //列印出來看看 | 大於5的分一組，小於等於5的分一組 |
| val nameRDD=sc.makeRDD(Array("John", "Mary", "Jack", "Nancy", "Mark", "Maggie", "Melissa"))  nameRDD.foreach(println)  val aRDD=nameRDD.groupBy(x => x.substring(0,1))  aRDD.foreach(println)  val bRDD=nameRDD.groupBy(x => x.contains("a"))  bRDD.foreach(println) | 以姓名的開頭來做分組  以姓名中是否包含小寫a來做分組 |
| sortBy運算 | |
| val xRDD=sc.makeRDD(Array(2,4,6,8,1,3,5,7,9))  xRDD.collect  val yRDD=xRDD.sortBy(x => x)  yRDD.collect  val zRDD=xRDD.sortBy(x => x, false)  zRDD.collect | 以升冪（由小到大）  降冪（由大到小）排序 |
| val nameRDD=sc.makeRDD(Array("Maggie", "John", "Mary", "Jack", "Melissa", "Nancy", "Mark"))  nameRDD.collect  val aRDD=nameRDD.sortBy(x => x)  aRDD.collect  val bRDD=aRDD.sortBy(x => x.length)  bRDD.collect | 對字串做升冪進行排序，以及根據字串的長度由短到長做排序 |
| RDD轉換運算—統計 | |
| sample運算 | |
| val xRDD = sc.makeRDD(1 to 100)  xRDD.collect  val yRDD = xRDD.sample(false, 0.1)  yRDD.collect | 對xRDD隨機取十分之一的元素 |
| randomSplit運算 | |
| val xRDD = sc.makeRDD(1 to 100)  xRDD.collect  val yRDD = xRDD.randomSplit(Array(0.7, 0.3))  yRDD(0).count  yRDD(0).collect  yRDD(1).count  yRDD(1).collect | 將xRDD以七三比的比率切割成兩個字集合，並計算出它們的筆數 |
| RDD轉換運算—集合 | |
| union運算 | |
| val xRDD=sc.makeRDD(1 to 5)  val yRDD=sc.makeRDD(4 to 8)  xRDD.collect()  yRDD.collect()  val zRDD=xRDD.union(yRDD)  zRDD.collect() //注意：4和5有重複，union結果不會去除重複  val wRDD=xRDD ++ yRDD  wRDD.collect() | 將xRDD和yRDD進行聯集運算 |
| intersection運算 | |
| val xRDD=sc.makeRDD(1 to 5)  val yRDD=sc.makeRDD(4 to 8)  xRDD.collect()  yRDD.collect()  val zRDD=xRDD.intersection(yRDD)  zRDD.collect() //只有4和5是xRDD和yRDD所共有 | 求出交集 |
| subtract運算 | |
| val xRDD=sc.makeRDD(1 to 5)  val yRDD=sc.makeRDD(4 to 8)  xRDD.collect()  yRDD.collect()  val zRDD=xRDD.subtract(yRDD)  zRDD.collect() //xRDD的4和5被減掉了 | xRDD與yRDD所共有的元素是4和5，xRDD.subtract(yRDD)的指令便是從xRDD中將4和5去除，並保留下剩餘的部份 |
| distinct運算 | |
| val xRDD=sc.makeRDD(1 to 5)  val yRDD=sc.makeRDD(4 to 8)  val zRDD=xRDD.union(yRDD).distinct  zRDD.collect() | 去除重複項 |
| cartesian運算 | |
| val xRDD=sc.makeRDD(Array("老師","學生"))  val yRDD=sc.makeRDD(Array("起床","吃早餐","去學校"))  val zRDD=xRDD.cartesian(yRDD)  zRDD.foreach(println) | 2個元素的RDD和一個有3個元素的RDD交乘之後產生2\*3＝6個元素 |
| RDD轉換運算—組合 | |
| zip運算 | |
| val xRDD=sc.makeRDD(Array("老師","學生"))  val yRDD=sc.makeRDD(Array("教書","讀書"))  xRDD.foreach(println)  yRDD.foreach(println)  val zRDD=xRDD.zip(yRDD)  zRDD.foreach(println) | 將xRDD和yRDD一一對應並組合成鍵值對 |
| zipWithIndex運算 | |
| val xRDD=sc.makeRDD(Array("老師","學生","家長","教育部長"))  xRDD.foreach(println)  val yRDD=xRDD.zipWithIndex  yRDD.foreach(println) | 將RDD的元素與其元素的序號索引組合起來變成鍵值對 |
| RDD轉換運算—鍵值對 | |
| keys、values運算 | |
| val studRDD=sc.makeRDD(Array(("ID01","曾子"),("ID02","子路"),("ID03","顏回"),("ID04","宰予")))  val xRDD=studRDD.keys  val yRDD=studRDD.values  xRDD.collect  yRDD.collect | 將鍵值對RDD的鍵或值的部份取出 |
| groupByKey運算 | |
| val pairRDD=sc.makeRDD(Array(("老師","孔子"), ("學生","曾子"), ("學生","子路"), ("學生","海倫凱勒"), ("老師","蘇利文")))  val groupRDD=pairRDD.groupByKey() //依照鍵值來分組，學生一組，老師一組  groupRDD.foreach(println) | 依照鍵值來分組，學生一組，老師一組 |
| val kvARR=Array((1,1),(1,2),(2,2),(2,3),(2,4))  val xRDD=sc.makeRDD(kvARR)  xRDD.foreach(println)  val yRDD=xRDD.groupByKey() //依照鍵值來分組  yRDD.foreach(println) | 補充：數字分組範例 |
| reduceByKey運算 | |
| val moneyRDD=sc.makeRDD(Array(("收入",1000),("支出",200),("收入",2000),("支出",100),("收入",3000),("支出",300)))  moneyRDD.foreach(println)  val sum1RDD=moneyRDD.reduceByKey((value1, value2) => value1 + value2) //依照鍵值來分別加總  sum1RDD.foreach(println)  val sum2RDD=moneyRDD.reduceByKey(\_+\_)  sum2RDD.foreach(println) | 將「收入」和「支出」分別加總 |
| def sumFunc(x:Int, y:Int):Int={  x+y  }  val sumRDD=moneyRDD.reduceByKey(sumFunc)  sumRDD.foreach(println) | 補充：建立精簡函數的方式 |
| sortByKey運算 | |
| val studRDD=sc.makeRDD(Array(("ID02","子路"),("ID04","宰予"),("ID01","曾子"),("ID03","顏回")))  studRDD.foreach(println)  studRDD.sortByKey(true).foreach(println) //按照key的升冪(由小排到大)排序  studRDD.sortByKey(false).foreach(println) //按照key的降冪排序 | 根據鍵由小排到大 |
| subtractByKey運算 | |
| val studRDD=sc.makeRDD(Array(("ID01","曾子"),("ID02","子路"),("ID03","顏回"),("ID04","宰予")))  val dropoutRDD=sc.makeRDD(Array(("ID02","子路")))  studRDD.subtractByKey(dropoutRDD).collect | 取差集將學生 ("ID02","子路") 去除 |
| join運算 | |
| val studRDD=sc.makeRDD(Array(("ID01","曾子"),("ID02","子路"),("ID03","顏回"),("ID04","宰予")))  val engRDD=sc.makeRDD(Array(("ID01",100),("ID02",85),("ID03",91)))  studRDD.join(engRDD).foreach(println) | 讓相同「學號」的「姓名」和「英文成績」連結在一起 |
| studRDD.leftOuterJoin(engRDD).foreach(println) | 缺考的同學也會出現 |
| studRDD.leftOuterJoin(engRDD).map { case (a, (b, c: Option[Int])) => (a, (b, (c.getOrElse()))) }.foreach(println) | 補充：去除Option類別（some()或None） |
| filter的鍵值對運算 | |
| val studRDD=sc.makeRDD(Array(("ID01","曾子"),("ID02","子路"),("ID03","顏回"),("ID04","宰予"),("ID10","子貢")))  studRDD.filter(x => x.\_1 == "ID03" ).collect  studRDD.filter(x => x.\_2 == "子路" ).collect  studRDD.filter { case (k,v) => k == "ID03" } .collect  studRDD.filter { case (k,v) => v == "子路" } .collect | 找出學號（鍵）為ID03的學生，還有名字（值）為 "子路" 的學生 |
| studRDD.filter { case (k,v) => v.contains("子") } .collect | 過濾出值包含 "子" 的學生 |
| studRDD.filter { case (k,v) => k.contains("1") && v.contains("子")} .collect | 過濾出鍵包含 "1"，且值包含 "子" 的學生 |

#### 行動類

|  |  |
| --- | --- |
| RDD行動運算—基礎 | |
| reduce運算 | |
| val numRDD = sc.makeRDD(1 to 10)  numRDD.collect  numRDD.reduce(\_+\_) | 將numRDD中1～10的元素做精簡（加法）產生一個「立即」的整數結果 |
| val numRDD = sc.makeRDD(Array("A","B","C","D","E"))  numRDD.collect  numRDD.reduce(\_+\_)  numRDD.reduce((x,y)=>y+x)  numRDD.reduce((x,y)=>y.concat(x)) | 示範將字串「黏」起來 |
| collect、first、top、take、takeOrdered運算 | |
| val numRDD = sc.makeRDD(Array(2,4,6,8,1,3,5,7,9))  numRDD.collect // 將所有元素放入陣列  numRDD.first // 取得numRDD的第一筆元素  numRDD.top(3) // 將numRDD的前三大（降冪排序）元素放入陣列  numRDD.take(3) // 取得前三筆元素放入陣列  numRDD.takeOrdered(3) // 按照升冪順序取得前三筆元素放入陣列 | （參考指令後的註解） |
| collectAsMap運算 | |
| val studRDD=sc.makeRDD(Array(("ID01","曾子"),("ID02","子路"),("ID03","顏回"),("ID04","宰予")))  val studMap=studRDD.collectAsMap  studMap("ID01") | 將RDD的元素轉成映射資料結構後，可以提供以鍵查找值的功能。 |
| foreach運算 | |
| val numRDD = sc.makeRDD(1 to 9)  numRDD.collect  numRDD.foreach(x=>println("2 \* "+ x + " = " + 2\*x)) |  |
| RDD行動運算—統計 | |
| 統計運算 | |
| val numRDD = sc.makeRDD(1 to 9)  numRDD.count  numRDD.mean  numRDD.stdev  numRDD.max  numRDD.min  numRDD.sum  numRDD.variance  numRDD.stats | 計算統計量 |
| RDD行動運算—鍵值對 | |
| countByKey、countByValue運算 | |
| val xRDD=sc.makeRDD(Array(("A",1),("A",1),("A",1),("B",1),("B",2),("B",2)))  xRDD.countByKey  xRDD.countByValue | countByKey指令計算相同的鍵的出現數量，而countByValue則計算相同鍵值對（也就是鍵和值都相同）的出現數量 |
| val yRDD=sc.makeRDD(Array("A","A","B","C","C","C"))  yRDD.countByValue | 非鍵值對也可以使用countByValue |
| lookup運算 | |
| val xRDD=sc.makeRDD(Array(("A",1),("A",2),("A",3),("B",4),("B",5),("C",6)))  xRDD.lookup("A") | 查詢xRDD中鍵為 "A" 的值有哪些 |
| RDD行動運算—檔案輸出 | |
| saveAsTextFile運算 | |
| val studRDD=sc.makeRDD(Array(("ID01","曾子"),("ID02","子路"),("ID03","顏回"),("ID04","宰予")))  studRDD.saveAsTextFile("hdfs://master:9000/stud\_output") | 將studRDD的內容存成文字檔，saveAsTextFile的參數file:///home/mis/stud\_output必須是一個「不存在」的目錄名稱 |
| :q | 用:q離開spark-shell環境 |
| hdfs dfs -ls /stud\_output  hdfs dfs -cat /stud\_output/part-00000 | 用:q離開spark-shell環境，或開啟另外一個「終端機」程式，到Ubuntu提示字元下，查詢輸出的結果 |