Apache Spark

108703028,108703013,108703007,108703005
108703022,110753201,110971005

問題背景

Apache Spark 的核心概念

Apache Spark 透過記憶體存取中間運算結果，加強了資料處理的即時性，也達成高容錯性和高伸縮性。Spark 還基於 DAG（有向無環圖）分配任務，構建平行計算框架。此外，採用 RDD（Resilient Distributed Dataset）擔任最小運算單位，得以在讀取等操作時，精確到每筆紀錄。

Spark 可以支援 Scala、Java 和 Python 的程式撰寫，還能執行在任何 Hadoop 資料來源上，讓使用者可以輕易轉移已有的資料，塑造出極高的易用及普遍性。

Spark 也在核心基礎上，開發出各類型的 API。像是能夠進行 Stream processing（流處理程式）的 Spark Streaming；提供 Graph Computation（圖譜計算）服務的 GraphX；支援 Machine Learning（機器學習）的 ML Base 和 MLlib 庫；協助結構化資料 SQL 查詢和分析的 Spark SQL 和 Shark。

Apache Spark 是個完整的大數據分析生態系統，其便利性和低延遲性使它成為眾多資料科學家、工程師愛用的大數據處理平台。

Apache Spark 的應用場景

在 Apache Spark 中，使用者只需建立一個系統，就能完整地進行所有資料分析可能遇到的步驟，不用再像過去一樣，使用不同的軟體並花時間及心力做資料格式的轉換。因此，許多知名企業都採用 Spark 來架構內部的資料分析系統。

像是 Uber 使用 Spark 構建了 Uber Spark Compute Service (uSCS)，幫助進行 Uber 乘客和 Uber 駕駛者的定價運算、需求預測以及餐廳推薦等商業任務，還有協助系統的 ETL 操作和數據探索，讓 uSCS 這個管理過億活躍用戶的大數據平台能更加穩定。

還有，騰訊旗下的「廣點通」也是使用 Spark 的應用之一。騰訊大數據使用了 Spark 平台來支援採擷分析類別計算、互動式即時查詢以及允許誤差範圍的快速查詢計算。目前有超過 200 台的 Spark 叢集，並且累積了大量的案例和營運經驗，已成功應用在廣點通 pCTR 投放系統上，支援了每日上百億的請求量。

此外，淘寶使用了 Spark 得以解決複雜的機器學習，像是得多次反覆運算或高計算複雜度的演算法，並將其運用在阿里搜索、廣告業務和淘寶的推薦相關演算法。此外，還使用 GraphX 解決了許多生產問題，像是採用「最大連通圖」實現社區發現；以「隨機遊走」為基礎分析使用者屬性傳播；使用「度分析」找出中樞節點等計算場景。

Yahoo 則是採用 Spark 叢集，取代原先的商業 BI、OLAP 工具。Spark 承擔報表功能的同時，還能達成更高品質的大數據 SQL 服務。Yahoo 也將此運用在一種讓廣告能尋找到其適合對象的應用：Audience Expansion 的演算法中，讓企業能夠更有效率的投放廣告。

Spark 不僅是在商業面上，還是系統效能上，都能讓企業或是使用者得到卓越的成效。而且在如今 app 風行的時代，更讓 Spark 能夠大展風采，讓大數據得以快速流通，並在高速且極小誤差下處理、分析。

架構解析

基本概念

- RDD（Resilient Distributed Dataset）彈性分散式資料集：

  是 Spark 中的核心概念。每一個 RDD 都是由一堆 RDD 資料元素所組成的資料結構，具有自動容錯、位置感知性排程和可伸縮性這幾個特性，因此容錯性跟並行性極高，原則上是一個只能讀取不能修改的集合，如果想要修改已知的 RDD 的資料，就必須要對這個 RDD 進行轉換產生另一個新的 RDD，而不是直接修改裡面的資料內容。

  一般來說，一堆的 RDD 元素可以被組裝成一到數個的 partition(分區)，這些 partition 可以分散在不同的機器上平行執行，而且大多數的情況是存在各個本機的記憶體中。RDD 會將資料資料存在電腦記憶體裡面，但是如果記憶體資源不足，Spark 會把 RDD 的數據寫入到硬碟中；此外，如果某個節點上的 RDD partition 發生故障，導致數據遺失，RDD 會自動藉由自己的數據來源重新計算 partition。
- RDD API：

  提供了一組豐富的操作以支援常見的資料運算，可以分為 Action 跟 Transformation 兩種類型。
- DAG（Directed Acyclic Graph）有向無環圖：

在 Spark 中負責邏輯計算，DAG 是一組頂點和邊的組合（頂點代表 RDD，邊代表對 RDD 的一系列操作），可以反映 RDD 之間的依賴關係

架構圖

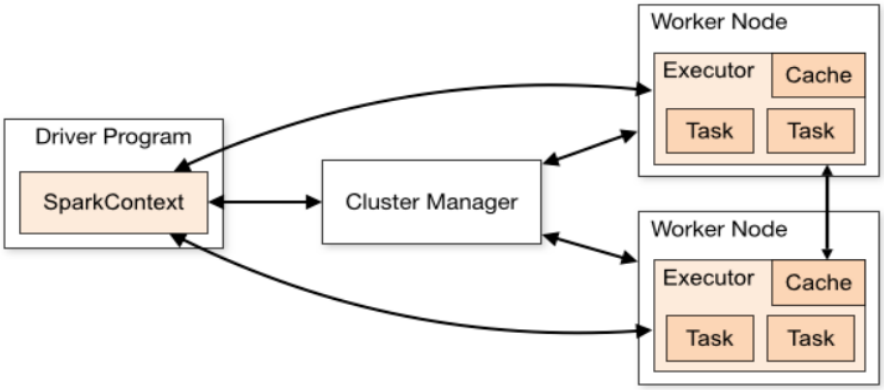
(1) 集群資源管理器(Cluster Manager)：對整個 Spark 應用程式進行資源的分配和管理調度，從 Cluster Manager 得到資源來執行 spark job，可以是自帶的 CM、Mesos 或 YARN。

(2) 工作節點(WorkerNode)：管理工作的單位，用來運行 Task。每個 Worker 可以管理多個 Executor，並且同時跟 Cluster Manager 溝通

(3) 任務控制節點(Driver)：建立 SparkContext 的 class，建立讓 Spark Application 可以運行的環境，此外還能跟 Cluster 以及 Worker 互相溝通，進行資源申請、任務分配和互相監控。例如：程式提交後產生有向無環圖 DAG、對 DAG 分成多個階段、對多個階段進行任務拆解、把拆解後的任務分配到相關的 Executor 執行。

(4) 執行行程(Executor)：是每個工作節點上負責具體任務執行的行程（Task），同時負責跟 Driver 溝通工作項目

(5) Partition：是一種工作單位，每個 Task 存放在一個 Partition 中。



架構分析

首先 Spark context 會編寫自己的 Spark Application，根據反射的方式，創建和構造一個 Driver，Driver 會執行我們寫的代碼，而 SparkContext 中有兩個主要的組件 DAGScheduler 和 TaskScheduler。

(1) TaskScheduler 組件：

從後台連接 Master，並註冊 Application，使用內建的算法在 Worker 上啟動多個 Executor，Executor 啓動之後會自己反向註冊到 TaskScheduler 上。當所有的 Executor 都反向註冊到 TaskScheduler 之後，sparkContext 結束初始化，往下執行我們編寫的 spark 代碼。

(2) DAGScheduler：

在執行 Spark 代碼時，每一個 Action 都會創建一個 job，並提交到 DAGScheduler，根據 stage 的劃分算法將一個 job 分成多個 stage，每個 stage 會有一個 taskSet，DAGScheduler 根據 task 算法將 taskSet 中的每個 task 送到 executor 上執行，每個 task 針對 RDD 的一個 partition，執行我們定義的函數，依次類推，直至所有的操作執行完爲止。

Spark 解決什麼問題？

在 Spark 出來之前，Hadoop 算是很普遍的分散式系統，做為一個分佈式群集，將巨大的數據集分散到一個由普通計算機組成的集群中的多個節點進行儲存，讓整個從即有很高的帶寬；此外，Hadoop 中的 MapReduce，將應用程式被分割成許多小部分，每個部分都能在叢集中的任意節點上執行或重新執行。

然而不斷的讀取跟寫入會佔用大量的 I/O，浪費大量資源，因此 Spark 提出的 RDD 具有數據流模型的特點，像是自動容錯、可伸縮性跟可擴展性，此外，在同時進行多個查詢時，可以將工作及緩存到記憶體中，後續再做查詢時可以直接使用先前查詢的結果，這樣跟 Hadoop 相比，速度快了很多。

技術實作與心得

實驗記錄

一、spark 安裝與配置：

1. 進入 Apache Spark 官網進行 Spark 的下載，下載完之後解壓，在解壓文件夾所在目錄下打開 terminal，將解壓後的文件夾移動到/usr/local/目錄下，並改名爲 spark。

2. 安裝 pyspark

在 Terminal 輸入：pip install pyspark

3. 配置環境變量

vim ~/.bash_profile 進行編輯，增加環境變量：

```
export SPARK_HOME=/Users/wangjingzi/opt/spark
export PATH=$PATH:$SPARK_HOME/bin
export PYSPARK_PYTHON=python3
```

然後保存退出，執行 source ~/.bash_profile，使之生效。

4. 打開 terminal，輸入 pyspark，會出現

```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.2.1
      /_/
Using Python version 3.9.2 (v3.9.2:1a79785e3e, Feb 19 2021 09:06:10)
Spark context Web UI available at http://192.168.0.102:4040
Spark context available as 'sc' (master = local[*], app id = local-1649003812362).
SparkSession available as 'spark'.
>>>
```

檢查是否有正確啟動

```
>>> print(sc.version)
3.2.1
```

二、體驗 spark 的執行：對於文件「shakespear.txt」來做字數統計

1. PySpark 將會自動使用本地 Spark 配置創建一個 SparkContext, textFile 方法將 shakespeare.txt 加載到一個 RDD 命名文本

```
>>> text = sc.textFile("shakespeare.txt")
>>> print(text)
shakespeare.txt MapPartitionsRDD[1] at textFile at NativeMethodAccessorImpl.java:0
```

2.首先用 tokenize 把文本拆分為 words, 返回拆分的 word list。然後通過給 flatMap 傳遞 tokenize 對 textRDD 進行變換創建了一個 wordsRDD。

```
>>> from operator import add
>>> def tokenize(text):
...     return text.split()
...
>>> words = text.flatMap(tokenize)
>>> print(words)
PythonRDD[2] at RDD at PythonRDD.scala:53
```

3.將每個 word map 到一個 key-value pair，其中 key 就是 word，value 是 1，然後使用 reducer 計算每個 key 的 1 總數， 可以用 toDebugString 方法來查看 PipelinedRDD 是如何被轉換的

```
>>> wc = words.map(lambda x: (x,1))
```

```
>>> print (wc.toDebugString())
b'(2) PythonRDD[3] at RDD at PythonRDD.scala:53
|  shakespeare.txt MapPartitionsRDD[1] at textFile at
NativeMethodAccessorImpl.java:0
|  shakespeare.txt HadoopRDD[0] at textFile at NativeMethodAccessorImpl.java:0
```

4.使用 reduceByKey 進行字數統計，然後把統計結果寫到 disk，最終調用 saveAsTextFile，這個分布式任務就開始執行

```
>>> counts = wc.reduceByKey(add)
>>> counts.saveAsTextFile("wc")
```

```
(base) wangjingzi@MacBook-Pro:~/Desktop/110-2/分散式系統/project$ ls
wc  shakespeare.txt
(base) wangjingzi@MacBook-Pro:~/Desktop/110-2/分散式系統/project$ cd wc
(base) wangjingzi@MacBook-Pro:~/Desktop/110-2/分散式系統/project/wc$ ls
```

5. exit()退出後，會發現該目錄下多了一個 wc directory

每個 part 文件都代表進程計算得到的被保持到 disk 上的最終 RDD。對某一個 part 文件進

```
(base) wangjingzi@MacBook-Pro:~/Desktop/110-2/分散式系統/project/wc$ head part-00000
('Shakespeare', 1)
('fairest', 5)
('creatures', 2)
('we', 14)
('increase,', 4)
('That', 83)
('thereby', 1)
("beauty's", 17)
('rose', 3)
('never', 13)
```

行 head 命令，可以能看到字數統計元組。

三、以 standalone 單機模式實做簡單範例：

使用本機作為 master，搜索該程式同目錄下的 ontime 文件夾的 2 個 CSV file，將航班的延誤時間可視化出來，以下僅展示部分 code

```
import csv
import matplotlib.pyplot as plt
```

```
import io
from datetime import datetime
from collections import namedtuple
from operator import add, itemgetter
from pyspark import SparkConf, SparkContext
```

首先通過 setMaster 將 Spark 配置到 SparkConf，再創建了 SparkContext，使用了配置好的 context 執行 main

```
if __name__ == "__main__":
  # Configure Spark
  conf = SparkConf().setMaster("local[*]")
  conf = conf.setAppName(APP_NAME)
  sc   = SparkContext(conf=conf)
  main(sc)
```

Load 航空公司的 csv 到 RDD，將其 split 並返回代表每行的 tuple，再將 collect 傳給 RDD，使每一行以 list 的形式從 RDD 傳回 driver，並儲存為 dictionary，然後使用 sc.broadcast 廣播給 cluster 的每個 node。接著相同的對航班的 csv 做 split 並且將數據 parse 為方便處理的格式。有了 Flight 的 RDD，再通過 map 將 RDD 轉換為 key-value pair，其中 key 是航空公司的名字，value 是到達和出發的延誤時間總和。使用 reduceByKey 和 add 可以得到每個航空公司的延誤時間總和，然後 RDD 被傳遞給 driver，按照升序排列，print 出來再使用 matplotlib 進行了可視化。

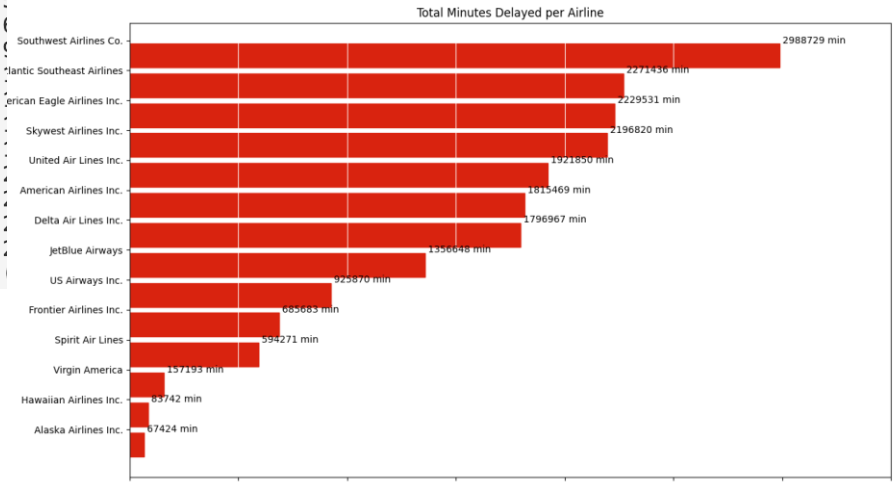
```
def main(sc):

  airlines =dict(sc.textFile(  “ontime/airlines.csv”  ).map(split).collect()
  airline_lookup = sc.broadcast(airlines)
  flights = sc.textFile("ontime/flights.csv").map(split).map(parse)
  delays  = flights.map(lambda f: (airline_lookup.value[f.AIRLINE],
                    add(f.DEPARTURE_DELAY,f.ARRIVAL_DELAY)))
  delays  = delays.reduceByKey(add).collect()
  delays  = sorted(delays, key=itemgetter(1))
  for d in delays:
    print("%0.0f minutes delayed\t%s" % (d[1], d[0]))
```

```
plot(delays)
```

運行結果如下：

```
With splitting
67424 minutes delayed   Alaska Airlines Inc.
83742 minutes delayed   Hawaiian Airlines Inc.
```



實驗心得

通過這次實作，我對 spark 有了基本的瞭解。Spark 擁有多種語言的函數式編程 API，提供了除 map 和 reduce 之外更多的運算符，這些操作是通過一個稱作彈性分布式數據集(resilient distributed datasets, RDDs)的分布式數據框架進行的，RDD 是 Spark 中最基本的數據抽象，它代表一個不可變、可分區、裏面的元素可並行計算的集合。其彈性具體可以體現在自動的進行內存和磁盤數據存儲的切換 ；基於 Lineage 的容錯，第 n 個節點出錯，會從第 n-1 個節點恢復 ；數據 partition 的高度彈性。

除此之外，本次實作我還體會到了 spark 的 lazy evaluation 的特性，這是指 spark 直到 action 動作之前，數據不會先被計算，這裡的 action 指的是 collect，count，reduce 等需要拉回產生結果的算子。spark 的 transform 處理的數據，都不會立刻執行，它會根據 rdd 之間的鏈式進行傳遞，這裡的指的就是 transformmap，union，flatmap，groupByKey, join 等。例如在第二部分實作中的 counts = wc.reduceByKey(add)，此時 spark UI 上不會有任何 job 被提交，因爲這個計算沒有任何 action 算子，實際上根本沒有被計算，這就是 lazy 特性。只有當 counts.saveAsTextFile(「wc」)，這個 job 才會開始執行，將數據輸出存儲到 wc 的目錄下。而這樣的特性可以將代碼的程序變成一塊塊的操作鏈，能夠極大的減少中間計算過程提高計算效率，只有真正需要用到的數據才會計算。

評論

優點

Spark 目前已成熟，數據處理工具包可以對大量數據集進行處理，且不用擔心底層架構。工具包可以進行數據採集、查詢、處理，也可以進行機器學習，進而建構出分散式系統的數據抽象模型。

處理速度也是 Spark 的優勢之一，MapReduce 在處理過程中將數據放到內存中，而不是放在磁碟上進行持久化，使 Spark 的處理速度提升。

應用方面:

1.數據分析:即對進入的數據流作實時分析。Spark 能夠高效處理來自各數據源的大量數據，支持 HDFS、Kafka、Flume、Twitter 和 ZeroMQ，也能對自定義的數據源進行處理。

2.趨勢數據:Spark 能夠用來對進入的事件流進行處理，用於計算趨勢數據。

3.物聯網:物聯網系統產生了大量的數據，而這些數據會被推往後台處理。Spark 能夠建構出數據管線，在特定的時間間隔內進行轉換，還可以基於一組事件觸發一系列動作。

4.機器學習:由於 Spark 能夠對數據進行批量處理，並且提供機器學習類庫（MLib），因此我們能夠對數據集使用機器學習算法，結合 MLib 與 Streaming 一同使用，就可以構建起機器學習系統。

缺點

1.複雜的部署過程

Spark 支持 Mesos 和 Yarn，但如果對這兩者任一個不熟悉的話，部署過程就會變得艱難。如果不能正確處理，Spark 雖然會單獨運行，但在 cluster 模式下，會遇到發出 Classpath 異常的情況。

2.內存問題

由於 Spark 被用來處理海量數據，因此必須對內存的使用情況進行監控，並對 Spark 內存相關的設置進行調整。

3.頻繁的版本更新導致 API 發生變化

Spark 每三個月就進行一次副版本發布；而每隔三到四個月，進行一次主版本發布。儘管頻繁的版本發布代表推出了更多的功能，但這也意味著這些版本更迭的背後，某些情況下會要求 API 也要發生變化。如果沒有意識到新版本所帶來的變化，就會陷入困境，而若想要確保 Spark 應用不受這些版本更迭變化影響，也會帶來額外的開銷。

4.對 Python 的功能不完善

Spark 支持 Scala、Java 和 Python，但是 Spark 的最新版本中，對 Python 語言 API 的支持不像對 Java 和 Scala 語言的支持完善。如果使用 Spark 最新版本，可能需要用 Scala 或 Java 語言來實現，至少需要檢查是否已經有 Python 版本功能或 API 的實現。

結語

Spark 在運算時，將中間產生的資料暫存在記憶體中，因此可以加快執行速度。尤其需要反覆操作的次數越多，所需讀取的資料量越大，越能看出 Spark 的效能，因此 Spark 適合用來處理大量數據，但如果只是要處理小規模數據，spark 就並非首選，因其處理程序較為複雜，反而會消耗更多的時間。

在這門課學了系統優化、大型系統、分層...等等，而除了這些課內知識，更體會到了實作的樂趣，在做每個作業時就像進入了新世界，了解到課內內容的實際應用。