Spark架構

1. 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。

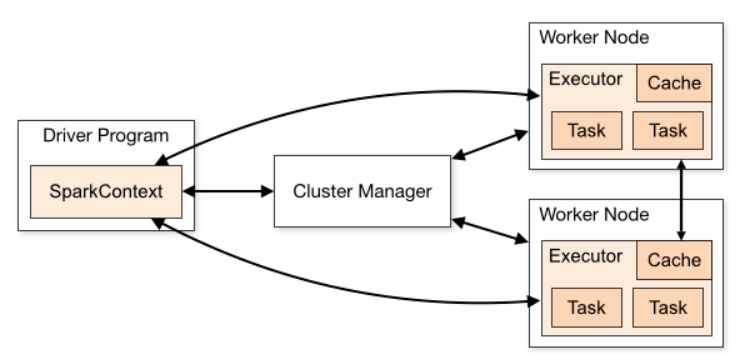
* RDD API：

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

1. 架構圖
2. 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。
3. 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
4. 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。
5. 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
6. Partition：是一種工作單位，每個Task存放在一個Partition中。



1. 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。

1. 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。

1. 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。