資訊管理學系 陳士杰老師

巨量資料技術與應用

Big Data Technology and Application

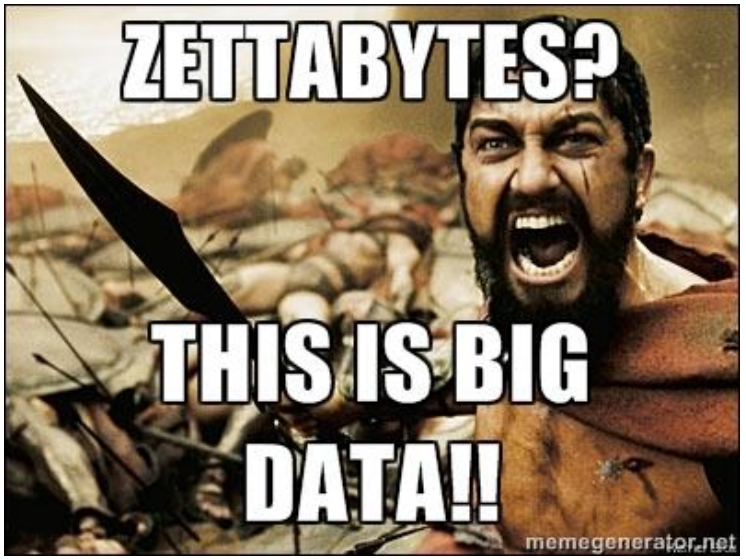
分散式檔案系統HDFS

Introduction to HDFS

國立聯合大學
NATIONAL UNITED UNIVERSITY

大綱

- 分散式檔案系統
- HDFS簡介
- HDFS相關概念
- HDFS體系結構
- HDFS儲存原理
- HDFS Shell操作實踐



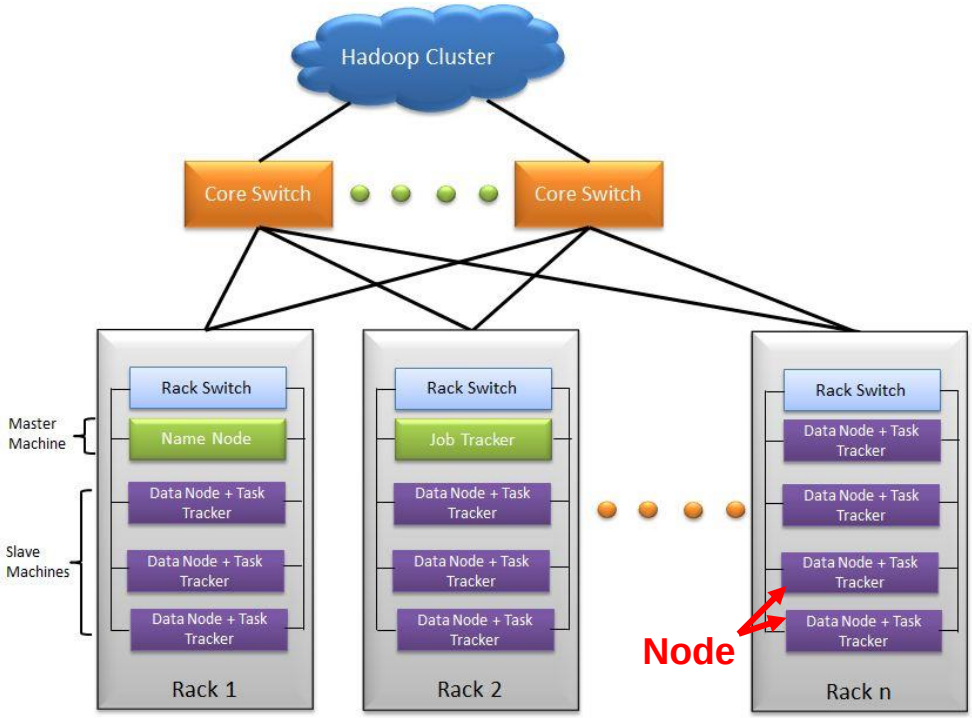
分散式檔案系統

- 電腦集群結構
- 分散式檔案系統的結構

電腦集群結構

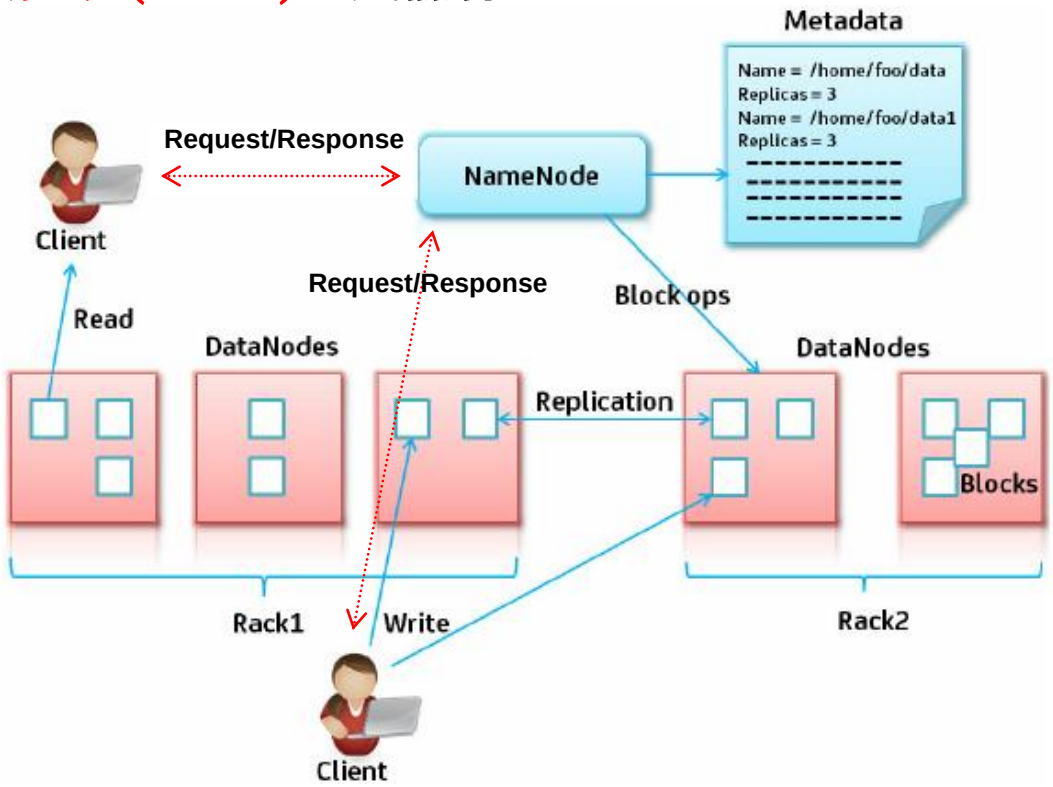
- **分散式檔案系統把檔分散儲存到多個電腦節點上，成千上萬的電腦節點構成電腦集群**
- **與之前使用多個處理器和專用高級硬體的並行化處理裝置不同的是，目前的分散式檔案系統所採用的電腦集群，都是由普通硬體構成的，大幅降低了硬體上的成本**

- 集群中的電腦節點都是放在Rack(機架)
- 每個Rack約可放8~64個節點
- 同一Rack上的不同節點間，透過網路互連
- 不同Rack之間採用另一層級網路或Switch互連



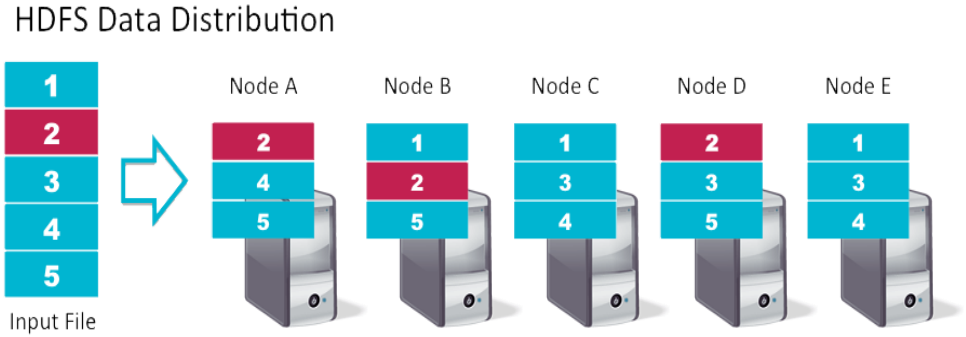
分散式檔案系統的結構

- 分散式檔案系統在實體結構上是由電腦集群中的多個節點構成的，這些節點分為兩類：
  - 一類為主節點(Master Node)或稱名稱節點(NameNode)
  - 另一類為從節點(SlaveNode)或稱資料節點(DataNode)
  - 資料以區塊(Block)型式儲存



HDFS簡介

- Hadoop Distributed File System (Hadoop分散式檔案系統)
  - Hadoop架構下的分散式檔案儲存系統。
  - 於Hadoop的運作框架下，將檔案資料切割成許多區塊 (Blocks)，並分散儲存到不同的電腦主機(節點)上。
  - 資料是以批量處理為主，不會一個區塊一個區塊個別處理。
  - 採用了主/從(Master/Slave)式結構。
  - 可以在廉價電腦系統上執行。



- 問題：
  - 不適合低延遲資料訪問(HDFS即時性不佳)
  - 無法高效儲存大量小檔(檔案太小、索引多/大)

- 它模仿並提供了基於Google File System (GFS) 的所有功能。

The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung
Google*

ABSTRACT

We have designed and implemented the Google File System, a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.

While sharing many of the same goals as previous distributed file systems, our design has been driven by observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system assumptions. This has led us to reexamine traditional choices and explore radically different design points.

1. INTRODUCTION

We have designed and implemented the Google File System (GFS) to meet the rapidly growing demands of Google's data processing needs. GFS shares many of the same goals as previous distributed file systems such as performance, scalability, reliability, and availability. However, its design has been driven by key observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system design assumptions. We have reexamined traditional choices and explored radically different points in the design space.

First, component failures are the norm rather than the

- **HDFS的設計理念：**
  - 將分散式的檔案環境，模擬成**單一的檔案目錄系統**。
  - 異地備份
    - 每個檔案被分割成許多的**區塊(Block)**，每個區塊的大小通常為64MB或128MB。
    - 系統會將每個區塊**複製成許多複本**，並分散儲存於不同的資料節點上。預設是3個複本，用戶可自行調整。
  - 具備硬體錯誤容忍能力
    - 由於是採用一般PC或伺服器，故硬體易出狀況，而HDFS採用**複製資料**以因應硬體的故障。
    - 當偵測到錯誤時，迅速地從複製的資料執行資料回復工作。
  - 處理大規模資料集
    - 支援超過1萬個節點、Perabytes等級的資料量空間需求。

- 資料存取特性
  - Write-once-read-many存取模式，一次寫入，多次存取。
  - 檔案一旦建立、寫入，就不允許修改。
- 在地運算
  - 移動運算到資料端比移動資料到運算端來得成本低。
  - 由於資料的所在位置有被考慮，因此運算工作可以移至距離客戶端最近的資料節點之複本。

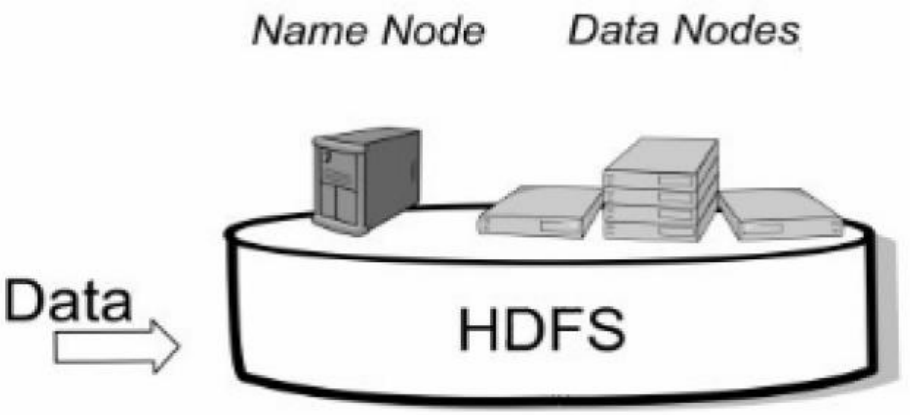
HDFS相關概念

- 區塊(Block)
- 名稱節點(Name Node)
- 資料節點(Data Node)
- 第二名稱節點(Secondary Name Node)

區塊(Block)

- 與傳統的檔案系統一樣，為提升硬碟讀寫的效率，HDFS以資料區塊(Block)為單位，而不是Byte。
- HDFS預設一個區塊是64/128MB，明顯大於傳統的檔案系統。這是為了最小化搜尋成本。
  - HDFS搜尋成本有二：硬碟搜尋成本、資料區塊定位成本
  - 但也不能太大，不然會降低並行處理的速度
- HDFS採用區塊概念可以帶來以下幾個明顯好處：
  - 支援大規模檔儲存：一個大規模檔案可以被分拆成若干個區塊，不同的區塊可以被分發到不同的節點上。因此，一個檔案的大小不會受到單個節點的儲存容量的限制，可以遠遠大於網路中任意節點的儲存容量
  - 簡化系統設計：首先，因為檔案區塊大小是固定的，這樣就可以很容易計算出一個節點可以儲存多少檔案塊，大大簡化了儲存管理；其次，方便了中繼資料(Metadata)的管理，中繼資料不需要和檔案區塊一起儲存，可以由其他系統負責管理中繼資料
  - 適合資料備份：每個區塊都可以冗餘儲存到多個節點上，大大提高了系統的容錯性和可用性

名稱節點(Name Node)和資料節點(Data Node)



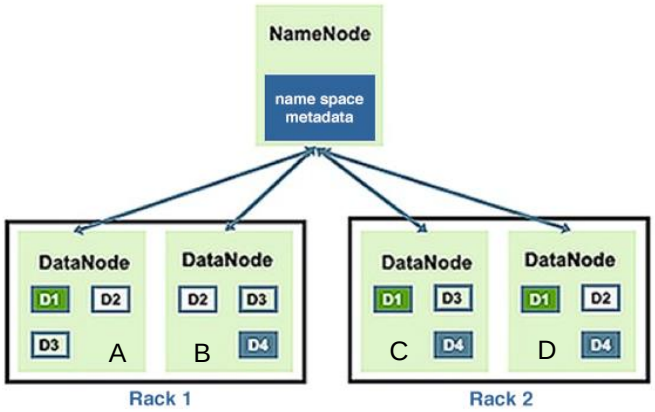
metadata

File.txt=
Blk A:
DN1, DN5, DN6

Blk B:
DN7, DN1, DN2

Blk C:
DN5, DN8,DN9

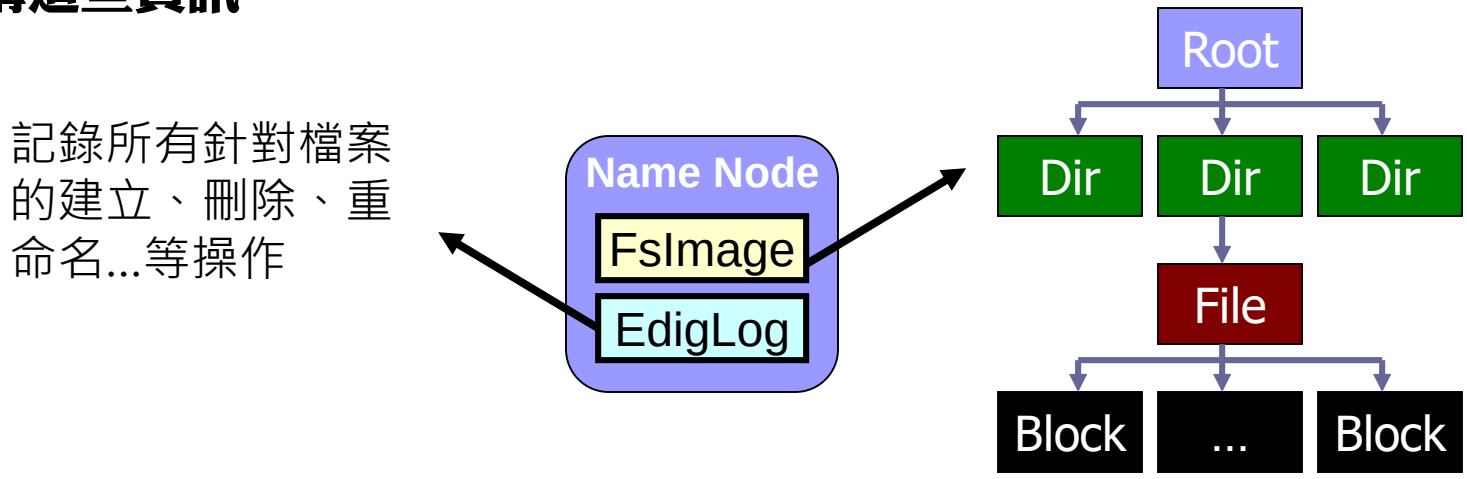
Name Node	Data Node
存放Metadata	存放檔案內容
Metadata保存在記憶體中	檔案內容保存在硬碟中
保存檔案、區塊與data node之間的對應關係	維護Block_id到Data Node本地檔案的對應關係



- **NameNode**
  - 負責記錄與維護HDFS的Metadata
  - 僅有一個NameNode，故NameNode掛掉，會使整個HDFS無法正常存取，致使所有的工作失敗(SPOF – Single Point of Failure)。
- **DataNode**
  - 檔案區塊(Blocks)實際儲存的地方，通常有多部DataNode。
  - DataNode會定期傳送現有的Blocks狀態與清單給NameNode (Heartbeat)。
  - 會根據客戶端或是名稱節點的調度來進行資料的儲存和檢索。
  - 若NameNode發現某個Block之複本數量少於現有的備份設定時，NameNode會自動增加該Block的複本。
  - 當某個DataNode掛掉時，NameNode會自動將此DataNode上的所有Blocks重新配置到其它DataNode上
  - 每個資料節點中的資料區塊，會被保存在各自節點的本地OS檔案系統中

NameNode的資料結構

- 在HDFS中，NameNode負責管理並保存分散式檔案系統的兩個核心資料結構：FsImage和EditLog
  - FsImage用於維護檔案系統樹以及樹中所有的檔案和資料夾的中繼資料
  - 操作日誌檔EditLog中記錄所有針對檔案的建立、刪除、重命名等操作
- 名稱節點記錄了每個檔案中各個區塊所在的資料節點之位置資訊，但不會永久儲存，而是當系統每次重啟時，掃描所有資料節點以重構這些資訊。



FsImage

- FsImage主要存放相關的Metadata，如：檔案的複製等級、修改和存取時間、存取權限、區塊大小以及組成檔案的區塊。
- FsImage不會記錄每個區塊儲存在哪個資料節點。
  - 是由名稱節點把這些對應資訊保留在記憶體的其它區域中，
  - 當資料節點加入HDFS集群時，資料節點會把自己所包含的區塊列表告知名稱節點，此後也會定期執行這種告知操作(by Heartbeat)，以確保名稱節點的區塊對應是最新的。

名稱節點的啟動

- 在名稱節點啟動的時候，它會將FsImage的內容載入到記憶體中，之後再與EditLog中的各項先前操作記錄合併，使得記憶體中的Metadata和實際環境同步。
- 一旦在記憶體中完成上述合併，則利用合併結果建立一個新的FsImage以取代舊的FsImage，並再建立一個空的EditLog。
- 啟動過程中，名稱節點處於“安全模式”，存在記憶體的Metadata僅支援客戶端的讀操作。
- 名稱節點啟動之後，名稱節點會退出“安全模式”，進入正常運作模式。HDFS後續的更新操作會寫到EditLog中。
  - 因為FsImage檔一般都很大（GB級別的很常見），如果所有的更新操作都往FsImage檔中添加，這樣會導致系統運行的十分緩慢，但是，如果往EditLog檔裡面寫就不會這樣，因為EditLog要小很多。

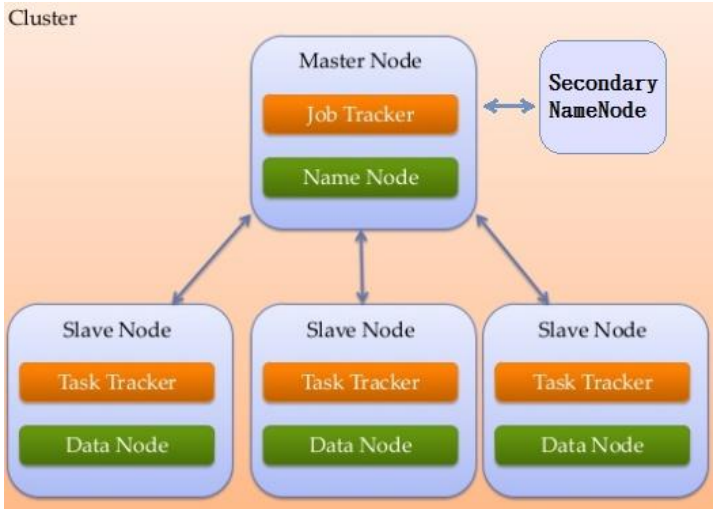
名稱節點不斷變大的問題

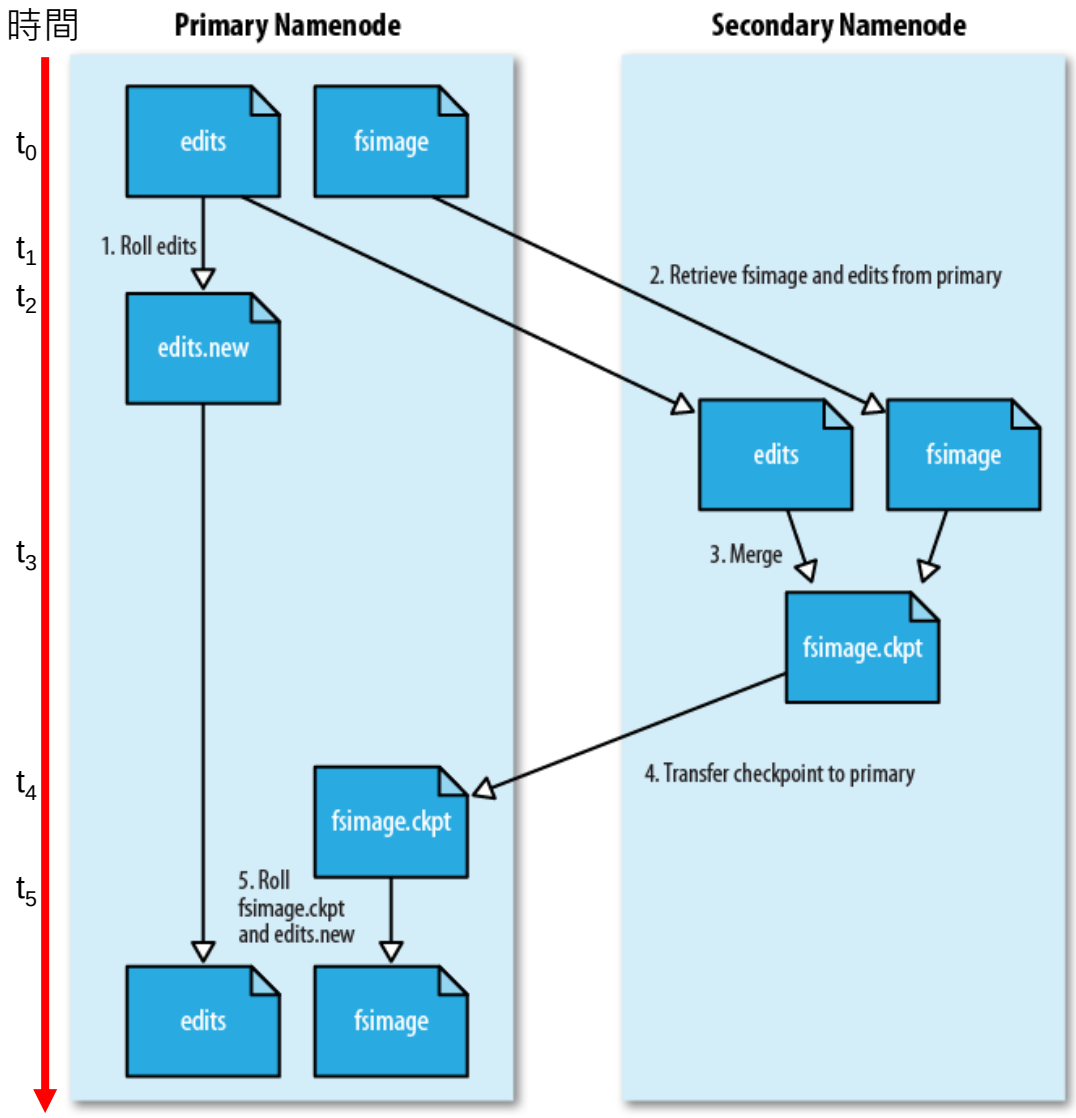
- 在名稱節點運行期間，HDFS的所有更新操作都是直接寫到EditLog中。
- EditLog起初很小，但久而久之， EditLog將會變得很大，大到一定程度還是會影響到HDFS的性能。
- 當名稱節點重啟的時候，名稱節點需要先將FsImage裡面的所有內容載入到記憶體中，然後再一條一條地執行EditLog中的記錄以進行合併。當EditLog非常大的時候，會導致名稱節點啟動操作非常慢，而在這段時間內HDFS系統處於安全模式，一直無法對外提供寫操作，影響了用戶的使用。

如何解決？
Secondary Name Node第二名稱節點

第二名稱節點(Secondary Name Node)

- 第二名稱節點是HDFS架構中的一個組成部分。
- 用來保存名稱節點中對HDFS中繼資料的備份。
- 可以完成FsImage與EditLog的合併操作，減少EditLog大小，縮短名稱節點重啟的時間。
- SecondaryNameNode一般是單獨運行在一台機器上。
- 可做為名稱節點的冷備份/檢查點。





Secondary Name Node工作流程：

(1) Secondary Name Node會定期和Name Node通信，請求其停止使用EditLog，暫時將新的寫操作寫到一個新的edit.new檔案上。這個操作是瞬間完成，上層寫日誌的函數完全感覺不到差別；

(2) Secondary Name Node透過HTTP GET方式從Name Node上獲取到FsImage和EditLog，並下載到本地相對應目錄下；

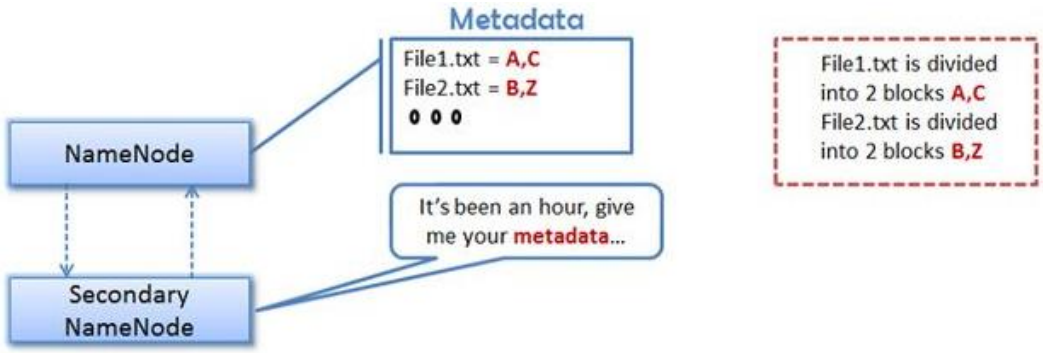
(3) Secondary Name Node將下載下來的FsImage載入到記憶體，然後一條一條地執行EditLog中的各項操作，使得記憶體中的FsImage保持最新；這個過程就是EditLog和FsImage合併；

(4) Secondary Name Node執行完（3）操作之後，會透過post方式將新的FsImage文件發送到Name Node節點上

(5) Name Node將從Secondary Name Node接收到的新的FsImage替換成舊的FsImage，同時將edit.new替換成EditLog，透過這個過程EditLog就變小了

- 上述的工作流程中：
  - Secondary Name Node周期性地備份Name Node的Metadata，相當於Name Node的Check Point(檢查點)。
  - 在時間點t_1進行合併以產生新FsImage，但是在時間點t_1以後所有在Name Node新產生的更新操作是被另記錄於edits.new中，無法反應到此次所產生的新FsImage中。故當Name Node發生故障時，雖可由Secondary Name Node進行系統恢復，但仍會遺失部份的Metadata。
  - 在HDFS設計中，不支援把系統直接切換到Secondary Name Node，因此無法做到熱備份。

如何改善？
High Availability (HA)
~另開單元~



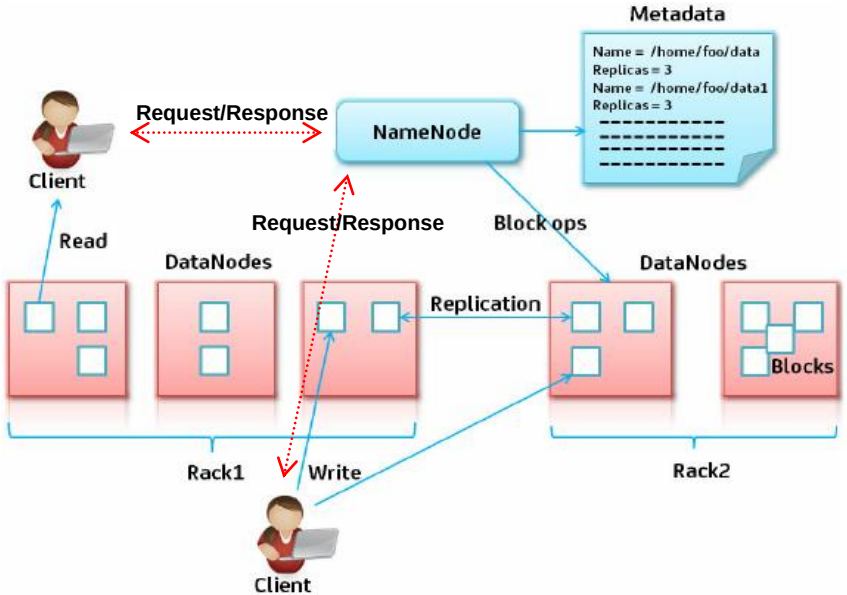
- Not a hot standby for the NameNode
- Connects to NameNode every hour(its configurable)
- Housekeeping, backup of NameNode metadata
- Saved metadata can rebuild a NameNode

HDFS體系結構

- HDFS體系結構概述
- HDFS命名空間管理
- 通信協定
- 客戶端
- HDFS體系結構的局限性

HDFS體系結構概述

- HDFS採用了主從(Master/Slave)架構模型，一個HDFS集群包括一個名稱節點(NameNode)和若干個資料節點(DataNode)。
- 名稱節點作為中心伺服器，負責管理檔案系統的命名空間及客戶端對檔案的訪問。
- 集群中的資料節點一般是一個節點運行一個資料節點行程，負責處理檔案系統客戶端的讀/寫請求，在名稱節點的統一調度下進行資料區塊的建立、刪除和複製等操作。
- 每個資料節點的區塊實際上是保存在本地Linux檔案系統中



HDFS命名空間管理

- HDFS的命名空間(Namespace)包含目錄、檔案和區塊
- 在HDFS1.0體系結構中，在整個HDFS集群中只有一個命名空間，並且只有唯一一個名稱節點，該節點負責對這個命名空間進行管理
- HDFS使用的是傳統的分級(層)檔案體系。因此，用戶可以像使用普通檔案系統一樣，建立、刪除目錄和檔案，在目錄間轉移檔案，重命名檔案等

【註】Namespace：主要是讓大家避免使用相同的名稱來為資料或物件命名的機制。例如：假設學校禁止同名同姓的學生在同一班(避免命名衝突)，當有同名同姓的兩個學生可能被分到同一班時，強制要求分到不同班(不同命名空間)，這樣就不會弄錯人了。

通信協定

- HDFS是部署在集群上的分散式檔案系統，因此，很多資料需要透過網路進行傳輸
- 所有的HDFS通信協定都是建構在TCP/IP協定基礎之上
- 客戶端透過一個可配置的Port向名稱節點主動發起TCP連接，並使用客戶端協定與名稱節點進行互動
- 名稱節點和資料節點之間則使用專門的資料節點協定進行互動
- 客戶端與資料節點的互動是透過RPC（Remote Procedure Call）來實現的。在設計上，名稱節點不會主動發起RPC，而是回應來自客戶端和資料節點的RPC請求

客戶端

- 客戶端是用戶操作HDFS最常用的方式，HDFS在部署時都提供了客戶端
- HDFS客戶端是一個程式庫，顯示了HDFS檔案系統介面，這些介面隱藏了HDFS實現中的大部分複雜性
- 嚴格來說，客戶端並不算是HDFS的一部分
- 客戶端可以支援打開、讀取、寫入等常見的操作，並且提供了類似Shell的命令列方式來訪問HDFS中的資料
- 此外，HDFS也提供了JavaAPI，作為應用程式訪問檔案系統的客戶端程式設計介面

HDFS體系結構的局限性

- HDFS**只設置唯一一個名稱節點，這樣做雖然大大簡化了系統設計，但也帶來了一些明顯的局限性，具體如下：**
  - (1) **命名空間的限制**：名稱節點所有的Metadata是保存在**記憶體**中的，因此，名稱節點能夠容納的物件(檔案、區塊)的個數會受到**記憶體空間大小的限制**。
  - (2) **性能的瓶頸**：整個分散式檔案系統的輸送量，受限於**單一個名稱節點的輸送量**。
  - (3) **隔離問題**：由於集群中只有一個名稱節點，只有**一個命名空間**，因此，無法對不同應用程式進行隔離。
  - (4) **集群的可用性**：一旦這個唯一的名稱節點發生故障，會導致整個集群變得不可用(**單點故障**)。

HDFS儲存原理

- 冗餘資料保存
- 資料存取策略
- 資料錯誤與恢復

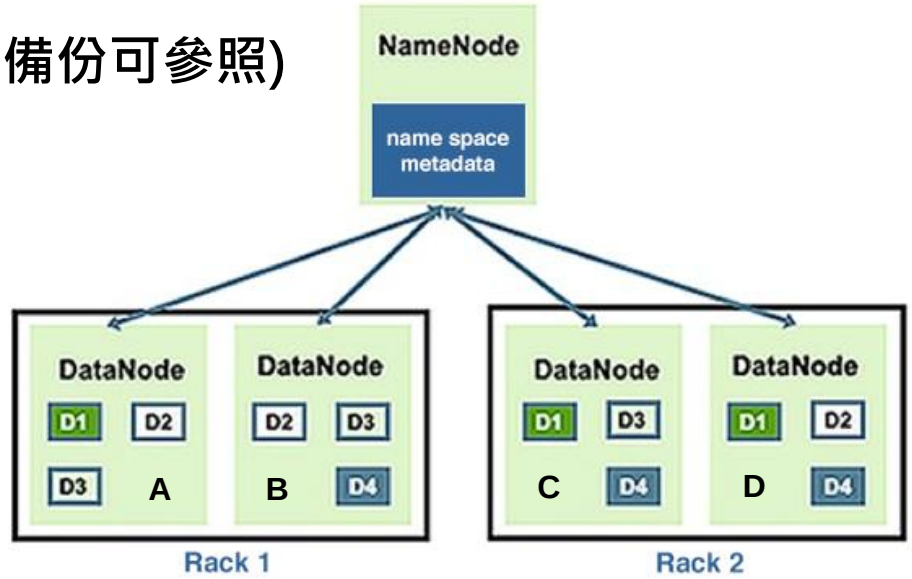
冗餘資料保存

作為一個分散式檔案系統，其資料皆儲存在廉價機器上，可能經常出狀況。為了保證系統的容錯性和可用性，HDFS採用了多副本方式對資料進行冗餘儲存(一般預設是3份)，通常一個資料塊的多個副本會被分佈到不同的資料節點上，如下圖所示，資料塊D1被分別存放到資料節點A、C、D上，資料塊D2被存放在資料節點A、B和D上。這種多副本方式具有以下幾個優點：

（1）加快資料傳輸速度

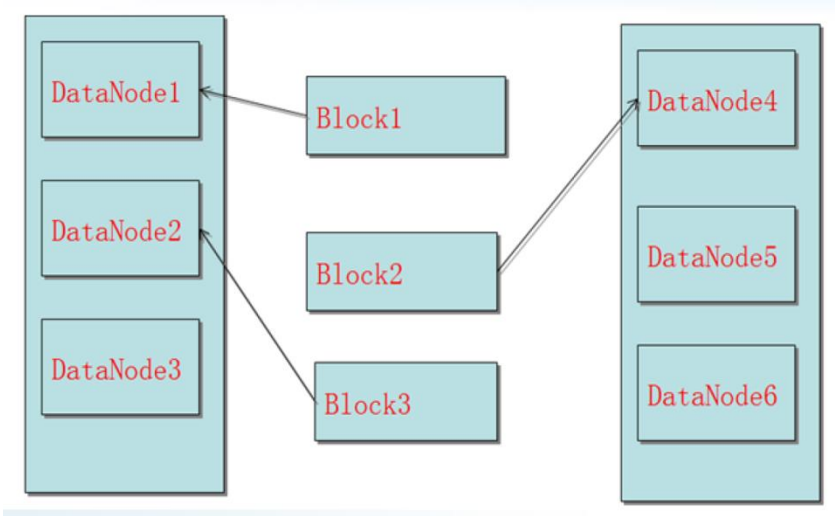
（2）容易檢查資料錯誤(因為有備份可參照)

（3）保證資料可靠性



資料存取策略

- 資料存放
  - 第一個副本：
    - 如果是集群內某DataNode發起的寫入操作，則就近放置在該資料節點以實現就近寫入資料
    - 如果是集群外發起的寫入操作，則在集群內隨機挑選一台硬碟不太滿、CPU不太忙的資料節點存放
  - 第二個副本：放置在與第一個副本不同的機架的節點上
  - 第三個副本：與第一個副本相同機架的其他節點上
  - 更多副本：從集群中隨機挑選節點



資料讀取

- HDFS提供了一個API可以確定一個資料節點所屬的機架ID，客戶端也可以調用API獲取自己所屬的機架ID
- 當客戶端讀取資料時，從名稱節點獲得資料塊不同副本的存放位置列表，列表中包含了副本所在的資料節點，可以調用API來確定客戶端和這些資料節點所屬的機架ID，當發現某個資料塊副本對應的機架ID和客戶端對應的機架ID相同時，就優先選擇該副本讀取資料，如果沒有發現，就隨機選擇一個副本讀取資料

資料錯誤與恢復

- **由於HDFS可以運作於廉價的硬體，它把硬體出錯視為常態，而非異常，因此設計上具有較高的容錯性。**
- **HDFS具有相對應的機制檢測資料錯誤和進行自動恢復，主要包括以下幾種情形：名稱節點出錯、資料節點出錯和資料出錯。**

1. 名稱節點出錯

- 名稱節點保存了所有的中繼資料資訊，其中，最核心的兩巨量資料結構是FsImage和Editlog，如果這兩個檔案發生損壞，那麼整個HDFS實例將失效。
- 因此，HDFS設置了備份機制，把這些核心檔同步複製到備份伺服器Secondary Name Node上。
- 當名稱節點出錯時，HDFS會暫停一段時間，根據備份伺服器Secondary Name Node中的FsImage和Editlog資料進行恢復 (屬於冷備份，HDFS 2.0已克服此問題)。

2. 資料節點出錯

- 每個資料節點會定期向名稱節點發送“心跳(Heartbeat)”資訊，向名稱節點報告自己的狀態
- 當資料節點發生故障，或者網路發生斷網時，名稱節點就無法收到來自一些資料節點的心跳資訊，這時，這些資料節點就會被標記為“當機”，節點上面的所有資料都會被標記為“不可讀”，名稱節點不會再給它們發送任何I/O請求
- 此時，由於一些資料節點不可用，會導致一些**資料區塊的副本數量小於冗餘因數**
- 名稱節點會**定期檢查**這種情況，一旦發現某個資料塊的副本數量小於冗餘因數，就會啟動資料冗餘複製，為它生成新的副本
- HDFS和其它分散式檔案系統的最大區別就是可以調整冗餘資料的存放位置，用以平衡不同機器的負載。

3. 資料出錯

- 網路傳輸和硬碟錯誤等因素，都會造成資料錯誤
- 客戶端在讀取到資料後，會採用md5和sha1對資料區塊進行校驗，以確定讀取到正確的資料 (採用驗證碼校驗的方式)
  - 在**檔案建立**時，客戶端就會對每一個檔案區塊進行資訊摘錄，並把這些資訊寫入到**同一個路徑的隱藏檔**裡面
  - 當**客戶端讀取檔案**的時候，會先讀取該資訊檔，然後，利用該資訊檔對每個讀取的資料區塊進行校驗。如果校驗出錯，客戶端就會請求到另外一個資料節點讀取該區塊，並且向名稱節點報告這個區塊有錯誤，名稱節點會定期檢查並且重新複製這個區塊

HDFS Shell操作實踐

在此僅做簡要說明，詳細使用請見本教材線上講義：

- **實務操作-單元03：HDFS簡易操作**

課程網頁：http://debussy.im.nuu.edu.tw/sjchen/BigData_final.html
教師網頁：http://web.nuu.edu.tw/~sjchen

- Hadoop提供了關於HDFS在Linux作業系統上進行檔案操作的常用Shell命令以及JavaAPI。同時還可以利用Web介面查看和管理Hadoop檔案系統。

  【註】Hadoop安裝成功後，已經包含HDFS和MapReduce，不需要額外安裝。而HBase等其他元件，則需要另外下載安裝。

- 在學習HDFS程式設計實踐前，我們需要啟動Hadoop。執行如下命令：

```
$ cd /usr/local/hadoop
$ ./sbin/start-dfs.sh
```

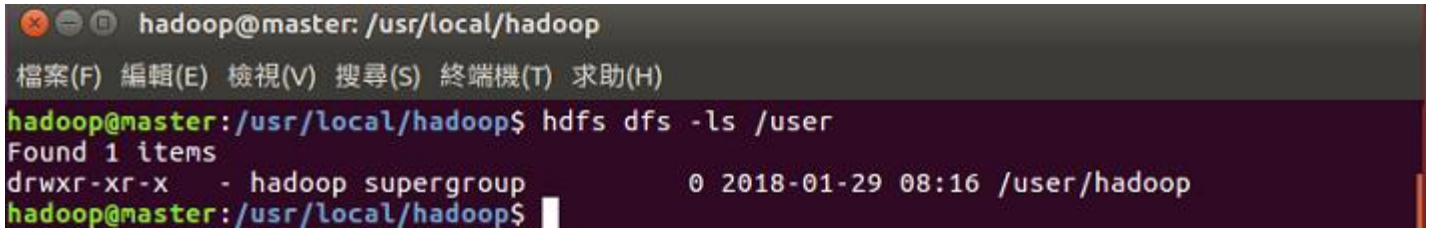
HDFS常用shell命令

- HDFS有很多shell命令，可以查看HDFS檔案系統的目錄結構、上傳和下載資料、建立文件等。
- Hadoop中有三種HDFS Shell命令方式：
  - hadoop fs：適用於任何不同的檔案系統，比如本地檔案系統和HDFS檔案系統
  - hadoop dfs：只能適用於HDFS檔案系統 (不建議使用)
  - hdfs dfs：跟hadoop dfs的命令作用一樣，也只能適用於HDFS檔案系統
- 以hdfs dfs為例，該命令的通用格式為：

  hdfs dfs [genericOptions] [commandOptions]

實例：

hdfs dfs -ls <path>: 顯示<path>指定的檔案之詳細資訊



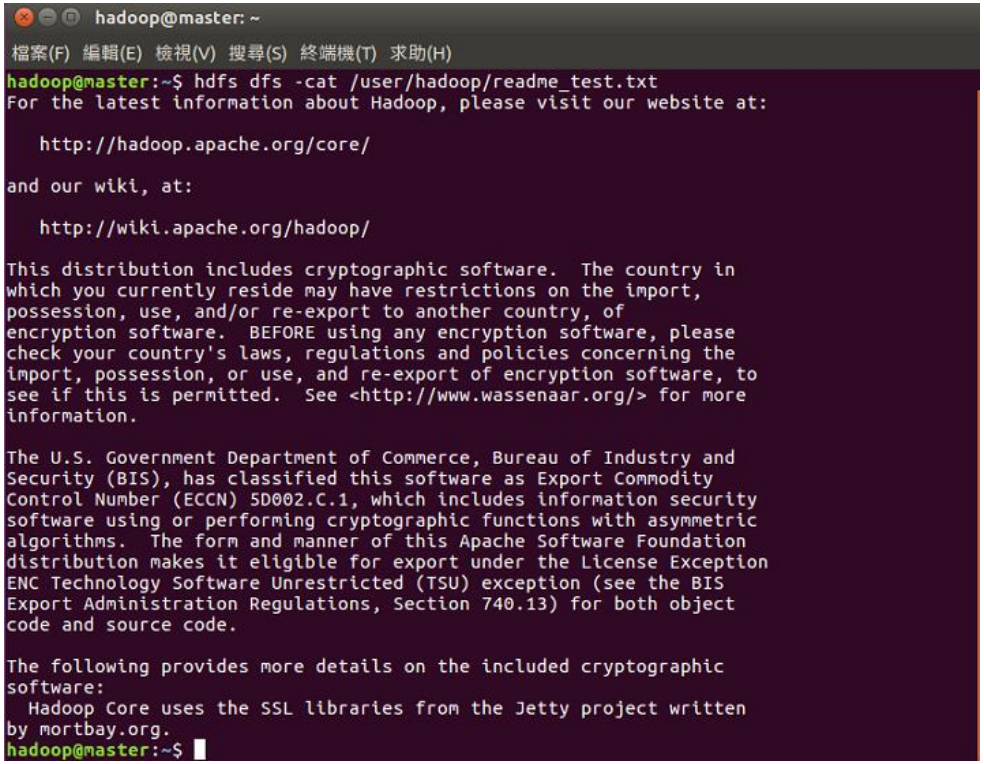
- hadoop fs版本...

hadoop fs -ls <path>

hdfs dfs -mkdir <path>:在指定的<path>建立新的資料夾

實例：

hdfs dfs -cat <path>: 將<path>指定的檔案內容輸出到標準輸出（stdout，即：螢幕）

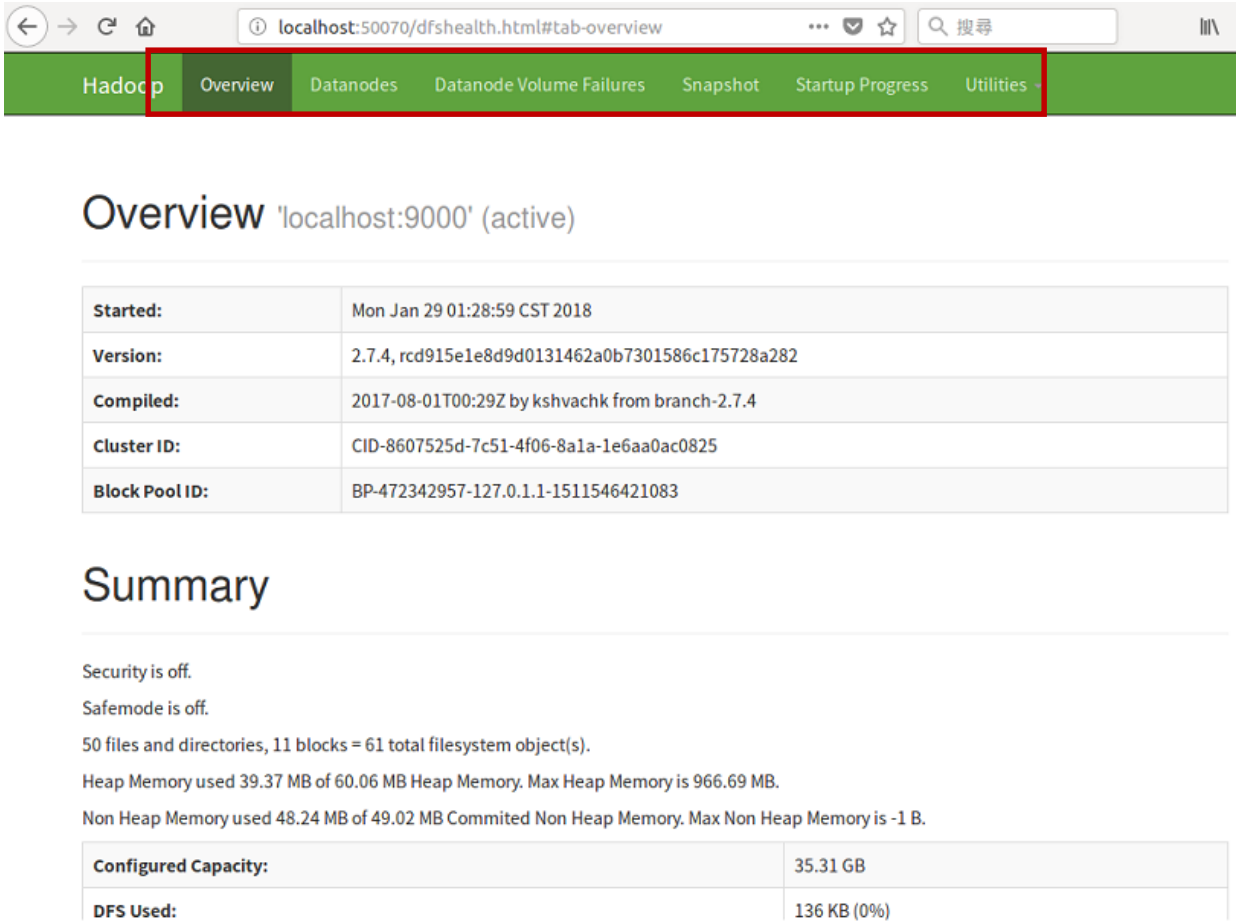


hdfs dfs -copyFromLocal <localsrc> <dst>: 將本地原始檔案<localsrc>複製到HDFS指定路徑<dst>的資料夾中

HDFS的Web介面

在配置好Hadoop集群之後，可以透過瀏覽器登錄

“http://<NameNodeIP>:50070” 訪問HDFS檔案系統



本章小結

- 分散式檔案系統是巨量資料時代解決大規模資料儲存問題的有效解決方案
- HDFS具有相容廉價的硬體設備、巨量資料集、簡單的檔案模型、強大的跨平臺相容性等特點。但是，也要注意到，HDFS也有自身的局限性，比如不適合低延遲資料訪問、無法高效儲存大量小檔和不支援多用戶寫入及任意修改檔等
- 區塊是HDFS核心的概念，一個大的檔會被拆分成很多個區塊。具有支援大規模檔儲存、簡化系統設計、適合資料備份等優點
- HDFS採用了主從（Master/Slave）結構模型，一個HDFS集群包括一個名稱節點和若干個資料節點。名稱節點負責管理分散式檔案系統的命名空間；資料節點負責資料的儲存和讀取
- HDFS採用了冗餘資料儲存，增強了資料可靠性，加快了資料傳輸速度。HDFS還採用了相對應的資料存放、資料讀取和資料複寫原則，來提升系統整體讀寫回應性能。HDFS把硬體出錯看作一種常態，設計了錯誤恢復機制
- 本章最後介紹了HDFS的資料讀寫策略以及HDFS Shell操作方面的相關知識

-本單元結束-
感謝您的聆聽