Paper Title： The Google File System

Student Name：康智詠

Department/Year：資工四甲

Student ID：406261523

Date: 2020/10/16

Q1. What is the problem the authors are trying to solve?

主要在闡述Google File System的設計理念，如何運作及呈現來自micro-benchmarks及read world的測量結果

Q2. What other approaches or solutions existed at the time that this work was done?

Boxwood project某些部分與Chubby、GFS、Bigtable相同，例如分散式協議、locking、分散式chunk storage、分散式B-tree storage

Oracle’s Real Application Cluster database 共享磁碟來儲存資料以及分散式lock管理者

IBM’DB2 Parallel Edition 的架構與Bigtable相似，是依據shared-nothing

Q3. What was wrong with the other approaches or solutions?

1. Boxwood project的目標是提供基礎設施給高階的服務
2. 藉由分散式B-tree或分散式hash table所實作出的key-value模型有許多限制

Q4. What is the authors' approach or solution?

**Interface**

1. 檔案用目錄分層組織且用路徑名稱來辨識。
2. 提供常用的操作進行創造、刪除、開啟、關閉、讀取、寫入檔案
3. Use snapshot creates a copy of file or a directory tree at low cost
4. Record append 可以允許多個client同時附加資料到相同檔案，保證每個client附加資料的原子性

**Architecture**

1. GFS cluster由一個server與多個chunkservers組成，可以被多個client訪問
2. 檔案會被切割成多個適當大小的chunk，當chunk被創造時，master指派獨特的64 bit chunk handle辨別每個chunk
3. 每個chunk會有多個複製品存放在多個chunkservers以提升可靠性
4. 由master保存所有檔案系統的metadata，定期與每個chunkserver進行交流、下達指令或蒐集狀態
5. Client或chunkserver皆沒有快取檔案資料

**Single Master**

Client經由詢問master來得知哪個chunkserver可以聯繫

**Chunk Size**

採取64 MB的大小，比典型的檔案系統block大小來的大

**Metadata**

* 1. Master主要儲存三種metadata，檔案、 chunk namespace(檔案映射到的chunk)、每個chunk的複製品所在位置。
  2. 所有的metadata皆保存在master的記憶體當中。主要分為兩種類別，第一種在master的本地磁碟中註冊以永久保存，第二種被複製到遠端機器進行保存。
  3. Master並不會儲存chunk永久儲存chunk所在位置的資訊，而是每當chunkserver加入cluster時，經由詢問每個chunkserver取得。

Q5. Why is it better than the other approaches or solutions?

**Architecture**

擁有單一的master可以簡化設計使master可以將chunk放置良好，並利用對全域的認識進行複製決策

**In-Memory Data Structures**

Metadata儲存在記憶體，讓master操作比較快。使得master比較簡單有效率在後台定期掃描整個狀態

**Chunk Size**

1. 減少client與master的互動
2. client可以在Chunk中進行許多操作，減少占用TCP連線的時間
3. 減少metadata在master占用的空間

**Metadata**

使用log可以容易、可靠更新master的狀態，當master崩潰的時候沒有不同步的風險

Q6. How does it perform?

**Leases and Mutation Order**

Control flow：

1. Client詢問master哪個chunkserver有該chunk的lease，如果都沒有，則給複製品所選擇的
2. Master回應primary的身分以及其他複製品的所在位置
3. Client將資料交給所有的複製品
4. 一旦所有的複製品收到資料，client將會傳送寫入的請求給primary
5. Primary會將寫入請求轉發給所有次等複製品
6. 所有次等複製品會告知primary它們已經完成操作
7. Primary將回報給client

**Master Operation**

**Namespace Management and Locking**

* 1. 允許多個操作可以啟動並使用lock鎖住namespace正在被使用的地方來確保正確的序列
  2. GFS 邏輯表示它的namespace作為lookup table將路徑名稱映射到metadata

**Replica Placement**

Spread chunk replicas across racks.以確保某些chunk的複製品即使損毀或離線也可以存活和使用

**Creation, Re-replication, Rebalancing**

1. 當master create a chunk時，會考慮許多因素來選擇複製品
2. 一旦複製品低於使用者的期望數量，master就會re-replicate a chunk
3. Master會經常rebalance複製品。當發現複製品的有更好的分布時，會移動複製品，以移動到更好的磁碟空間和平衡負載能力

**Garbage Collection**

在檔案被刪除之後，GFS不會馬上回收儲存空間，僅是被更改為可以隱藏的名稱。Master會經常掃描檔案系統的namespace，如果隱藏後超過三天就會被刪除

**Fault Tolerance And Diagnosis**

1. **Fast Recovery**

不會區分是正常還是不正常的關閉，server會藉由殺掉process

定期關閉

1. **Chunk Replication**

每個chunk會被複製到多個 chunkserver的不同rack。使用者可以將不同部分的file namespace指定成不同等級。當 chunkserver斷線或用checksum偵測到corrupt複製品時master可以利用現存需要的複製品來進行修復

1. **Master Replication**

當primary master 關閉時，Shadow masters提供唯獨的權力給檔案系統

(4) **Data Integrity**

1.每個chunkserver使用checksummming偵測corrupt的儲存資料。

2. 利用複製品修復corrupt的檔案

3. 在回覆要求者之前，chunkserver會利用checksum驗證讀取範圍內的data blocks

4. 每個chunkserver獨立利用checksum驗證確保完整性

(4) **Diagnostic Tools**

GFS使用diagnostic logs紀錄許多重大的事件以及所有RPC的請求和回應，而RPC logs包含在網上發送的請求及回應。用這些紀錄可以重新建構出整個互動的歷史，並藉此診斷問題。

Q7. Why is this work important?

設計一套高容錯能力的file system。設計理念與以往不同。將發生錯誤視為很正常的事情。檔案大小逐漸變大，過往的file system已經很難支援。大部分的檔案修改藉由附加新資料而不是覆寫現有資料。共同設計應用程式以及檔案系統API有助於增加整個系統的彈性

Q8. Can any improvement be done?

多個使用者可以同時讀取同一個檔案，但是必須劃分後才能讀取