The Google File System

學號：R07944058　系級：網媒碩一　姓名：陳鵬宇

1 Summary

此篇論文是由 Google 在 2003 年發表的，旨在說明 Google 是如何在超大型的分散式系統設計它們的 Google File System (GFS)。Google 需要滿足非常大量使用者的各種讀、寫 (overwrite 或 append) 等需求，同時又必需兼顧了效能、可擴展性、可靠性、可用性等等。

GFS 是在對 Google 的工作量進行嚴格觀察後設計的，旨在提供高聚合性能，和在廉價硬體上的容錯率。它具有單個 master 和多個 chunkservers，以最大限度地減少 master 的負荷。

設計的重點主要有四點：第一，當資料損壞時如何復原，例如：不斷監控、錯誤偵測、容錯等做法；第二，檔案大小通常都很巨大，GB 量級是常有的事，因此選擇過小的 block size 是不切實際的，必需選擇一個合理的 block size，本文在 2003 年的時間點選擇了 64MB 的大小；第三，由於 write 可再進一步區分成完全的覆寫原檔案，或僅僅只是在原本的檔案後面再 append；第四，如何保有靈活性，且多個 clients 同時 append 同一個檔案時，能達到 atomic。

總結來說，GFS 妥善處理了以下幾種情況（傳統檔案系統無法處理的）：有效率的處理不適合超大型數據集的單一硬碟、低成本、對 write 中的 append 操作特別優化。

2 Methodology

GFS 的架構為一個 single master 加上多個 chunkservers 和多個 clients 進行訪問。檔案會被切成許多固定大小的 chunks，並且預設會有三份副本 (replicas)。Master 包含 metadata，像是 namespace、access control information、mapping from files to chunks、chunks' locations 等。Client 主要和 master 溝通，但若和 data-bearing 有關則會直接和 chunkservers 溝通。

最主要的架構論文中給了很清楚的流程圖，client 透過跟 master 請求 (file name, chunk index)，master 回傳給 client (chunk handle, chunk locations) 後，client 透過剛從 master 那裡拿到的資訊再向 chunkserver 拿 chunk data。同時 master 要「週期性地」跟 chunkservers 做溝通去給予指令和蒐集 chunkservers 的狀態。Chunk size 的設計也非常重要，Google 選擇的 64MB 有三大好處，其一，它降低了 client 和 master 溝通的頻率，其二，降低了網路負擔，其三，降低了 master 需要存的 metadata。

GFS 採用了 relaxed consistency model 以支持他高度分散的應用。一連串成功的修改檔案後，修改的檔案還能確保是 defined 且保有最有一次修改後的資料。GFS 透過在所有副本上以相同的順序將改變應用於 chunk 以及使用 chunk version number 來檢測任何已經過時的副本來達成此要求。

GFS 使用 lease 來維護跨副本的一致 mutation 順序。Master 將一個 chunk release 授予其中一個副本，我們將其稱為主副本 (primary)，而所有的 write control 和 data flow 都必需根據此主副本和其它副本們溝通。

當 master 收到快照 (snapshot) 請求時，它首先撤銷它即將快照的文件中的任何未完成 lease。當 lease 被撤銷或過期後，master 將操作記錄到磁碟。然後，它通過複製源文件或 directory tree 的 metadata 將此 log record 應用於其 in-memory state。

3 Contribution

論文有幾個重要貢獻，他提出了 master、chunkserver 的架構，大副度降低了 master 的負荷，此外，分配給多個叢集單一的 master 使得整體溝通變得簡單。

保存 chunks 的副本提高了在分散式環境下的讀取頻寬。Lookup table 和 prefix compression 提供了更快由 filename mapping 到 chunkserver 的過程，而不是使用傳統的目錄結構。基於觀察，有大量的 append 動作，因此使用 copy on write 的機制。Lazy garbage collection 提高整體 throughput，不會因為訪問單個獨立的 chunk 就犧牲掉大家的時間。Block checksuming 確保了 chunks 的一致性。Operation log 提供了 master 復原的機制。

論文也多次提到了 HeartBeat messages，之所以分成 master 對應多個 chunkservers 就是希望 master 不用存取所有資料，而是存取 metadata，而 HeartBeat messages 可以讓 master 保持著知道每個 chunkservers 目前的狀態，以方便日後 client 向 master 請求時，master 能正確的告訴 client 該去跟哪個 chunkserver 拿資料，同時 master 在刪除資料時，雖然他會馬上 log 此訊息，但真實的資料分配卻不是立即地發生，master 的做法是先重新命名成某保留字，再透過週期性的 HeartBeat message 去「真實地」刪除，這樣的好處是，master 能把時間挪用給更急迫的 client 需求，而垃圾回收沒那麼急迫，可以晚一點再執行就好。

4 Conclusion

此篇論文透過傳統檔案系統的慣例，延伸至 GFS 這種超大型的分散式檔案系統，提供了讀者有別於一般檔案系統不同的實現方式。將所有的 metadata 都放在 master 的主記體憶，意謂著所有的操作都會變得更快，但這可能會因為主記憶體的大小而限制整個叢集大小。本文選擇將 metadata 存在主記憶體中，因為增加額外的記憶體是更好的解決方式。

另外論文中還有用到不少技術，像是 master 批次處理數個 operation log 記錄以降低 flushing 造成的影響、分離 data flow 和 control flow 以提高推送副本的性能、一定時間內，在 client 端快取 metadata 等。

論文中有一些取捨，例如：使用較大的 chunk size 雖然能有效的降低 metadata 的數量，但同時也因為 fragmentation 浪費了空間，但 GFS 之所以還是選擇用較大的 chunk size 就是因為他有很大的好處：降低了 client 和 master 溝通上的負荷，也減輕了網路的負擔。